The combination of Hebbian and predictive plasticity learns invariant object representations in deep sensory networks

Manu Srinath Halvagal[1,2] and Friedemann Zenke[1,2,*]

[1]Friedrich Miescher Institute for Biomedical Research, 4058 Basel, Switzerland
[2]Faculty of Natural Sciences, University of Basel, 4033 Basel, Switzerland
[*]Correspondence: friedemann.zenke@fmi.ch

Abstract

Discriminating distinct objects and concepts from sensory stimuli is essential for survival. Our brains accomplish this feat by forming disentangled internal representations in deep sensory networks shaped through experience-dependent synaptic plasticity. To elucidate the principles that underlie sensory representation learning, we derive a local plasticity model that shapes latent representations to predict future activity. This Latent Predictive Learning (LPL) rule conceptually extends Bienenstock-Cooper-Munro (BCM) theory by unifying Hebbian plasticity with predictive learning. We show that deep neural networks equipped with LPL develop disentangled object representations without supervision. The same rule accurately captures neuronal selectivity changes observed in the primate inferotemporal cortex in response to altered visual experience. Finally, our model generalizes to spiking neural networks and naturally accounts for several experimentally observed properties of synaptic plasticity, including metaplasticity and spike-timing-dependent plasticity (STDP). We thus provide a plausible normative theory of representation learning in the brain while making concrete testable predictions.

Introduction

Recognizing objects and concepts from sensory inputs is crucial for perception. To that end, brains must effectively distinguish between highly entangled stimuli. For instance, the activity patterns that the retinal ganglion cells in our eyes send to the brain in response to viewing a cat or a dog may, in some cases, be more similar than two different views of the same dog (Fig. 1a). Yet, we have no problem distinguishing cats from dogs because our brains are able to internally represent them as separate objects or categories that remain invariant to different views and their context. What processing enables such invariant object recognition?

Visual stimuli evoke distinct activity patterns in individual sensory neurons that correspond to points in a high-dimensional space spanned by the neuronal activity levels. All possible stimuli of one category, for instance images of dogs, lie within a subregion of this space, a manifold, whereas other categories lie on different manifolds. However, for many stimuli, the corresponding manifolds are *entangled* like crumpled-up sheets of paper, which makes it impossible for downstream neurons to decode the underlying stimulus category through simple linear combinations of their inputs. *Disentangling* behaviorally relevant categories therefore requires deep sensory networks that can extract invariant linearly separable category representations, such as those found in the visual system (Fig. 1b; [3]). Crucially, successful disentangling requires specific network connectivity, which is thought to be shaped through experience-dependent

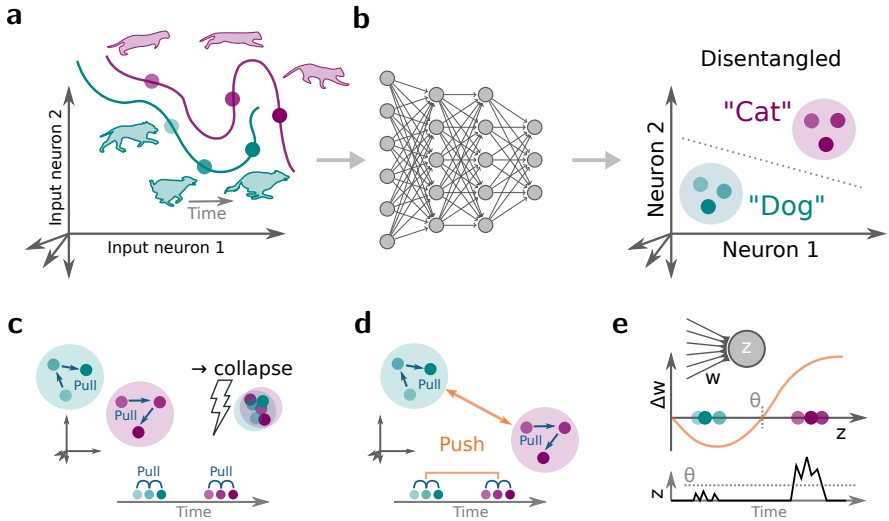

Figure 1: **Disentangling sensory stimuli with plastic neural networks.** **(a)** Schematic of neuronal responses at the sensory input level. The neuronal response patterns of different stimuli correspond to points in a high dimensional space. The response patterns from different stimulus classes, e.g., cats and dogs, form a low-dimensional manifold in the space of all possible response patterns. Generally, different class manifolds are entangled, which means that the stimulus identity cannot be readily decoded from a linear combination of the neuronal activities. **(b)** Sketch of a deep neural network (left) that transforms inputs into disentangled internal representations that are linearly separable (right). **(c)** Schematic of how predictive learning influences latent representations (left). Learning tries to "pull" together representations that frequently co-occur close in time (bottom). However, without opposing forces, such learning dynamics lead to "representational collapse" whereby all inputs are mapped to the same output and thereby become indistinguishable (right). **(d)** Self-supervised learning (SSL) avoids collapse by adding a repelling force that acts on temporally distant representations that are often semantically unrelated. **(e)** Plot of postsynaptic neuronal activity z over time (bottom) and the Bienenstock-Cooper-Munro (BCM) learning rule (top; [1, 2]) which characterizes the sign and magnitude of synaptic weight change Δw as a function of postsynaptic activity z. Notably, the sign of plasticity depends on whether the evoked responses are above or below the plasticity threshold θ. Using the example of Neuron 1 in panel (b), the BCM rule potentiates synapses that are active when a "Cat" stimulus is shown, whereas "Dog" stimuli induce long-term depression (LTD). This effectively pushes the evoked neuronal activity levels corresponding to both stimuli away from one another, and thereby prevents representational collapse.

plasticity [4]. However, current data-driven plasticity models are unable to account for the emergence of disentangled representations in deep biological neural networks.

In contrast, artificial deep neural networks (DNNs) used in machine learning *do* yield highly disentangled representations by dint of supervised training algorithms. These algorithms gradually transform the output of a neural network model to match a set of targets or "labels" associated with given input data. Over the last decade, the combination of supervised learning with rich datasets and larger network models has created powerful machine learning systems that achieve human-level performance on a diversity of tasks [5, 6]. However, what is most striking from a neuroscience perspective is that DNNs also reproduce essential aspects of the representational geometry of biological neural networks, even when they are not explicitly optimized to do so [7–9]. This similarity suggests that DNNs are valuable tools to elucidate neural information processing in the brain [10, 11]. Yet, two central issues remain concerning their interpretation as models of biological learning. First, the training algorithms used in deep learning are typically end-to-end, i.e., the algorithms optimize network connections across the

hierarchy with the goal of minimizing errors at the network's output. Doing so requires solving the "credit assignment problem," whereby targeted information must be sent to neurons in the network's hidden layers about their respective contributions to the output errors. It remains unclear how neurobiology solves the credit assignment problem [12].

Second, supervised learning requires input data with specific semantic "labels" that are not conceivably available to animals and humans. Here, self-supervised learning (SSL), a family of unsupervised machine learning algorithms, may offer a remedy as they do not need labeled data but instead require that internal network representations belonging to related inputs be predictive of one another [13, 14]. Hence, network representations themselves act as targets for similar inputs. For example, a network has to predict visual features across different parts of an image or predict future representations of sensory inputs from those in the past. It has been proposed that biological neural networks may similarly rely on prediction as a learning principle [15–18], for instance, by extracting slowly varying features from their sensory inputs as done in slow feature analysis (SFA) [19, 20]. In other words, predictive learning posits that biological networks change their connections with the goal of "pulling" together related internal representations, and therefore form similar representations for stimuli that frequently occur close in time (Fig. 1c).

However, a major issue with this strategy is that without any forces opposing this representational pull, such learning inevitably leads to "representational collapse," whereby all inputs are mapped to the same internal activity pattern which precludes linear separability (Fig. 1c). One typical solution to this issue is to add forces that "push" representations corresponding to *different* unrelated stimuli away from one another (Fig. 1d). This is usually done by invoking so-called "negative samples," which are inputs that do not frequently occur together in time. This approach has been linked to biologically plausible three-factor learning rules [21–23], but it requires constant switching of the sign of the plasticity rule depending on whether two successive inputs are related to each other or not. Yet, it is unknown whether and how such a rapid sign switch is implemented in the brain.

Another possible solution for avoiding representational collapse without negative samples is to prevent neuronal activity from becoming constant over time, for instance, by maximizing the variance of the activity [24]. Interestingly, variance maximization is a known signature of Hebbian plasticity [25, 26], which has been found ubiquitously in the brain [27–29]. While Hebbian learning is usually thought of as the primary plasticity mechanism rather than playing a supporting role, Hebbian plasticity alone has had limited success at disentangling representations in deep hierarchical neural network models [10, 30, 31].

In this article, we introduce Latent Predictive Learning (LPL), a conceptual learning framework that overcomes this limitation and reconciles SSL with Hebbian plasticity. Specifically, the learning rules derived within our framework combine a BCM-like plasticity threshold [1, 2] as observed in experiments (Fig. 1e; [27, 32–34]), with a predictive component inspired by SFA [19, 20] that renders single neurons selective to temporally contiguous features in their inputs. When applied to the layers of deep hierarchical networks, LPL yields disentangled representations of objects present in natural images while neither requiring labels nor negative samples. Crucially, LPL effectively disentangles representations despite being a purely local learning rule, i.e., without requiring explicit spatial credit assignment mechanisms. We demonstrate that LPL captures central findings of unsupervised visual learning experiments in monkeys. Finally, the corresponding spiking LPL rule learns predictive representations in spiking neural networks and naturally yields classic spike-timing-dependent plasticity (STDP) windows and a characteristic firing-rate dependence observed in neurobiology. In light of these findings, we argue that LPL constitutes a plausible plasticity mechanism that may underlie representation learning in biological sensory networks.

Results

To study the interplay of Hebbian and predictive plasticity in sensory representation learning, we derived a plasticity model from an SSL objective function that is reminiscent of and extends the classic BCM learning rule [1, 2] (Methods; Supplementary Note S1). According to our learning rule, the temporal dynamics of a synaptic weight W_j are given by

$$\frac{\mathrm{d}W_j}{\mathrm{d}t}(t) = \eta x_j(t) f'(a(t)) \left(\underbrace{-\frac{\mathrm{d}z(t)}{\mathrm{d}t}}_{\text{predictive}} + \underbrace{\frac{\lambda}{\sigma_z(t)^2}\left(z(t) - \bar{z}(t)\right)}_{\text{Hebbian}} \right) \tag{1}$$

where η is a small positive learning rate, $x_j(t)$ denotes the activity of the presynaptic neuron j, $z(t) = f(a(t))$ is the neuronal activity with the activation function f (Fig. 2a), and the net input current $a(t) = \sum_k W_k x_k(t)$. We call the first term in parentheses the predictive term because it promotes learning of slow features [19, 20] by effectively "pulling together" postsynaptic responses to temporally consecutive input stimuli. Importantly, it cancels when the neural activity does not change and, therefore, accurately *predicts* future activity. In the absence of any additional constraints, the predictive term leads to collapsing neuronal activity levels [19]. In our model, collapse is prevented by the Hebbian term in which $\bar{z}(t)$, the running average of the neuronal activity appears, which is reminiscent of BCM-theory [1, 2]. Its strength further depends on an online estimate of the postsynaptic variance of neuronal activity $\sigma_z^2(t)$. This modification posits an additional metaplasticity mechanism controlling the balance between predictive and Hebbian plasticity depending on the postsynaptic neuron's past activity.

To make the link to BCM explicit, we rearrange the terms in Eq. (1) to give:

$$\frac{\mathrm{d}W_j}{\mathrm{d}t}(t) = \eta\lambda \frac{x_j(t) f'(a(t))}{\sigma_z(t)^2} \left(z(t) - \underbrace{\left(\bar{z}(t) + \frac{\sigma_z(t)^2}{\lambda} \frac{\mathrm{d}z(t)}{\mathrm{d}t} \right)}_{\text{Sliding threshold } \Theta(t)} \right) \tag{2}$$

where $\Theta(t)$ corresponds to a time-dependent sliding plasticity threshold (Fig. 2b). While the precise shape of the learning rule depends on the choice of neuronal activation function, its qualitative behavior remains unchanged as long as the function is monotonic (Supplementary Fig. S1). Despite the commonalities, however, there are three essential differences to the BCM model. First, in our model, the threshold depends only linearly on $\bar{z}(t)$ (Fig. 2b), whereas in BCM, the threshold is typically a supralinear function of the moving average $\bar{z}(t)$. Second, the added dependence on the predictive term $-\frac{\mathrm{d}z}{\mathrm{d}t}$ constitutes a separate mechanism that modulates the plasticity threshold depending on the rate-of-change of the postsynaptic activity (Fig. 2c,d). Third, our model adds a variance-dependence that has diverse effects on the sliding threshold when the neuronal output does not accurately predict future activity and, thus, changes rapidly (Fig. 2c,d). We will see that these modifications are crucial to representation learning from the temporal structure in sensory inputs. Because the predictive term encourages neurons to predict future activity at their output, and thus in latent space rather than the input space, we refer to Eq. (1) as the Latent Predictive Learning (LPL) rule.

LPL finds contiguous features in temporal data

To investigate the functional advantages of LPL over BCM and other classic Hebbian learning rules (Supplementary Note S2) in a well-controlled setting, we first designed a synthetic two-dimensional learning task in which we parametrically controlled the proportion of predictable

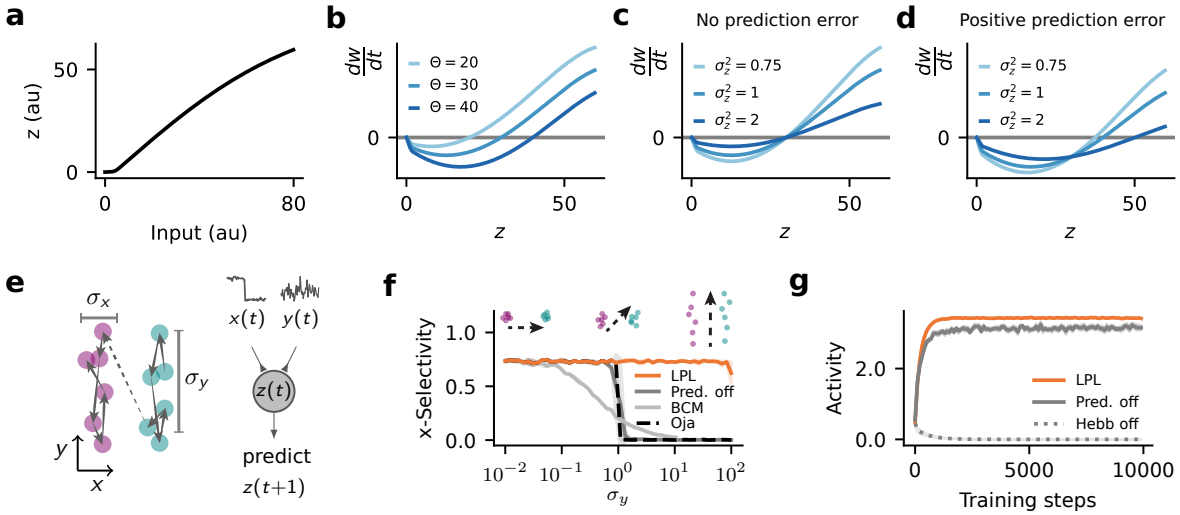

Figure 2: **LPL extends BCM theory by adding a variance- and rate-of-change dependence.** **(a)** Example of a typical neuronal input-output function with postsynaptic activity z. **(b)** Weight change induced by the LPL rule for co-varying input and the postsynaptic activity z for different values of the plasticity threshold Θ, with $\sigma_z^2 = 1$. The functional shift of the threshold is reminiscent of the BCM rule. **(c)** Same as (b) but for different values of the variance of the postsynaptic activity with zero prediction error $\frac{dz}{dt} = 0$ and fixed mean activity $\bar{z} = 30$. **(c)** Same as (c) but with a positive prediction error $\frac{dz}{dt} = +10$. **(e)** Illustration of the two-dimensional synthetic data generating process. Consecutive data points stay within the same cluster separated along the x-direction and are drawn independently from the corresponding normal distribution centered in that cluster (left). These data are fed into a linear neuron that learns via LPL (right). **(f)** Cluster selectivity of the features learned by LPL with and without the predictive term, by the BCM rule and by Oja's rule for different values of σ_y. By varying σ_y, we obtain a family of transition sequences with differing amplitudes of within-cluster transitions (top). LPL selects temporally contiguous features and therefore ensures that the neuron always becomes selective to cluster identity. Oja's rule finds PC1, the direction of highest variance, which switches to the noise direction at $\sigma_y = 1$. BCM and the LPL rule without the predictive component show the same behavior, although in case of BCM, the change was more gradual. Selectivity values were averaged over ten random seeds with the shaded area corresponding to the standard deviation. **(g)** Mean output activity of the neuron over training time for $\sigma_y = 1$ under different versions of LPL. LPL initially increases its response and saturates at some activity level, even when the predictive term is disabled. However, without the Hebbian term, the activity collapses to zero.

changes between subsequent observations (Fig. 2e; Methods). Specifically, the data sequence consisted of noisy inputs distributed over two clusters separated along the x-axis. Consecutive inputs had a high probability of staying within the same cluster, thus making cluster identity a temporally contiguous feature. By varying the noise amplitude σ_y in the y-direction, we effectively controlled the amount of unpredictable changes in the input sequences. We exposed a single rate neuron model to different datasets with varying values of σ_y, while the two input connections were plastic and evolved according to the LPL rule (Eq. (1)) until convergence. We then measured the selectivity of the neuron to cluster identity, defined as the normalized difference between the neuron's average response to inputs from the two clusters (Methods).

We found that LPL rendered the neuron selective to the cluster identity for a large range of σ_y values (Fig. 2f). However, without the predictive term, the neuron's selectivity to cluster identity was lost for large σ_y values. This behaviour was expected because omitting the predictive term renders the learning rule purely Hebbian, which biases selectivity toward directions of highest variance. To illustrate this point, we repeated the same simulated learning experiments with Oja's rule, a classic Hebbian rule that finds the principal component in the input, and found similar qualitative behaviour, except the change was more abrupt at $\sigma_y \approx 1$. Thus LPL's predictive term changes the learning rule's behavior substantially by selecting predictable features in the input instead of directions of high variance.

Next, we sought to confirm that the Hebbian term is essential for LPL to prevent representational collapse. To that end, we simulated learning with $\sigma_y = 1$ under LPL without the Hebbian term (cf. Eq. (1)) and measured the mean output activity of the neuron in response to random inputs in the sequence. We observed that without the Hebbian term, the neuron's activity collapses to zero as expected (Fig. 2g). Conversely, learning with the Hebbian term without the predictive term did not result in collapse. Therefore, LPL's Hebbian component is essential to prevent activity collapse.

Moreover, Hebbian plasticity needs to be dynamically regulated to prevent run-away activity. In LPL this regulation is achieved by inversely scaling the Hebbian term by a moving estimate of the variance of the postsynaptic activity $\sigma_z^2(t)$. Without this variance-modulation, neural activity either collapsed or succumbed to runaway activity depending on whether the predictive term or the Hebbian term was dominant (Supplementary Note S3). Either case precluded the neuron from acquiring cluster selectivity. We verified that these findings generalized to higher-dimensional tasks with more complex co-variance structure (Supplementary Note S4). Hence, the combination of the predictive with variance-modulated Hebbian plasticity in LPL is needed to learn invariant predictive features independently of the co-variance structure in the data.

LPL disentangles representations in deep hierarchical networks

As we view a scene, we often move through it which causes us to see the objects in the scene from different angles. Similarly, the objects, animals, or people in the scene themselves may move. Finally, our gaze constantly shifts due to saccadic eye movement (Fig. 3a; [35]). All of these influences result in different visual projections of the same objects within a scene onto our retina. Therefore, the objects themselves constitute temporally contiguous features in normal vision.

We thus wondered whether training an artificial neural network with LPL on sequences of images in which object identity is preserved results in disentangled object representations at the network's output. To that end, we built a convolutional DNN model in which we "stacked" layers whose synaptic connections evolved according to the LPL rule. Additionally we included a decorrelation term to prevent neurons within a given layer from becoming correlated. In biological neural networks with separate excitatory and inhibitory neuron types this role can be

readily achieved through plasticity at inhibitory synapses [36–39]. The whole learning rule was implemented in a "layer-local" manner, meaning that there were no error signals backpropagated through layers (Methods).

To simulate temporal sequences of related visual inputs, we generated pairs of images by applying different randomized transformations to images sampled from STL-10, a large object-recognition dataset from computer vision (Supplementary Fig. S2; Methods). We exposed our network model to these visual data until learning converged and evaluated the linear decodability of object categories from the learned representations using a separately trained linear classifier.

We found that in networks trained with LPL, object categories could be linearly decoded at the output with an accuracy of $(63.2 \pm 0.3)\%$ (Fig. 3b; Table 1), suggesting that the network has formed partially disentangled representations (Supplementary Fig. S3). To elucidate the contributions of the different learning rule components to disentangling, we conducted several ablation experiments. First, we repeated the same simulation but now excluding the predictive term. This modification resulted in an accuracy of $(27.0 \pm 0.2)\%$, which is *lower* than the linear readout accuracy of a classifier trained directly on the pixels of the input images (see Table 1), indicating that the network did not learn disentangled representations of object identity. This finding is consistent with previous studies that suggested that purely Hebbian plasticity without a predictive component fails to learn disentangled representations in deep networks [10, 30, 31]. We measured a similar drop in accuracy when we disabled either the Hebbian or the decorrelation component during learning (Fig. 3b).

Convolutional DNNs trained through supervised learning use depth to progressively separate representations as activity propagates through their layers [5, 6]. We sought to understand whether networks trained with LPL similarly leverage depth. To answer this question, we measured the linear readout accuracy of the internal representations at every intermediate layer in the network. Crucially, we found that in the LPL-trained networks, the readout accuracy increased with the number of layers until it gradually saturated (Fig. 3c), whereas this was not the case when any of the components of LPL was disabled. Together, these results suggest that each of the three terms of LPL is crucial for learning disentangled representations in hierarchical DNNs.

We made an additional observation worth noting. When learning occurs without the predictive term, linear readout accuracy in the early layers up to Layer 3 was improved before decreasing below the pixel-level baseline accuracy with increasing depth (Fig. 3c). We observed the same effect for the full LPL rule when we exposed the network to temporally inconsistent image sequences by showing random consecutive pairs of images. These observations suggest that learning in early layers may not critically depend on predictive plasticity, but that it plays an increasingly important role in deeper layers. This finding is reminiscent of experimental work in the rat visual cortex showing that temporally inconsistent visual experience in early life leads to impaired complex cell development while simple cells remain largely unaffected [40].

	STL-10		CIFAR-10	
	Layer-local (%)	End-to-end (%)	Layer-local (%)	End-to-end (%)
DNN with LPL	63.2±0.3	72.5±0.1	59.4±0.4	70.4±0.2
Raw pixel values	31.6		35.9	

Table 1: Linear classification accuracy on the STL-10 and CIFAR-10 datasets for LPL and for a linear decoder trained on the raw pixel values (Methods). Error values correspond to SEM over four simulations with different random seeds.

In DNNs exposed to normal visual experience in which object identity changes slowly, the two most common causes for failure to form disentangled representations are representational collapse and dimensional collapse (Supplementary Fig. S4), which results from excessively high

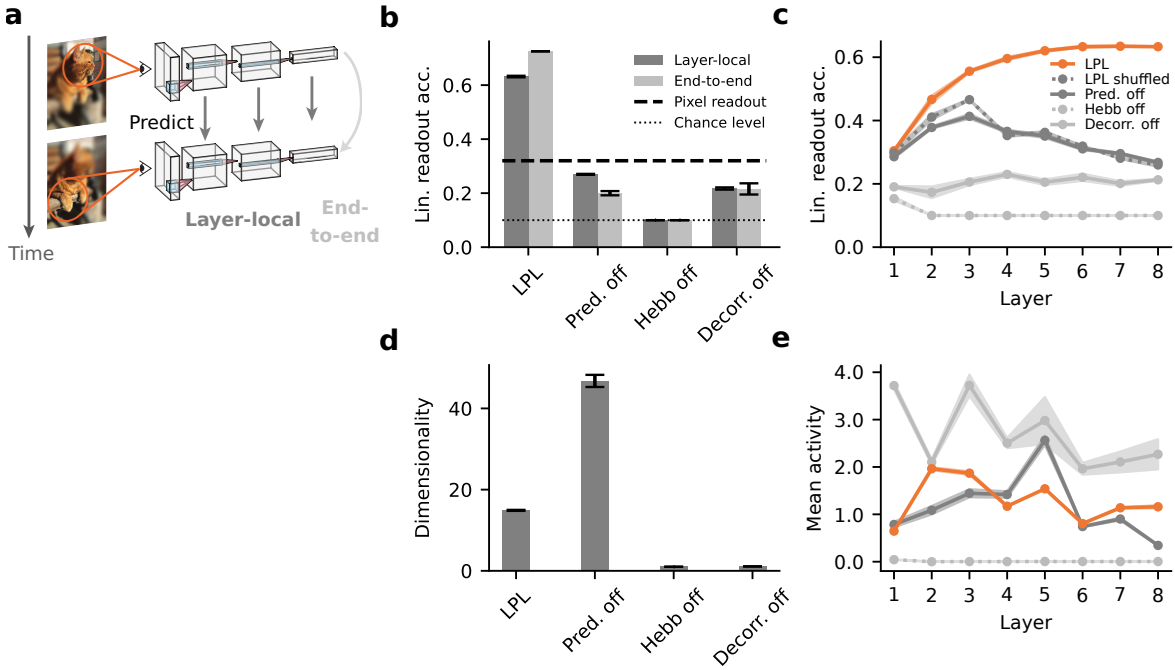

Figure 3: **LPL disentangles representations in DNNs.** **(a)** Schematic of the DNN trained using LPL. We distinguish two learning strategies: Layer-local and end-to-end learning. In layer-local LPL, each layer's learning objective ($\mathcal{L}_i$) is to predict representations within the same layer, whereas end-to-end training takes into account the output layer representations only ($\mathcal{L}_{\text{out}}$), and updates hidden layer weights using backpropagation. **(b)** Linear readout accuracy of object categories decoded from representations at the network output after training it on natural image data (STL-10; see Methods for details) with different learning rules in layer-local (dark) as well as the end-to-end configuration (light). "Pred. off" corresponds to LPL but without the predictive term in the learning rule (cf. Eq. 7). "Hebb off" refers to the configuration without the BCM-like Hebbian term. Finally, "Decorr. off" is the same as the single neuron learning rule (Eq. (1)) without the decorrelation term. LPL yields features with high linear readout accuracy. In contrast, when any component of LPL is disabled, linear readout accuracy drops below the pixel-decoding accuracy of ∼32% (dashed line). Error bars indicate standard error of the mean (SEM) over four trials. **(c)** Linear readout accuracy of the internal representations at different layers of the DNN after layer-local training. LPL's representations improve up to six layers and then settle at a high level. In contrast, readout accuracy is close to chance level without the Hebbian component, and similarly remains at low levels when the decorrelating mechanism is switched off. Interestingly, when the predictive term is off, the readout accuracy initially increases in early layers, but then ultimately decreases back below the pixel-level accuracy with further increasing depth. Finally, the full LPL learning rule applied to inputs in which temporal contingency is destroyed (LPL shuffled) behaves qualitatively similar to the purely Hebbian rule. **(d)** Dimensionality of the internal representations for the different learning rule configurations shown in (b). When either the Hebbian or decorrelation term are disabled, the dimensionality of the representations collapses to one. **(e)** Mean neuronal activity at different layers of the DNN after training with the different learning rule variants shown in (c). Excluding the Hebbian term (dotted line) leads to collapsed representations in all layers.

correlations between neurons [41, 42]. To disambiguate between these two possibilities in our model, we computed the dimensionality of the representations and the mean neuronal activity at every layer (Methods). We found that disabling either the Hebbian or the decorrelation component led to a dimensionality of approximately one, whereas the LPL rule and the learning rule without the predictive term resulted in high-dimensional representations of dimensionality ≈ 15 or ≈ 50 respectively (Fig. 3d). When we disabled the Hebbian term, this resulted in zero activity across all layers (Fig. 3e), which suggests that representational collapse underlies the network's inability to disentangle its input. In contrast, disabling the decorrelation term did not lead to zero activity levels indicating that the reason for poor linear readout accuracy is dimensional collapse (Fig. 3e). Finally, we verified that excluding the predictive component of LPL did not cause collapse either in activity levels or in dimensionality. This suggests that the decreasing linear readout accuracy with depth is due to the network's inability to learn good internal representations. Taken together, these results show that the predictive term is crucial for disentangling object representations in DNNs (Fig. 3) whereas the other terms are essential to prevent different forms of collapse.

In all of the above we assumed layer-local learning, meaning that no learning signals were backpropagated along the hierarchy. Such learning could conceivably be implemented by local learning rules, but it is markedly different from typical DNN training which is end-to-end, namely, training optimizes an objective at the network's output layer. End-to-end training is typically achieved with backpropagation, an algorithm that assigns credit or blame for output errors to neurons in intermediate layers, thereby allowing them to update their weights in a direction that is conducive to reducing the overall error at the network level. It is an ongoing debate whether and how neurobiology achieves credit assignment [10, 12, 43, 44]. In any case, we wanted to know how coordinating learning across layers might influence representation learning with LPL. To answer this question, we repeated our simulations with end-to-end learning using the LPL objective at the network's output (Methods) and found that it reproduced all our key findings of layer-local learning albeit with an improved overall linear readout accuracy (Fig.3b; Table 1). These findings show that LPL disentangles representations irrespective of whether a layer-local objective or end-to-end optimization are used, but its overall performance improves when the underlying learning rules comprise elements of end-to-end optimization. End-to-end optimization in the brain presumably requires dedicated neural circuitry for solving the credit assignment problem, which may be implemented through neuromodulators and dedicated neuronal circuit elements [44–46].

LPL captures invariance learning in the primate inferotemporal cortex

Changing the temporal continuity structure of visual stimuli has been shown to induce neuronal selectivity changes in primate inferotemporal cortex (IT). This effect has been interpreted as a consequence of unsupervised temporal slowness learning (UTL) [16], a principle directly captured by the predictive term in the LPL rule. In Li et al. [16], a macaque freely viewed a blank screen, with objects appearing in the peripheral visual field at one of two alternative locations relative to the (tracked) center of its gaze, prompting the macaque to perform a saccade to this location (Fig. 4a). The experimenters differentiated between normal exposures in which the object does not change during the saccade and "swap exposures" in which the initially presented object was consistently swapped out for a different one as the monkey saccaded to a specific target location X_{swap}. Hence, such swap exposures created an "incorrect" temporal association between one object at position X_{swap} and a different one at the animal's center of gaze X_c. For any particular pair of swap objects, either the location above or below the center of gaze was chosen as X_{swap}, and transitions from the opposite peripheral position $X_{nonswap}$ to

the center X_c were kept consistent as a control. The authors found that the position tolerance of object selectivity of individual neurons in the monkey's IT was systematically altered by swap exposures, an effect they attributed to unsupervised learning. Specifically, a neuron initially selective to an object P over another object N, reduced or even reversed its selectivity at the swap position X_{swap} while preserving its selectivity at the non-swap position $X_{nonswap}$ (Fig. 4b).

We wanted to know whether LPL can account for these observations. To that end, we developed a computational model of the experiment conducted by Li et al. [16] based on the DNN introduced in the previous section. To simulate the prior visual experience that animals had before entering the experiment, we pretrained our network model with LPL on a large natural image dataset (Methods). After pretraining, the learned representations were invariant to where an object was presented on a canvas (Supplementary Fig. S5), a known property of neural representations in the primate IT [4]. Next, we simulated targeted perturbations in the model inputs analogous to the original experimental design. Following Li et al. [16], for a given pair of images from different classes, we switched object identities during transitions from a specific peripheral position, say X_1, to the central position X_c, while keeping transitions from the other peripheral position X_2 to the center unmodified. We used X_1 as the swap position for half of the image pairs, and X_2 for the other half. During exposure to these swapped inputs, we recorded neuronal responses in the network's output layer while the weights in the network model evolved according to the LPL rule.

We observed that the neuronal selectivity between preferred inputs P, as defined by their initial preference (Methods), in comparison to non-preferred stimuli N in the model qualitatively reproduced the evolution of object selectivity reported in the experiments (Fig. 4b). Effectively, LPL trained the network's output neurons to reduce their selectivity to their preferred inputs P at the swap position while preserving their selectivity at the non-swap position. Furthermore, we observed that object selectivity between pairs of control objects not used during the swap training protocol showed no changes, consistent with the experiment (Fig. 4b). Further analysis revealed that the origin of the selectivity changes between P and N stimuli at the swap position was due to both increases in responses to N and decreases in responses to P, an effect also observed in the experiments (Fig. 4c). Thus, LPL can account for neuronal selectivity changes observed in monkey IT during in-vivo unsupervised visual learning experiments.

Spiking neural networks with LPL selectively encode predictive inputs

So far we considered LPL in discrete-time rate-based neuron models without an explicit separation of excitatory and inhibitory neurons. In contrast, cortical circuits employ spiking neurons that obey Dale's law, and learn in continuous time through STDP. Hence, we wanted to test whether our theory would extend to such biologically plausible plastic spiking neural networks (SNNs). To that end, we simulated a recurrent network model consisting of 100 excitatory and 25 inhibitory neurons with sparse connectivity (Fig. 5a; Methods). The circuit received input from five Poisson populations consisting of 100 neurons each, whose firing rates encoded temporally varying signals with different temporal properties (Fig. 5b; Methods). Specifically, input population $P0$ had a constant firing rate, whereas $P1$'s and $P2$'s firing rates followed two independently varying signals. We also defined two control populations $P1_{\text{ctl}}$ and $P2_{\text{ctl}}$ whose firing rates were temporally shuffled versions of $P1$ and $P2$ using bins of 10 ms duration. To avoid confounding effects due to firing rate differences, we ensured that all populations had the same mean firing rate of 5 Hz. The inputs were connected to the excitatory network neurons through plastic connections that evolved according to a spiking generalization of the local LPL rule (cf. (1)) without the decorrelation term. Decorrelation was achieved through plasticity of inhibitory connections onto excitatory neurons. Specifically, we relied on an inhibitory STDP

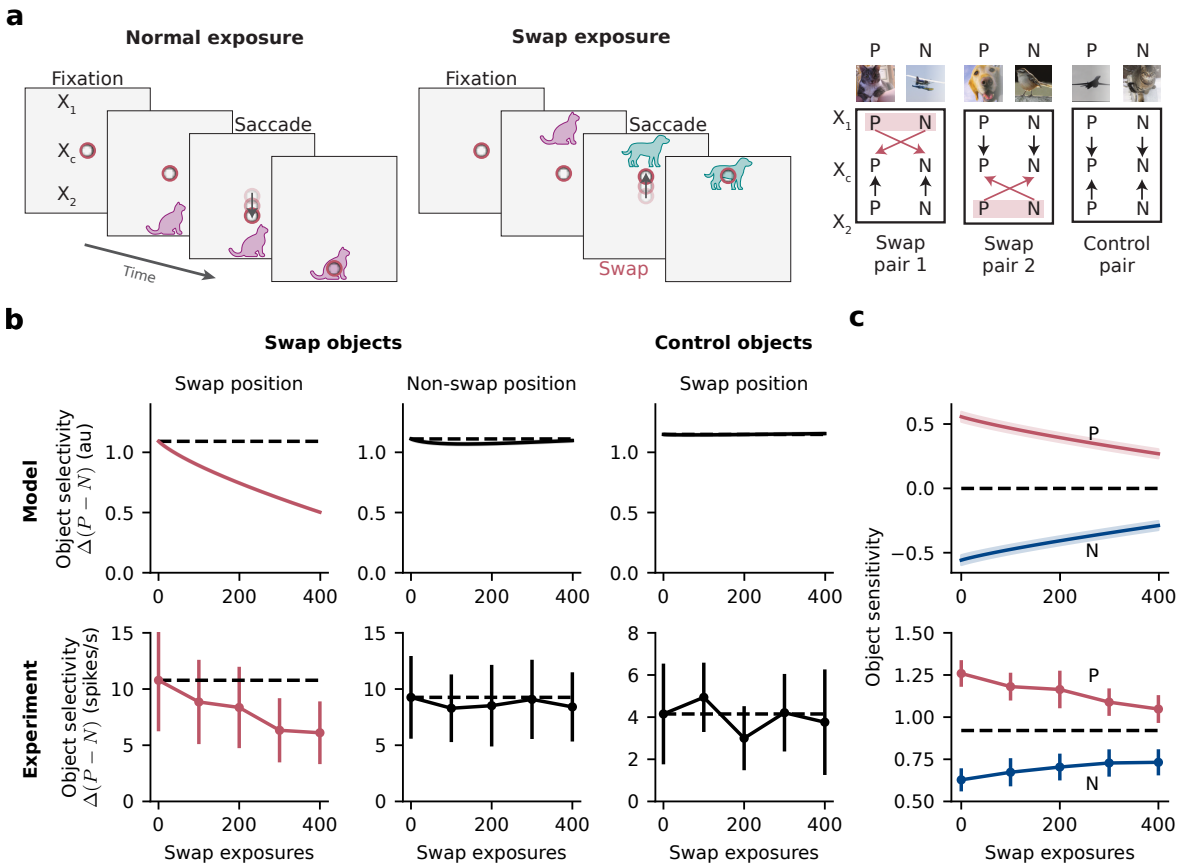

Figure 4: **LPL captures invariance learning in the primate inferotemporal cortex (IT). (a)** Schematic of the simulation setup modeled after Li et al. [16]. The inputs to the model consist of images of objects presented at three different positions X_1, X_c, and X_2 on a blank canvas. Following the original experiment, we performed a targeted perturbation in the simulated visual experience that the model network was exposed to (left and center). Specifically, we switched object identities during transitions from a specific peripheral position, say X_1, to the central position X_c, while keeping transitions from the other peripheral position to the center unmodified (right). **(b)** Evolution of object selectivity as a function of number of swap exposures in the model (top row) and observed in-vivo (bottom row; data points extracted and replotted from [16], see Methods for details). We differentiate between pairs of swapped objects at the Swap (left) and Non-swap positions (center) as well as control objects at the Swap position (right). LPL qualitatively reproduces the evolution of swap position-specific remapping of object selectivity as observed in IT. Control objects at the Swap position, i.e., images not used during the swap training protocol, show no selectivity changes in agreement with the experiment. **(c)** Average response to objects P and N as a function of number of swap exposures. The change in object selectivity between preferred objects P and non-preferred objects N is due to both increased responses to N and decreased responses to P in both our model (top) and the experimental recordings (bottom).

rule that actively decorrelates neuronal activity by promoting excitatory-inhibitory balance ([37]; Methods).

We ran the SNN model for approximately 28 h of simulated time, at which point the network's firing dynamics had settled into an asynchronous irregular activity regime from which the slowly varying input signals could be decoded linearly with high fidelity (Fig. 5b). In contrast, the rate fluctuations of the shuffled control signals ($P1_{\text{ctl}}$ and $P2_{\text{ctl}}$) and the constant firing-rate input ($P0$) could not be reconstructed linearly with high accuracy consistent with the idea that the network preferentially represents the slowly varying inputs in its activity. To directly check whether the neurons had developed preferential selectivity for the slow input signals, we analyzed the afferent connectivity matrix and computed the relative difference between the average afferent weight from each signal in comparison to its associated control pathway. As expected, we found that neuronal weights were preferentially tuned to the slow input channel instead of the associated shuffle control inputs (Fig. 5c,d). However, this selectivity was lost when we turned either the predictive or the Hebbian term off. The absence of Hebbian plasticity was further accompanied by activity collapse (Fig. 5e), consistent with our findings in the rate-based setting.

To investigate the role of inhibition in successful representation learning in the spiking setting, we repeated the above simulation without the inhibitory population. This manipulation resulted in excessively high firing rates (Fig. 5e; Supplementary Fig. S6), a notable reduction of the representational dimensionality (Fig. 5f; Methods), and lower selectivity to the slow signals (Fig. 5d). The reasons for this reduction can be seen in the distribution of weight vectors. In the network with plastic inhibition, weights were more decorrelated and exclusively selective to either $P1$ or $P2$ (Fig. 5g). In contrast, removing inhibition resulted in more correlated weights with few neurons preferentially tuned to one signal or the other (Fig. 5h). Finally, we repeated the same simulation in a network in which an inhibitory population was present but without inhibitory plasticity. This manipulation led to comparable representational dimensionality as for LPL (Fig. 5f), but caused a loss of selectivity relative to the shuffled controls (Fig. 5d). These results indicate that inhibition is needed to prevent correlated neuronal activity and the ensuing reduction in representational dimensionality. Further, inhibitory plasticity is required to ensure that the slow signals are preferentially represented (Supplementary Fig. S6). Together, these findings illustrate that LPL extends to realistic spiking circuits with separate excitatory and inhibitory neuronal populations.

LPL qualitatively reproduces experimentally observed rate and spike-timing dependence of synaptic plasticity

Next, we wanted to examine whether the spike-based LPL rule is consistent with experimental observations of plasticity induction. Experiments commonly report intertwined rate and spike-timing dependence presumably mediated through nonlinear voltage- and calcium-dependent cellular mechanisms [28, 29, 47]. Theoretical work has further established conceptual links between phenomenological STDP models, SFA, and BCM theory [20, 48–52].

To compare LPL to experiments, we simulated a standard STDP induction protocol. Specifically, we paired 100 pre- and post-synaptic action potentials with varying relative timing Δt for a range of different repetition frequencies ρ. During the entire plasticity induction protocol, the postsynaptic cell was kept depolarized close to its firing threshold and weights evolved according to spike-based LPL. We repeated the simulated induction protocol for different initial values of the slowly moving averages of the postsynaptic firing rate $\bar{S}_i(t)$ and variance $\sigma_i^2(t)$ (Methods). This was done because these variables do not change much over the course of a single induction protocol due to their slow dynamics. Their presence, however, makes LPL a

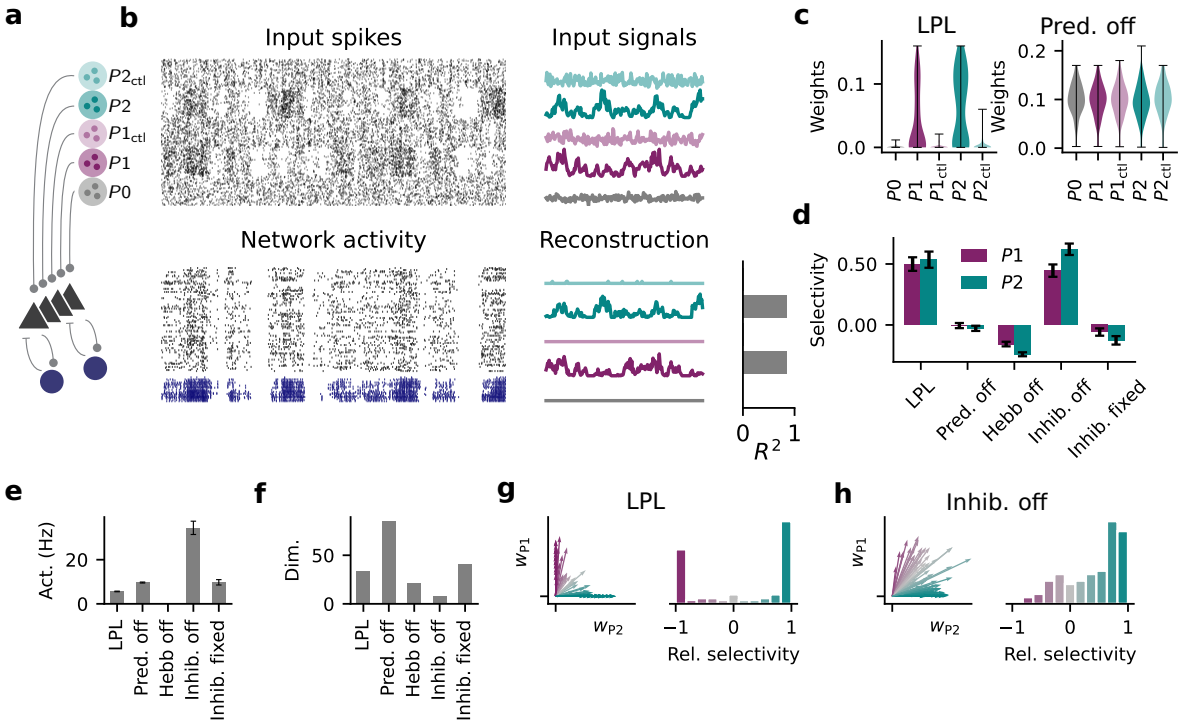

Figure 5: **LPL in a spiking neural network (SNN). (a)** Wiring diagram of the SNN with five distinct input populations. **(b)** Snapshot of spiking activity over 100 ms after LPL plasticity for the inputs (top left) and the network (bottom left) separated into excitatory (black) and inhibitory neurons (blue). The input spikes are organized in five distinct Poisson populations whose firing rates evolve according to five different temporal input signals (top right). Population activity of two slowly varying signals ($P_{1/2}$) can be linearly reconstructed (Methods) with high R^2 values from the network activity whereas temporally shuffled control signals ("ctl"; Methods) are heavily suppressed (bottom right). **(c)** Distribution of synaptic connection strengths grouped by input population. Input connections from slowly varying signals are larger than those from the shuffle controls (left), but not when learning with the predictive term turned off (right). **(d)** Signal selectivity as relative difference between signal and control pathway for networks trained with different learning rule variations (Methods). "LPL" refers to learning with the spiking LPL rule combined with inhibitory plasticity on the inhibitory-to-excitatory connections. "Pred. off" corresponds to learning without the predictive term, and "Hebb off" to learning without the Hebbian term. "Inhib. off" refers to a setting without any inhibitory neurons, whereas "Inhib. fixed" indicates a setting where the inhibitory-to-excitatory weights are held fixed. The network with LPL and inhibitory plasticity acquires high selectivity to both signals. Selectivity is lost if the predictive term, the Hebbian term, or inhibitory plasticity are switched off. When inhibition is removed altogether, selectivity remains but is significantly reduced. Error bars indicate SEM over all excitatory neurons. **(e)** Average firing rate of excitatory neurons in the network for the different configurations in (d). When the Hebbian term is off, spiking activity collapses to low activity levels in contrast to all other configurations in which it settles at intermediate activity levels. **(f)** Dimensionality of the neuronal representations (Methods) for the different configurations in (d). Inhibition prevents dimensionality collapse, even in cases where inhibition is not plastic. **(g)** Averaged weight vectors of all excitatory neurons corresponding to input populations $P1$ and $P2$ (left) and the distribution of relative neuronal selectivities between these populations (right). Most neurons become selective either to $P1$ or $P2$, but few to both signals simultaneously. Color indicates relative preference of their weight vectors to either signal (Methods). **(h)** Same as (g), but without an inhibitory population. Most neurons develop selectivity to $P2$ or mixed selectivity to both signals, and their weight vectors are more correlated.

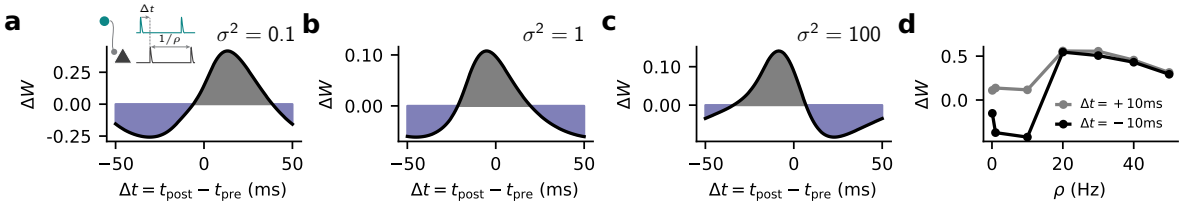

Figure 6: **LPL accounts for STDP and predicts metaplasticity of the STDP window.** **(a)** Relative weight change due to LPL in response to a standard STDP induction protocol with varying spike timing Δt for 100 pairings at a repetition frequency of $\rho = 10\,\text{Hz}$ (inset) for an initial value of $\sigma^2(t = 0) = 0.1$. **(b)** Same as (a), but with initial value of $\sigma^2(0) = 1$. **(c)** Same as (a), but with $\sigma^2(0) = 100$. **(d)** Relative weight change as a function of repetition frequency ρ for positive and negative relative spike timings ($\Delta t = \pm 10\,\text{ms}$).

form of metaplasticity, i.e., the strength of plasticity is induction dependent on past neuronal activity.

We found that for small initial values of σ_i^2, the induced weight changes followed an antisymmetric temporal profile consistent with STDP experiments (Fig. 6a). For larger initial values of σ_i^2, the STDP window changed to a more symmetric and then ultimately an antisymmetric profile while the plasticity amplitude was suppressed as expected due to the variance-dependent suppression of the Hebbian term in the learning rule (Fig. 6b,c). Next we investigated the effect of different initial values for $\bar{S}_i(t)$, which acts as a moving threshold reminiscent of BCM. Specifically, we recorded plastic changes at two fixed spike timing intervals $\Delta t = \pm 10\,\text{ms}$ for $\sigma_i^2(t = 0) = 0.1$. For intermediate threshold values $\bar{S}_i(t = 0) = 20\,\text{Hz}$, causal spike timing induced long-term potentiation (LTP) with a nonlinear frequency dependence (Fig. 6d) whereas acausal pre-after-post timings showed a characteristic cross-over from LTD to LTP similarly observed in experiments [27]. In contrast, a low initial threshold $\bar{S}_i(t = 0) = 0$, which would occur in circuits that have been quiescent for extended periods of time, resulted in LTP induction for both positive and negative spike timings whereas a high initial value ($\bar{S}_i(t = 0) \geq 50\,\text{Hz}$), corresponding to circuits with excessively high activity levels, led to LTD (Supplementary Fig. S7). Importantly such slow shifts in activity-dependent plasticity behavior are consistent with the metaplasticity observed in monocular deprivation experiments [2, 33, 52]. Thus, LPL qualitatively captures key phenomena observed in experiments such as STDP, the rate-dependence of plasticity, and metaplasticity, despite not being optimized to reproduce these phenomena. Rather our model offers a simple normative explanation for the necessity of different plasticity patterns that are also observed experimentally [47].

Discussion

In this article, we have introduced LPL, a local plasticity rule that extends BCM theory by adding a predictive component to Hebbian learning. We demonstrated that LPL disentangles latent object representations in DNNs through mere exposure to temporal data in which object identity is preserved across successive inputs provided neuronal activity is decorrelated. Crucially, we show that both predictive and Hebbian learning have to work in symphony to accomplish this. Moreover, we demonstrated that LPL qualitatively captures the representational changes observed in unsupervised learning experiments in monkey IT [16]. Finally, we extended LPL to spiking neural networks and found that the resulting learning rule naturally reproduces STDP and its rate-dependence as observed in experiments, while further predicting a new form of metaplasticity with distinct variance-dependent changes to the STDP window.

The idea that sensory networks use temporal prediction as a learning objective to form disentangled internal representations has been studied extensively in both machine learning and neuroscience. The model introduced in this article combines and extends aspects of biologically plausible plasticity models closely related to BCM theory with central ideas from SFA and more recent SSL approaches in machine learning. While SSL has shown great promise in representation learning without labelled data, it is typically formulated as a contrastive learning problem requiring negative samples [13, 14] to prevent representional collapse. However, negative samples explicitly break temporal contiguity during learning and are thus not biologically plausible. LPL does not require negative samples. Instead, it relies on variance regularization as proposed in Bardes et al. [24] to prevent collapse. Our model uses virtually the same mechanism while building a conceptual bridge from variance regularization to BCM theory. However, we used a logarithmic variance dependence instead of a piece-wise linear dependence (Supplementary Note S3). LPL's logarithmic dependence yields a smooth derivative which more clearly exposes the relationship to Hebbian metaplasticity.

Like most SSL approaches, Bardes et al. [24] used an end-to-end learning approach whereby the objective function is formulated on the embeddings at the network output. In contrast, we studied the case of greedy layer-wise learning in which the objective is applied to each layer individually. Doing so alleviates the need for backpropagation of errors into hidden layers and, as a result, permitted us to formulate the weight updates as local learning rule. Such a local learning rule formulation is similar to works combining a contrastive objective with greedy layer-wise training [30, 53]. Furthermore, recent work by Illing et al. [21] showed that greedy layer-wise learning is directly linked to local learning rules that rapidly switch between Hebbian and anti-Hebbian learning through a global third factor. However, all of these models required implausible negative samples, whereas LPL does neither require end-to-end training nor negative samples.

LPL shares the shape of the BCM rule, which has been qualitatively confirmed in numerous experimental studies both in-vitro [27, 32, 33] and in-vivo [34]. Furthermore, BCM has been linked to STDP [29] and informed numerous phenomenological plasticity models [48–51, 54]. However, unequivocal evidence for the predicted supralinear behavior of the firing rate-dependence of the BCM sliding threshold remains scarce [33] and the fast sliding threshold required for network stability seems at odds with experiments [52, 55, 56]. In contrast, LPL resembles the shape of the BCM learning rule, but does not require a fast nonlinear sliding threshold for stability. Instead, it posits a fast-acting variance-dependence that ensures stability by suppressing Hebbian plasticity when the variance of the output activity is too high. This suppressive effect allows the sliding threshold, which could be implemented by either neuronal or circuit mechanisms [33, 57, 58], to catch up slowly, more consistent with experiments. LPL, therefore, offers a simple explanation that could help close the current gap between theory and experiment. Validating this theory will require future studies investigating whether and how neuronal circuits regulate plasticity.

The notion of slowness learning has been studied extensively in the context of the Trace Rule [59] and SFA [19, 39] which have conceptual ties to STDP [20]. However, the Trace Rule enforces a hard constraint on the norm of the weight vector to prevent collapse, while SFA explicitly restricts neuronal variance to one. In contrast, LPL merely enforces a soft variance constraint [24] through variance dependence of Hebbian plasticity to the same effect. A similar soft constraint on the variance can be derived from statistical independence arguments [60] within a mutual information view of SSL [13]. However, these studies used negative samples, assume rapid global sign switching of the learning rule, and did not connect their work to biological plasticity mechanisms.

Our study has several limitations which we aim to address in future work. First, our study

is limited to visual tasks of core object recognition, whereas other sensory modalities may use LPL as a mechanism to form disentangled representations of the external world. Moreover, we restricted ourselves to artificial data augmentation techniques borrowed from SSL to generate temporally related images, for instance, by randomly cropping two images from an underlying larger image. We did so for computational efficiency and due to the lack of suitable alternative datasets. It seems clear that this methodology can only serve as a crude proxy for the type of successive stimuli that our visual system experiences as we sample an image through a complex sequence of saccades. Additionally, the image itself may be non-stationary as the state of the world evolves around us. Finally, there remains a performance gap in classification performance compared to less plausible fully supervised and contrastive approaches (Supplementary Table S3) showing that there remains room for improvement, possibly by incorporating biological circuit mechanisms into the model. It is left as future work to show how LPL can be extended to the circuit level and to more ethologically realistic sensory modalities [61] and video input while further combining them with plausible models of saccadic eye movement [62].

We have evaluated our model's ability to disentangle object representations in DNNs and to reproduce neuronal activity signatures of unsupervised visual learning in experiments performed in monkey IT [16]. A logical next question is how well our model matches the representational geometry of other experimental recordings as evaluated previously for contrastive models [7, 63–65]. An appealing metric for such an evaluation is provided by Brain-Score [66]. However, it is typically used for much larger network models trained on ImageNet, a rich 1000-class dataset [7]. In contrast, our networks were smaller and trained on CIFAR-10 and STL-10 with only ten classes. When we evaluated our model on Brain-Score (Supplementary Fig. S8), we found that the models trained end-to-end with either LPL or a supervised objective showed the highest Brain-Score, whereas networks trained with layer-local LPL exhibited a low score. One possible explanation for this finding is that the brain may indeed rely on elements of end-to-end optimization implemented through biologically plausible credit assignment mechanisms at the circuit level. Still, our architecture's Brain-Scores were lower overall than larger network models trained on larger datasets. Due to these architectural differences, interpreting these results is not straightforward. While a more detailed analysis of the origins of these difference goes beyond the scope of the present article, we intend to study this question in more detail in the future.

Despite these limitations, our model makes several concrete predictions about synaptic plasticity. As we have shown, modulating the strength of Hebbian plasticity as a function of the variance of the postsynaptic activity is essential to LPL. A direct prediction of our model is, therefore, that the predictive contribution to plasticity should be best observable when postsynaptic activity is highly variable, while it should be barely observable at low variance levels. While our model does not make quantitative predictions about the time scale on which each neuron would have to estimate its output variance, one would expect that a neuron that has been inactive for a long time, as may be the case in slice experiments, would show stronger Hebbian learning than neurons participating in in-vivo activity. Moreover, LPL should manifest in metaplasticity experiments as a transition from an asymmetric Hebbian STDP window, via a symmetric window to, to ultimately an anti-Hebbian window (cf. Fig. 6) when priming the postsynaptic neuron with increasing output variance. Specifically, we expect a neuron which has remained quiescent for a long period of time to display a classic STDP window, whereas a neuron whose activity has undergone substantial fluctuations in the recent past should show an inverted STDP window. Such metaplasticity may account for the diversity of different shapes of STDP windows observed in experiments [47, 67].

To fathom how established data-driven plasticity models are related to theoretically motivated learning paradigms such as SFA and SSL is essential to understanding the brain. A

central open question in neuroscience remains: How do the different components of such learning rules interact with the rich local microcircuitry to yield useful representations at the network level? In this article, we have only scratched the surface by proposing a local plasticity rule and illustrating its aptitude for disentangling internal representations. However, a performance gap remains compared to learning algorithms that can leverage top-down feedback. We expect that extending predictive learning to the circuit and network level will narrow this gap and generate deep mechanistic insights into the underlying principles of neural plasticity.

Methods

Plasticity model

The LPL rule is derived from an objective function approach. It consists of three distinct parts, each stemming from a different additive term in the following combined objective function:

$$\mathcal{L}_{\text{LPL}} = \mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{Hebb}} + \mathcal{L}_{\text{decorr}} \tag{3}$$

First, the predictive component $\mathcal{L}_{\text{pred}}$ minimizes neuronal output fluctuations for inputs that occur close in time. Second, a Hebbian component $\mathcal{L}_{\text{Hebb}}$ maximizes variance and thereby prevents representational collapse. Finally, $\mathcal{L}_{\text{decorr}}$ is a decorrelation term that we use in all non-spiking network simulations to prevent excessive correlations between neurons within the same layer in a network. In SNNs decorrelation is achieved without this term through lateral inhibition and inhibitory plasticity.

In the following, we consider a network layer with N input units and M output units trained on batches of B pairs of consecutive stimuli. In all simulations we approximate the temporal derivative $\mathrm{d}z/\mathrm{d}t$ which appears in Eqn. (1) by finite differences $z(t)-z(t-\Delta t)$ assuming a discrete timestep Δt while absorbing all constants into the learning rate. In this formulation, the LPL rule has a time horizon of two time steps in the sense that only one temporal transition enters into the learning rule directly. We used this insight to efficiently train our models using minibatches of paired consecutive input stimuli which approximates learning on extended temporal sequences consisting of many time steps. Let $x^b(t) \in \mathbb{R}^N$ be the input to the network at time t, $W \in \mathbb{R}^{M\times N}$ be the weight matrix to be learned, $a^b(t) = Wx^b(t) \in \mathbb{R}^M$ be the pre-activations, and $z_i^b(t) = f(a_i^b(t))$ the activity of the ith output neuron at time t. Finally, b indexes the training example within a minibatch of size B.

Predictive component. We define the predictive objective $\mathcal{L}_{\text{pred}}$ as the mean squared difference between neuronal activity in consecutive time steps.

$$\mathcal{L}_{\text{pred}}(t) = \frac{1}{2MB}\sum_{b=1}^{B}\|z^b(t) - \text{SG}(z^b(t-\Delta t))\|^2 = \frac{1}{2MB}\sum_{b=1}^{B}\sum_{i=1}^{M}\left(z_i^b(t) - \text{SG}(z_i^b(t-\Delta t))\right)^2 \tag{4}$$

Here SG denotes the Stopgrad function, which signifies that the gradient is not evaluated with respect to quantities in the past, thereby removing the need for backpropagation through time.

Hebbian component. To avoid representational collapse we rely on the Hebbian plasticity rule that results from minimizing the negative logarithm of the variance of neuronal activity:

$$\mathcal{L}_{\text{Hebb}}(t) = \frac{1}{M}\sum_{i=1}^{M} -\log\left(\sigma_i^2(t)\right) \tag{5}$$

where $\bar{z}_i(t) = \mathrm{SG}(\frac{1}{B}\sum_{b=1}^{B} z_i^b(t))$ and $\sigma_i^2(t) = \frac{1}{B-1}\sum_{b=1}^{B}\left(z_i^b(t) - \bar{z}_i(t)\right)^2$ are the current estimates of the mean and variance of the activity of the ith output neuron. Note that we do not compute gradients with respect to the mean estimate, which would require backpropagation through time. Assuming that the mean is fixed allows formulating LPL as a temporally local learning rule (cf. Eq. (3)). To minimize the computational burden in DNN simulations, we performed all necessary computations on minibatches, which includes estimating the mean and variance. However, these quantities could also be estimated using stale estimates from previous inputs, a requirement for implementing LPL as an online learning rule. Using stale mean and variance estimates from previous minibatches in our DNN simulations did cause a drop in readout performance (Supplementary Table S1). Still, such a drop could possibly be avoided either using larger mini batch sizes, by further reducing the learning rate, or by computing the estimates as running averages over past inputs. All of the above manipulations result in essentially the same learning rule (see Supplementary Note S1).

Decorrelation component. Finally, we use a decorrelation objective to prevent excessive correlation between different neurons in the same layer as suggested previously [24, 36, 68]. The decorrelation loss function is the sum of the squared off-diagonal terms of the covariance matrix between units within the same layer, which is given as

$$\mathcal{L}_{\mathrm{decorr}}(t) = \frac{1}{(B-1)(M^2-M)}\sum_{b=1}^{B}\sum_{i=1}^{M}\sum_{k\neq i}(z_i^b(t) - \bar{z}_i(t))^2(z_k^b(t) - \bar{z}_k(t))^2 \tag{6}$$

with a scaling factor that keeps the objective invariant to the number of units in the population.

The full learning rule. We obtain the LPL rule as the negative gradient of the total objective $\mathcal{L}_{\mathrm{LPL}}$ plus an added weight decay. For a single network layer, this yields the layer-local LPL rule where we omitted the time argument t from all present quantities for brevity:

$$\begin{aligned}\Delta W_{ij} &= -\eta\left(\frac{\partial\mathcal{L}_{\mathrm{pred}}}{\partial W_{ij}} + \lambda_1\frac{\partial\mathcal{L}_{\mathrm{Hebb}}}{\partial W_{ij}} + \lambda_2\frac{\partial\mathcal{L}_{\mathrm{decorr}}}{\partial W_{ij}}\right) - \eta\eta_w W_{ij} \\ &= \eta\frac{1}{MB}\sum_{b=1}^{B}\left(-(z_i^b - z_i^b(t-\Delta t)) + \lambda_1\frac{\alpha}{\sigma_i^2}(z_i^b - \bar{z}_i) - \lambda_2\beta(z_i^b - \bar{z}_i)\sum_{k\neq i}(z_k^b - \bar{z}_k)^2\right) f'(a_i^b)x_j^b \\ &\quad -\eta\eta_w W_{ij}\end{aligned} \tag{7}$$

Here λ_1 and λ_2 are parameters which control the relative strengths of each objective, α and β are the appropriate normalizing constants for batch size and number of units, and η_w is a parameter controlling the strength of the weight decay.

Numerical optimization methods. We implemented all network models learning with LPL using gradient descent on the equivalent objective function in PyTorch [69] with the Lightning framework [70]. DNN simulations were run on five Linux workstations equipped with Nvidia Quadro RTX 5000 graphic processing units (GPUs) and a compute cluster with Nvidia V100 and A100 GPUs. In case of the DNNs, we used the Adam optimizer [71] to accelerate learning. Parameter values used in all simulations are summarized in Supplementary Table S2.

Learning in the single neuron setup

We considered a simple linear rate-based neuron model whose output firing rate z is given by the weighted sum of the firing rates x_j of the input neurons, i.e, $z = \sum_j W_j x_j$, where W_j

corresponds to the synaptic weight of input j. We trained the neuron using stochastic gradient descent (SGD) on the corresponding objective function:

$$\mathcal{L} = \frac{1}{B}\left(z(t) - SG(z(t-\Delta t))\right)^2 - \log\left(\sigma_z^2(t) + \epsilon\right) - \eta_w \sum_j W_j^2 \tag{8}$$

Here, and in all following simulations, we fixed the Hebbian coefficient $\lambda_1 = 1$. We also added a small constant $\epsilon = 10^{-6}$ to the estimate of the variance σ_z for numerical stability. In case of a single rate neuron, the LPL rule (Eq. (7)) simplifies to Eq. (1) without the decorrelation term.

Synthetic two-dimensional dataset generation. The two-dimensional synthetic data sequence (Fig. 2e) consists of two clusters of inputs, one centered at $x = -1$, and the other at $x = +1$. Pairs of consecutive data points were drawn independently from normal distributions centered at their corresponding cluster. To generate a family of different datasets, we kept the standard deviation in the x-direction fixed at $\sigma_x = 0.1$ and varied σ_y. Additionally, to account for occasional transitions between clusters with probability p, we included a corresponding fraction of such "crossover pairs" in the training batch. For each value of σ_y, we simulated the evolution of the input connections of a single linear model neuron that received the x and y as its two inputs, and updated its input weights according to LPL. In the simulations in Fig. 2 we assumed $p \to 0$, however, the qualitative behavior remained unchanged for noise levels below $p = 0.5$, i.e, as long as the "noisy" pairs of points from different clusters were rare in each training batch (Supplementary Fig. S9).

Neuronal selectivity measure. After training weights to convergence, we measured the neuron's selectivity to the x-input as the normalized difference between mean responses to stimuli coming from the two respective input clusters. Concretely, let $\langle z_1 \rangle$ be the average output caused by inputs from the $x = 1$ cluster, and $\langle z_2 \rangle$ from the $x = -1$ cluster, then the selectivity χ is defined as:

$$\chi = \frac{|\langle z_1 \rangle - \langle z_2 \rangle|}{z_{\max} - z_{\min}} \tag{9}$$

with $z_{\max}$ the maximum and $z_{\min}$ the minimum response across all inputs.

Learning in deep convolutional neural networks

For all network simulations, we used a convolutional DNN based on the VGG-11 architecture [72] (see Supplementary Note S5 for details). We trained this network on STL-10 [73] and CIFAR-10 [74] (Supplementary Fig. S10), two natural image datasets (see Supplementary Table S2 for hyperparameters). To simulate related consecutive inputs, we used two differently augmented versions of the same underlying image, a typical approach in vision-based SSL methods. Specifically, we first standardized the pixel values to zero mean and unit standard deviation within each dataset before using the set of augmentations originally suggested in [14], which includes random crops, blurring, color jitter and random horizontal flips (see Supplementary Fig. S2 for examples).

Network training. We trained our network models on natural image data by minimizing the equivalent LPL objective function. For both datasets, we trained the DNN using the Adam optimizer with default parameters [71] and a cosine learning rate schedule that drove the learning rate to zero after 800 epochs. We distinguish between two cases: layer-local and end-to-end learning. End-to-end learning corresponds to training the network by optimizing $\mathcal{L}_{\text{LPL}}^{(\text{out})}$ at the

network's output while using backpropagation to train the hidden layer weights. This is the standard approach used in deep learning. In contrast, in layer-local learning, we minimized the LPL objective $\mathcal{L}_{\text{LPL}}$ at each layer in the network independently without backpropagating loss gradients between layers similar to previous work [21, 30]. In this case, every layer greedily learns predictive features of its own inputs, i.e, its previous layer's representations. To achieve this behavior, we prevented PyTorch from backpropagating gradients between layers by detaching the output of every layer in the forward pass and optimizing the sum of per-layer losses $\sum_l \mathcal{L}^{(l)}_{\text{LPL}}$.

Unless mentioned otherwise, we used global average pooling (GAP) to reduce feature maps to a single vector before applying the learning objective at the output of every convolutional layer for layer-local training, or just at the final output in the case of end-to-end training. Although pooling was not strictly necessary and LPL could be directly applied on the feature maps, it substantially sped up learning and led to an overall improved linear readout accuracy on CIFAR-10 (Supplementary Table S1). However, we observed that GAP was essential on the STL-10 dataset for achieving readout accuracy levels above the pixel-level baseline (cf. Table 1). This discrepancy was presumably due to the larger pixel dimensions of this dataset and the resulting smaller relative receptive field size in early convolutional layers. Concretely, feature pixels in the first convolutional layer of VGG-11 have a receptive field of 3×3 pixels covering a larger portion of the 32×32 CIFAR-10 images as compared to the 96×96 STL-10 inputs. This hypothesis was corroborated by the fact that when we sub-sampled STL-10 images to a 32×32 resolution, the dependence on GAP was removed and LPL was effective directly on the feature maps (Supplementary Table S1).

Baseline models. As baseline models for comparison (Supplementary Table S3), we trained the same convolutional neural network (CNN) network architecture either with a standard cross-entropy supervised objective, which requires labels, or with a contrastive objective, which relies on negative samples. To implement contrastive learning, the network outputs $z(t)$ were passed through two additional dense projection layers $v(t) = f_{\text{proj}}(z(t))$, which is considered crucial in contrastive learning to avoid dimensional collapse [41]. Finally, the following contrastive loss function was applied to these projected outputs

$$\mathcal{L}_{\text{contrast}}(t) = \sum_{b=1}^{B} \left(-\text{sim}(v^b(t), \text{SG}(v^b(t - \Delta t))) + \sum_{b' \neq b}^{B} \text{sim}(v^b(t), v^{b'}(t)) \right) \tag{10}$$

where $\text{sim}(v_1, v_2) = \frac{v_1^{\text{T}} v_2}{\|v_1\| \|v_2\|}$ is the cosine similarity between two representations v_1 and v_2. The second term in the loss function is a sum over all pairwise similarities between inputs in a given minibatch. These pairs correspond to different underlying base images and therefore constitute negative samples. During training the network is therefore optimized to reduce the representational similarity between them.

For training the layer-local versions of the supervised and contrastive models, we followed the same procedure as with LPL of optimizing the respective loss function at the output of every convolutional layer l of the DNN without backpropagation between the layers. Because projection networks are necessary for avoiding dimensional collapse in case of contrastive learning, we included two additional dense layers to obtain the projected representations $v^l(t) = f^l_{\text{proj}}(z^l(t))$ at every level of the DNN before calculating the layer-wise contrastive loss $\mathcal{L}^l_{\text{contrast}}$. This meant that gradients were backpropagated through each of these dense layers for training the corresponding convolutional layers of the DNN, but consecutive convolutional layers were still trained independent of each other.

Analysis of population activity and representational dimension

We adopted two different metrics in order to analyze the representations learned by the DNN after unsupervised training with LPL on the natural image datasets.

Linear readout accuracy. To evaluate how well the LPL rule trained the DNN to disentangle and identify underlying latent factors in a given image, we measured linear decodability by training a linear classifier on the network outputs in response to a set of training images. Crucially, during this step we only trained the readout weights while keeping the weights of the LPL-pretrained DNN frozen. We then evaluated the linear readout accuracy (Fig. 3b) on a held-out test set of images. We used the same procedure to evaluate the representations at intermediate layers (Fig. 3c), and for the baseline models.

Dimensionality and activity measures. To characterize mean activity levels in the network models, we averaged neuronal responses over all inputs in the validation set. To quantify the dimensionality of the learned representations, we computed the participation ratio [75]. Concretely, if $Z \in \mathcal{R}^{B \times N}$ are N-dimensional representations of B input images, and λ_i, $1 \leq i \leq N$ is the set of eigenvalues of $Z^T Z$, then the participation ratio is given by:

$$\text{Dim.} = \frac{\left(\sum_{i=1}^{N} \lambda_i\right)^2}{\sum_{i=1}^{N} \lambda_i^2} \tag{11}$$

Brain-Score. We computed Brain-Scores [66] for instances of our model by submitting it to `http://www.brain-score.org`. Specifically, we obtained the values for the untrained network and after training the model on STL-10 using both LPL and supervised training. We computed these scores for both end-to-end and layer-wise training objectives. Because our networks were trained on 96×96 images, we downsampled the Brain-Score inputs to our network's native resolution using the preprocessing function provided in the submission pipeline.

Model of unsupervised learning in inferotemporal cortex

Network model and pretraining dataset. To simulate the experimental setup of [16], we modeled the animal's ventral visual pathway with a convolutional DNN. To that end, we used the same network architecture as before except that we removed all biases in the convolutional layers in order to prevent boundary effects. This modification resulted in a drop in linear readout accuracy (Supplementary Table S1). Pre-training of the network model proceeded in two steps as follows. First, we performed unsupervised pre-training for 800 epochs on STL-10 using augmented image views exactly as before. Next, we added a fully-connected dense layer at the network's output, and trained it for 10 epochs with the LPL objective while keeping the weights of the convolutional layers frozen. During this second pre-training phase, we used augmented STL-10 inputs which were spatially extended in order to account for the added spatial dimension of different canvas positions in the experiment by Li et al. [16]. The expanded inputs consisted of images placed on a large black canvas at either the center position X_c or one of two peripheral positions $\mathrm{X}_{1/2}$ at the upper or lower end of the canvas. Concretely, these images had dimensions $(13 \times 96) \times 96$ which resulted in an expanded feature map at the output of the convolutional DNN with spatial dimensions 13×1 (see Supplementary Note S5 for details). Note that we only expanded the canvas in the vertical dimension instead of using a setup with a 13×13 feature map because it resulted in a substantial reduction of computational and memory complexity. During this second stage of pre-training, the network was only exposed to "true"

temporal transitions wherein the image was not altered between time steps apart from changing position on the canvas.

Data generation for simulated swap exposures. To simulate the experiment by [16], we exposed the network to normal and swap temporal transitions. In the latter case the image was consistently switched to one belonging to a different object category at the specific swap position. The swap position for a given pair of images was randomly pre-selected to be either X_1 or X_2, while the other non-swap position was used as a control. Specifically, we switched object identities during transitions from one peripheral swap position, say X_1, to the central position X_c, while keeping transitions from the other peripheral position X_2 to the center unmodified. As in the experiment, we chose several pairs of images as swap pairs, and fixed X_1 as the swap position for half the pairs of images and X_2 as the swap position for the other half. To simulate ongoing learning during exposure to these swap and non-swap input sequences, we continued fine-tuning the convolutional layers. To that end, we used the Adam optimizer we used during pre-training with its internal state restored to the state at the end of pre-training. Moreover, we used a learning rate of 10^{-7} during fine-tuning which was approximately $100\times$ larger than the learning rate reached by the cosine learning rate schedule during pre-training (4×10^{-9}, after 800 epochs). Finally, we trained the newly added dense layers with vanilla SGD with a learning rate of 0.02.

Neuronal selectivity analysis. Before training on the swap exposures, for each output neuron in the dense layer, we identified the preferred and non-preferred member of each swap image pair, based on which image drove higher activity in that neuron. This allowed us to quantify object selectivity on a per-neuron basis as $P - N$, where P is the neuron's response to its initially preferred image, and N to its nonpreferred image at the same position on the canvas. Note that, by definition, the initial object selectivity for every neuron is positive. Finally, we measured the changes in object selectivity $P - N$ during the swap training regimen, at the swap and non-swap positions averaging over all output neurons for all image pairs. As a control, we included measurements of the selectivity between pairs of control images that were not part of the swap set.

Comparison to experimental data. To compare our model to experiments, we extracted the data from Li et al. [16] using the Engauge Digitizer software [76] and replotted it in Fig. 4b.

Spiking neural network simulations

We tested a spiking version of LPL in networks of conductance-based leaky integrate-and-fire (LIF) neurons. Specifically, we simulated a recurrent network of 125 spiking neurons (100 excitatory and 25 inhibitory neurons) receiving afferent connections from 500 input neurons. In all simulations the input connections evolved according to the spike-based LPL rule described below. In our model, neurons actively decorrelated each other through locally connected inhibitory interneurons whose connectivity was shaped by inhibitory plasticity.

Neuron model. The neuron model was based on previous work [26, 55] in which the membrane potential U_i of neuron i evolves according to the ordinary differential equation

$$\tau^{\text{mem}} \frac{dU_i}{dt} = \left(U^{\text{leak}} - U_i\right) + g_i^{\text{exc}}(t)\left(U^{\text{exc}} - U_i\right) + g_i^{\text{inh}}(t)\left(U^{\text{inh}} - U_i\right) \tag{12}$$

where τ^{mem} denotes the membrane time constant, U^x are the synaptic reversal potentials (Supplementary Table S4), and the $g_i^x(t)$ the corresponding synaptic conductances expressed in units

of the neuronal leak conductance. The excitatory conductance is the sum of NMDA and AMPA conductances $g_i^{\text{exc}}(t) = 0.5(g_i^{\text{ampa}}(t) + g_i^{\text{nmda}}(t))$. Their dynamics are described by the following differential equations

$$\frac{dg_i^{\text{ampa}}}{dt}(t) = -\frac{g_i^{\text{exc}}(t)}{\tau^{\text{ampa}}} + \sum_{j \in \text{exc}} w_{ij} S_j(t) \tag{13}$$

$$\tau^{\text{nmda}} \frac{dg_i^{\text{nmda}}}{dt}(t) = g_i^{\text{ampa}}(t) - g_i^{\text{nmda}}(t) \tag{14}$$

whereas the inhibitory GABA conductance $g_i^{\text{inh}} = g_i^{\text{gaba}}$ evolves as

$$\tau^{\text{gaba}} \frac{dg_i^{\text{gaba}}}{dt} = -g_i^{\text{gaba}} + \sum_{j \in \text{inh}} w_{ij} S_j(t) \quad . \tag{15}$$

In the above expressions $S_j(t) = \sum_k \delta(t_j^k - t)$ refers to the afferent spike train emitted by neuron j, in which t_j^k are the corresponding firing times, and τ^x denotes the individual neuronal and synaptic time constants (Supplementary Table S4). Neuron i fires an output spike whenever its membrane potential reaches the dynamic firing threshold $\vartheta_i(t)$ that evolves according to

$$\frac{d\vartheta_i}{dt}(t) = \frac{\vartheta^{\text{rest}} - \vartheta_i(t)}{\tau^{\text{thr}}} + \Delta_\vartheta S_i(t) \tag{16}$$

to implement an absolute and relative refractory period. Specifically, ϑ_i jumps by $\Delta_\vartheta = 100\,\text{mV}$ every time an output spike is triggered after which it exponentially decays back to its rest value of $\vartheta^{\text{rest}} = -50\,\text{mV}$. All neuronal spikes are delayed by 0.8 ms to simulate axonal delay and to allow efficient parallel simulation before they trigger postsynaptic potential in other neurons.

Time varying spiking input model. Inputs were generated from 500 input neurons divided into five populations of 100 Poisson neurons each. All inputs where implemented as independent Poisson processes with the same average firing rate of 5 Hz and neurons within the same group shared the same instantaneous firing rate. Concretely, neurons in $P0$ had a fixed firing rate of 5 Hz, whereas the firing rates in groups $P1$ and $P2$ changed slowly over time. Specifically, we generated periodic template signals $x(t)$ from a Fourier basis

$$x(t) = \sum_k \frac{\theta_k}{\alpha^k} \sin\left(\frac{2\pi t + \phi_k}{T}\right) \tag{17}$$

with random uniformly drawn coefficients $0 \leq \theta_k, \phi_k < 1$. The spectral decay constant $\alpha = 1.1$ biased the signals toward slow frequencies and thus slowly varying temporal structure. We chose the period $T = 3\,\text{s}$ for $P1$ and $(3+1/13)\text{s}$ for $P2$ respectively. The different periods were chosen to avoid phase-locking between the two signals. Both signals were then sampled at 10 ms intervals, centered on 5 Hz, variance-normalized, and clipped below at 0.1 Hz before using them as periodic time varying firing rates for $P1$ and $P2$. Additionally, we simulated control inputs $P1/2_{\text{ctl}}$ of the two input signals by destroying their slowly varying temporal structure. To that end, we repeated the original firing rate profile for 13 periods before shuffling it on a time grid with 10 ms temporal resolution.

Spike-based LPL. To extend LPL to the spiking domain, we build on SuperSpike [77], a previously published online learning rule, which had only been used in the context of supervised learning in SNNs thus far. In this article, we replaced the supervised loss with the LPL loss

(Eq. (3)) without the decorrelation term. The resulting spiking LPL online rule for the weight w_{ij} is given by

$$\frac{\mathrm{d}w_{ij}}{\mathrm{d}t} = \eta\ \alpha * \left(\underbrace{\epsilon * S_j(t)}_{\text{pre}} \underbrace{f'(U_i(t))}_{\text{post}} \right) \left[\alpha * \left(\underbrace{-\left(S_i(t) - S_i(t-\Delta t)\right))}_{\text{predictive}} + \underbrace{\frac{\lambda}{\sigma_i^2 + \xi}\left(S_i(t) - \bar{S}_i(t)\right)}_{\text{Hebb}} \right) \right] + \eta \underbrace{\delta S_j(t)}_{\text{transmitter-triggered}} \tag{18}$$

with the learning rate $\eta = 10^{-2}$, a small positive constant $\xi = 10^{-3}$ to avoid division by zero. Further, α is a double exponential causal filter kernel applied to the neuronal spike train $S_i(t)$. Similarly, ϵ is a causal filter kernel that captures the temporal shape of how a presynaptic spike influences the postsynaptic membrane potential. For simplicity, we assumed a fixed kernel and ignored any conductance-based effects and NMDA dependence. Further, we added the transmitter-triggered plasticity term with $\delta = 10^{-5}$ to ensure that weights of quiescent neurons slowly potentiate in the absence of activity to ultimately render them active [78]. Finally, $\lambda = 1$ is a constant that modulates the strength of the Hebbian term. We set it to zero to switch off the predictive term where this is mentioned explicitly.

Further, $f'(U_i) = \beta\left(1 + \beta\left|U_i - \vartheta^{\text{rest}}\right|\right)^{-2}$ is the surrogate derivative with $\beta = 1\,\mathrm{mV}^{-1}$, which renders the learning rule voltage-dependent. Finally, $\bar{S}_i(t)$ and $\sigma_i^2(t)$ are slowly varying quantities obtained online as exponential moving averages with the following dynamics

$$\tau^{\text{mean}} \frac{\mathrm{d}\bar{S}_i(t)}{\mathrm{d}t} = S_i(t) - \bar{S}_i(t) \tag{19}$$

$$\tau^{\text{var}} \frac{\mathrm{d}}{\mathrm{d}t}\sigma_i^2(t) = -\sigma_i^2(t) + \left(S_i(t) - \bar{S}_i(t)\right)^2 \tag{20}$$

with $\tau^{\text{mean}} = 600\,\mathrm{s}$ and $\tau^{\text{var}} = 20\,\mathrm{s}$. These quantities confer the spiking LPL rule with elements of metaplasticity [33].

In our simulations, we computed the convolutions with α and ϵ by double exponential filtering of all quantities. Generally, for the time varying quantity $c(t)$ we computed

$$\tau^{\text{rise}} \frac{\mathrm{d}\bar{c}}{dt}(t) = -\bar{c}(t) + c(t) \tag{21}$$

$$\tau^{\text{fall}} \frac{\mathrm{d}\bar{\bar{c}}}{dt}(t) = -\bar{\bar{c}}(t) + \bar{c}(t) \tag{22}$$

which yields the convolved quantity $\bar{\bar{c}}$. Specifically, we used $\tau_\alpha^{\text{rise}} = 2\,\mathrm{ms}$, $\tau_\alpha^{\text{fall}} = 10\,\mathrm{ms}$, $\tau_\epsilon^{\text{rise}} = \tau_{\text{ampa}} = 5\,\mathrm{ms}$, and $\tau_\epsilon^{\text{fall}} = \tau_{\text{mem}} = 20\,\mathrm{ms}$.

Overall, one can appreciate the resemblance of Eq. (18) with the non-spiking equivalent (cf. Eq. (1)). As in the non-spiking case the learning rule is local in that it only depends on pre- and postsynaptic quantities. The predictive term in the learning rule can be seen as an instantaneous error signal which is minimized when the present output spike train $S_i(t)$ is identical to a delayed version of the same spike train $S_i(t - \Delta t)$ with $\Delta t = 20\,\mathrm{ms}$. In other words, the past output serves as a target spike train (cf. [77]).

Microcircuit connectivity. Connections from the input population to the network neurons and recurrent connections were initialized with unstructured random sparse connectivity with different initial weight values (Supplementary Table S5). One exception to this rule was the

excitatory-to-inhibitory connectivity which was set up with a Gaussian connection probability profile

$$P_{ij}^{\text{con}} = \exp\left(-\frac{(j - c(i))^2}{\sigma^2}\right) \tag{23}$$

with $c(i) = 0.25i$ with $\sigma^2 = 20$ to mimic the dense local connectivity onto inhibitory neurons due to which inhibitory neurons inherit some of the tuning of their surrounding excitatory cells.

Inhibitory plasticity. Inhibitory to excitatory synapses were plastic unless mentioned otherwise. We modeled inhibitory plasticity according to a previously published inhibitory STDP model [37].

$$\frac{dw_{ij}^{\text{inh}}}{dt} = \zeta\left((x_i(t) + 2\kappa\tau^{\text{stdp}})S_j(t) + (x_j(t)S_i(t))\right) \tag{24}$$

using pre- and postsynaptic traces

$$\frac{dx_k}{dt} = -\frac{x_j(t)}{\tau^{\text{STDP}}} + S_k(t) \tag{25}$$

with time constant $\tau^{\text{STDP}} = 20\,\text{ms}$, learning rate $\zeta = 1 \times 10^{-3}$, and target firing rate $\kappa = 10\,\text{Hz}$.

Reconstruction of input signals from network activity. To reconstruct the input signals, we first computed input firing rates of the five input populations by binning their spikes emitted during the last 100 s of the simulation in 25 ms bins. We further averaged the binned spikes over input neurons to provide the regression targets. Similarly, we computed the binned firing rates of the network neurons but without averaging over neurons. We then performed Lasso regression using SciKit-learn with default parameters to predict each target input signal from the network firing rates. Specifically, we trained on the first 95 s of the activity data, and computed R^2 scores on the Lasso predictions over the last 5 s of held-out data (Fig. 5b).

Signal selectivity measures. We measured signal selectivity of each neuron to the two slow signals relative to their associated shuffled controls (Fig. 5d) using the following relative measure defined on the weights:

$$\chi^i = \frac{w_P^i - w_{P_{\text{ctl}}}^i}{w_P^i + w_{P_{\text{ctl}}}^i} \tag{26}$$

where w_P^i is the average synaptic connection strength from the signal pathways $P1/2$ onto excitatory neuron i, and $w_{P_{\text{ctl}}}^i$ is the same but from the control pathways $P1/2_{\text{ctl}}$.

Representational dimension. To quantify the dimensionality of the learned neuronal representations (Fig. 5f), we binned network spikes in 25 ms bins and computed the participation ratio (Eq. (11)) of the binned data.

Neuronal tuning analysis of the learned weight profiles. To characterise the receptive fields of each neuron (Fig. 5g,h), we plotted w_{P1} against w_{P2} for every neuron in the excitatory population (Figs. 5g,h; left), and colored the resulting weight vectors by mapping the cosine of the vectors with the x-axis (w_{P2}) to a diverging color map. Furthermore, we calculated the relative tuning index as follows

$$\chi_{\text{rel}}^i = \frac{w_{P2}^i - w_{P1}^i}{w_{P2}^i + w_{P1}^i} \quad . \tag{27}$$

STDP induction protocols. To measure STDP curves, we simulated a single neuron using the spiking LPL rule (Eq. 18) with a learning rate of $\eta = 5 \times 10^{-3}$. In all cases, we measured plasticity outcomes from 100 pairings of pre- and postsynaptic spikes at varying repetition frequencies ρ. The postsynaptic neuron's membrane voltage was held fixed between spikes at -51mV for the entire duration of the protocol. To measure STDP curves, we set the initial synaptic weight at 0.5 and simulated 100 different pre-post time delays Δt chosen uniformly from the interval $[-50, 50]$ ms with $\rho = 10\,\text{Hz}$. To measure the rate-dependence of plasticity, we repeated the simulations for fixed $\Delta t = \pm 10\,\text{ms}$ while varying the repetition frequency ρ.

Numerical simulations. All SNN simulations were implemented in C++ using the Auryn SNN simulator[1] [79]. Throughout we used a 0.1 ms simulation time step. Simulations were run on seven Dell Precision workstations with eight-core Intel Xeon CPUs.

References

[1] Bienenstock, E. L., Cooper, L. N., and Munroe, P. W. "Theory of the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex". In: *J. Neurosci.* 2 (1982), pp. 32–48.

[2] Cooper, L. N. and Bear, M. F. "The BCM theory of synapse modification at 30: interaction of theory with experiment". en. In: *Nature Reviews Neuroscience* 13.11 (Jan. 2012), pp. 798–810. ISSN: 1471-003X. DOI: 10.1038/nrn3353.

[3] DiCarlo, J. J., Zoccolan, D., and Rust, N. C. "How Does the Brain Solve Visual Object Recognition?" en. In: *Neuron* 73.3 (Feb. 2012), pp. 415–434. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2012.01.010.

[4] DiCarlo, J. J. and Cox, D. D. "Untangling invariant object recognition". In: *Trends in Cognitive Sciences* 11.8 (Aug. 2007), pp. 333–341. ISSN: 1364-6613. DOI: 10.1016/j.tics.2007.06.010.

[5] LeCun, Y., Bengio, Y., and Hinton, G. "Deep learning". en. In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 0028-0836. DOI: 10.1038/nature14539.

[6] Schmidhuber, J. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003.

[7] Yamins, D. L. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., and DiCarlo, J. J. "Performance-optimized hierarchical models predict neural responses in higher visual cortex". en. In: *Proceedings of the National Academy of Sciences* 111.23 (Oct. 2014), pp. 8619–8624. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1403112111.

[8] Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., Pritzel, A., Chadwick, M. J., Degris, T., Modayil, J., Wayne, G., Soyer, H., Viola, F., Zhang, B., Goroshin, R., Rabinowitz, N., Pascanu, R., Beattie, C., Petersen, S., Sadik, A., Gaffney, S., King, H., Kavukcuoglu, K., Hassabis, D., Hadsell, R., and Kumaran, D. "Vector-based navigation using grid-like representations in artificial agents". en. In: *Nature* 557.7705 (May 2018), pp. 429–433. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0102-6.

[9] Nayebi, A., Attinger, A., Campbell, M., Hardcastle, K., Low, I., Mallory, C. S., Mel, G., Sorscher, B., Williams, A. H., Ganguli, S., et al. "Explaining heterogeneity in medial entorhinal cortex with task-driven neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12167–12179.

[1] https://github.com/fzenke/auryn

[10] Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R. P., Berker, A. de, Ganguli, S., Gillon, C. J., Hafner, D., Kepecs, A., Kriegeskorte, N., Latham, P., Lindsay, G. W., Miller, K. D., Naud, R., Pack, C. C., Poirazi, P., Roelfsema, P., Sacramento, J., Saxe, A., Scellier, B., Schapiro, A. C., Senn, W., Wayne, G., Yamins, D., Zenke, F., Zylberberg, J., Therien, D., and Kording, K. P. "A deep learning framework for neuroscience". In: *Nature Neuroscience* 22.11 (2019), pp. 1761–1770. ISSN: 15461726. DOI: 10.1038/s41593-019-0520-2.

[11] Chung, S. and Abbott, L. F. "Neural population geometry: An approach for understanding biological and artificial neural networks". en. In: *Current Opinion in Neurobiology.* Computational Neuroscience 70 (Oct. 2021), pp. 137–144. ISSN: 0959-4388. DOI: 10.1016/j.conb.2021.10.010.

[12] Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., and Hinton, G. "Backpropagation and the brain". en. In: *Nature Reviews Neuroscience* (Apr. 2020). Publisher: Nature Publishing Group, pp. 1–12. ISSN: 1471-0048. DOI: 10.1038/s41583-020-0277-3.

[13] Oord, A. van den, Li, Y., and Vinyals, O. "Representation Learning with Contrastive Predictive Coding". In: (2018). arXiv: 1807.03748.

[14] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning.* PMLR. 2020, pp. 1597–1607.

[15] Rao, R. P. and Ballard, D. H. "Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects". In: *Nature Neuroscience* 2.1 (Jan. 1999), pp. 79–87.

[16] Li, N. and DiCarlo, J. J. "Unsupervised Natural Experience Rapidly Alters Invariant Object Representation in Visual Cortex". In: *Science* 321.5895 (2008), pp. 1502–1507. DOI: 10.1126/science.1160028.

[17] Keller, G. B. and Mrsic-Flogel, T. D. "Predictive Processing: A Canonical Cortical Computation". English. In: *Neuron* 100.2 (Oct. 2018), pp. 424–435. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2018.10.003.

[18] Gillon, C. J., Pina, J. E., Lecoq, J., Ahmed, R., Billeh, Y., Caldejon, S., Groblewski, P., Henley, T. M., Kato, I., Lee, E., Luviano, J., Mace, K., Nayan, C., Nguyen, T., North, K., Perkins, J., Seid, S., Valley, M., Williford, A., Bengio, Y., Lillicrap, T. P., Richards, B. A., and Zylberberg, J. "Learning from unexpected events in the neocortical microcircuit". en. In: *bioRxiv* (Jan. 2021). Publisher: Cold Spring Harbor Laboratory Section: New Results, p. 2021.01.15.426915. DOI: 10.1101/2021.01.15.426915.

[19] Wiskott, L. and Sejnowski, T. J. "Slow Feature Analysis: Unsupervised Learning of Invariances". In: *Neural Computation* 14.4 (Apr. 2002), pp. 715–770. ISSN: 0899-7667. DOI: 10.1162/089976602317318938.

[20] Sprekeler, H., Michaelis, C., and Wiskott, L. "Slowness: An Objective for Spike-Timing–Dependent Plasticity?" In: *PLoS Comput Biol* 3.6 (June 2007), e112. DOI: 10.1371/journal.pcbi.0030112.

[21] Illing, B., Ventura, J., Bellec, G., and Gerstner, W. "Local plasticity rules can learn deep representations using self-supervised contrastive predictions". In: *Advances in Neural Information Processing Systems* 34 (2021).

[22] Frémaux, N. and Gerstner, W. "Neuromodulated Spike-Timing-Dependent Plasticity, and Theory of Three-Factor Learning Rules". English. In: *Frontiers in Neural Circuits* 9 (2016). ISSN: 1662-5110. DOI: 10.3389/fncir.2015.00085.

[23] Kusmierz, L., Isomura, T., and Toyoizumi, T. "Learning with three factors: modulating Hebbian plasticity with errors". eng. In: *Current Opinion in Neurobiology* 46 (Sept. 2017), pp. 170–177. ISSN: 1873-6882. DOI: 10.1016/j.conb.2017.08.020.

[24] Bardes, A., Ponce, J., and LeCun, Y. "VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning". In: (2021), pp. 1–17. arXiv: 2105.04906.

[25] Oja, E. "Simplified neuron model as a principal component analyzer". In: *Journal of Mathematical Biology* 15.3 (1982), pp. 267–273. ISSN: 0303-6812. DOI: 10.1007/BF00275687.

[26] Gerstner, W. and Kistler, W. *Spiking neuron models*. Cambridge University Press New York, 2002.

[27] Sjöström, P. J., Turrigiano, G. G., and Nelson, S. B. "Rate, Timing, and Cooperativity Jointly Determine Cortical Synaptic Plasticity". In: *Neuron* 32.6 (Dec. 2001), pp. 1149–1164. ISSN: 0896-6273. DOI: 10.1016/S0896-6273(01)00542-6.

[28] Markram, H., Gerstner, W., and Sjöström, P. J. "A history of spike-timing-dependent plasticity". In: *Frontiers in Synaptic Neuroscience* 3 (2011), p. 4. DOI: 10.3389/fnsyn.2011.00004.

[29] Feldman, D. E. "The Spike-Timing Dependence of Plasticity". In: *Neuron* 75.4 (Aug. 2012), pp. 556–571. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2012.08.001.

[30] Löwe, S., O'Connor, P., and Veeling, B. S. "Putting an end to end-to-end: Gradient-isolated learning of representations". In: *Advances in Neural Information Processing Systems* 32.NeurIPS (2019). ISSN: 10495258. arXiv: 1905.11786.

[31] Miconi, T. "Multi-layer Hebbian networks with modern deep learning frameworks". In: *arXiv preprint arXiv:2107.01729* (2021).

[32] Artola, A., Bröcher, S., and Singer, W. "Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex". en. In: *Nature* 347.6288 (Sept. 1990), pp. 69–72. DOI: 10.1038/347069a0.

[33] Abraham, W. C. "Metaplasticity: Tuning synapses and networks for plasticity". en. In: *Nature Reviews Neuroscience* 9.5 (Jan. 2008), pp. 387–387. ISSN: 1471-003X. DOI: 10.1038/nrn2356.

[34] Lim, S., McKee, J. L., Woloszyn, L., Amit, Y., Freedman, D. J., Sheinberg, D. L., and Brunel, N. "Inferring learning rules from distributions of firing rates in cortical neurons". en. In: *Nature Neuroscience* 18.12 (Dec. 2015). Number: 12 Publisher: Nature Publishing Group, pp. 1804–1810. ISSN: 1546-1726. DOI: 10.1038/nn.4158.

[35] Miura, S. K. and Scanziani, M. *Erasing motion: Scrambling direction selectivity in visual cortex during saccades*. en. Tech. rep. Section: New Results Type: article. bioRxiv, Sept. 2021, p. 2021.03.30.437338. DOI: 10.1101/2021.03.30.437338.

[36] Földiak, P. "Forming sparse representations by local anti-Hebbian learning". In: *Biological cybernetics* 64.2 (1990), pp. 165–170.

[37] Vogels, T. P., Sprekeler, H., Zenke, F., Clopath, C., and Gerstner, W. "Inhibitory Plasticity Balances Excitation and Inhibition in Sensory Pathways and Memory Networks". In: *Science* 334.6062 (Dec. 2011), pp. 1569–1573. DOI: 10.1126/science.1211095.

[38] King, P. D., Zylberberg, J., and DeWeese, M. R. "Inhibitory Interneurons Decorrelate Excitatory Cells to Drive Sparse Code Formation in a Spiking Model of V1". en. In: *The Journal of Neuroscience* 33.13 (Mar. 2013), pp. 5475–5485. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.4188-12.2013.

[39] Lipshutz, D., Windolf, C., Golkar, S., and Chklovskii, D. "A Biologically Plausible Neural Network for Slow Feature Analysis". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 14986–14996.

[40] Matteucci, G. and Zoccolan, D. "Unsupervised experience with temporal continuity of the visual environment is causally involved in the development of V1 complex cells". In: *Science Advances* 6.22 (2020), eaba3742.

[41] Jing, L., Vincent, P., LeCun, Y., and Tian, Y. "Understanding Dimensional Collapse in Contrastive Self-supervised Learning". In: *International Conference on Learning Representations*. 2022.

[42] Hua, T., Wang, W., Xue, Z., Ren, S., Wang, Y., and Zhao, H. "On feature decorrelation in self-supervised learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9598–9608.

[43] Baldi, P., Sadowski, P., and Lu, Z. "Learning in the machine: Random backpropagation and the deep learning channel". en. In: *Artificial Intelligence* 260 (July 2018), pp. 1–35. ISSN: 0004-3702. DOI: 10.1016/j.artint.2018.03.003.

[44] Payeur, A., Guerguiev, J., Zenke, F., Richards, B. A., and Naud, R. "Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits". In: *Nature neuroscience* 24.7 (2021), pp. 1010–1019.

[45] Guerguiev, J., Lillicrap, T. P., and Richards, B. A. "Towards deep learning with segregated dendrites". en. In: *eLife* 6 (Dec. 2017), e22901. ISSN: 2050-084X. DOI: 10.7554/eLife.22901.

[46] Sacramento, J., Costa, R. P., Bengio, Y., and Senn, W. "Dendritic error backpropagation in deep cortical microcircuits". In: *arXiv:1801.00062 [cs, q-bio]* (Dec. 2017). arXiv: 1801.00062.

[47] Inglebert, Y. and Debanne, D. "Calcium and Spike Timing-Dependent Plasticity". In: *Frontiers in Cellular Neuroscience* 15 (2021), p. 374. ISSN: 1662-5102. DOI: 10.3389/fncel.2021.727336.

[48] Shouval, H. Z., Bear, M. F., and Cooper, L. N. "A Unified Model of NMDA Receptor-Dependent Bidirectional Synaptic Plasticity". en. In: *Proceedings of the National Academy of Sciences* 99.16 (June 2002), pp. 10831–10836. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.152343099.

[49] Pfister, J.-P. and Gerstner, W. "Triplets of Spikes in a Model of Spike Timing-Dependent Plasticity". en. In: *The Journal of Neuroscience* 26.38 (Sept. 2006), pp. 9673–9682. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.1425-06.2006.

[50] Clopath, C., Büsing, L., Vasilaki, E., and Gerstner, W. "Connectivity reflects coding: a model of voltage-based STDP with homeostasis". en. In: *Nature Neuroscience* 13.3 (Mar. 2010), pp. 344–352. ISSN: 1097-6256. DOI: 10.1038/nn.2479.

[51] Gjorgjieva, J., Clopath, C., Audet, J., and Pfister, J.-P. "A triplet spike-timing–dependent plasticity model generalizes the Bienenstock–Cooper–Munro rule to higher-order spatiotemporal correlations". In: *Proceedings of the National Academy of Sciences* 108.48 (Nov. 2011), pp. 19383–19388. DOI: 10.1073/pnas.1105933108.

[52] Toyoizumi, T., Kaneko, M., Stryker, M. P., and Miller, K. D. "Modeling the Dynamic Interaction of Hebbian and Homeostatic Plasticity". English. In: *Neuron* 84.2 (Oct. 2014), pp. 497–510. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2014.09.036.

[53] Tang, M., Yang, Y., and Amit, Y. "Biologically Plausible Training Mechanisms for Self-Supervised Learning in Deep Networks". In: *arXiv:2109.15089 [cs, q-bio]* (Feb. 2022). arXiv: 2109.15089. DOI: 10.3389/fncom.2022.789253.

[54] Graupner, M. and Brunel, N. "Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location". en. In: *Proceedings of the National Academy of Sciences* 109.10 (June 2012), pp. 3991–3996. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1109359109.

[55] Zenke, F., Hennequin, G., and Gerstner, W. "Synaptic Plasticity in Neural Networks Needs Homeostasis with a Fast Rate Detector". In: *PLOS Computational Biology* 9.11 (Nov. 2013), e1003330. DOI: 10.1371/journal.pcbi.1003330.

[56] Zenke, F. and Gerstner, W. "Hebbian plasticity requires compensatory processes on multiple timescales". en. In: *Philosophical Transactions of the Royal Society B* 372.1715 (Mar. 2017), p. 20160259. ISSN: 0962-8436, 1471-2970. DOI: 10.1098/rstb.2016.0259.

[57] Hennequin, G., Agnes, E. J., and Vogels, T. P. "Inhibitory Plasticity: Balance, Control, and Codependence". In: *Annual Review of Neuroscience* 40.1 (July 2017), pp. 557–579. ISSN: 0147-006X. DOI: 10.1146/annurev-neuro-072116-031005.

[58] Miehl, C. and Gjorgjieva, J. *Stability and learning in excitatory synapses by nonlinear inhibitory plasticity.* en. Mar. 2022. DOI: 10.1101/2022.03.28.486052.

[59] Rolls, E. T. and Stringer, S. M. "Invariant visual object recognition: A model, with lighting invariance". en. In: *Journal of Physiology-Paris.* Theoretical and Computational Neuroscience: Understanding Brain Functions 100.1 (July 2006), pp. 43–62. ISSN: 0928-4257. DOI: 10.1016/j.jphysparis.2006.09.004.

[60] Li, Y., Pogodin, R., Sutherland, D. J., and Gretton, A. "Self-supervised learning with kernel dependence maximization". In: *Advances in Neural Information Processing Systems* 34 (2021).

[61] Mehrer, J., Spoerer, C. J., Jones, E. C., Kriegeskorte, N., and Kietzmann, T. C. "An ecologically motivated image dataset for deep learning yields better models of human vision". en. In: *Proceedings of the National Academy of Sciences* 118.8 (Feb. 2021). Publisher: National Academy of Sciences Section: Biological Sciences. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.2011417118.

[62] Samonds, J. M., Geisler, W. S., and Priebe, N. J. "Natural image and receptive field statistics predict saccade sizes". en. In: *Nature Neuroscience* 21.11 (Nov. 2018). Number: 11 Publisher: Nature Publishing Group, pp. 1591–1599. ISSN: 1546-1726. DOI: 10.1038/s41593-018-0255-5.

[63] Zhuang, C., Yan, S., Nayebi, A., Schrimpf, M., Frank, M. C., DiCarlo, J. J., and Yamins, D. L. "Unsupervised neural network models of the ventral visual stream". In: *Proceedings of the National Academy of Sciences* 118.3 (2021), e2014196118.

[64] Konkle, T. and Alvarez, G. A. "A self-supervised domain-general learning framework for human ventral stream representation". In: *Nature communications* 13.1 (2022), pp. 1–12.

[65] Bakhtiari, S., Mineault, P., Lillicrap, T., Pack, C., and Richards, B. "The functional specialization of visual cortex emerges from training parallel pathways with self-supervised predictive learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 25164–25178.

[66] Schrimpf, M., Kubilius, J., Lee, M. J., Murty, N. A. R., Ajemian, R., and DiCarlo, J. J. "Integrative Benchmarking to Advance Neurally Mechanistic Models of Human Intelligence". English. In: *Neuron* 108.3 (Nov. 2020). Publisher: Elsevier, pp. 413–423. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2020.07.040.

[67] Lisman, J., Spruston, N., and Sjöström, P. J. "Questions about STDP as a general model of synaptic plasticity". In: *Frontiers in Synaptic Neuroscience* 2 (2010), p. 140. DOI: 10.3389/fnsyn.2010.00140.

[68] Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. "Barlow twins: Self-supervised learning via redundancy reduction". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12310–12320.

[69] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[70] Falcon, W. and Cho, K. "A Framework For Contrastive Self-Supervised Learning And Designing A New Approach". In: *arXiv preprint arXiv:2009.00104* (2020).

[71] Kingma, D. and Ba, J. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]* (Dec. 2014). arXiv: 1412.6980.

[72] Simonyan, K. and Zisserman, A. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[73] Coates, A., Ng, A., and Lee, H. "An analysis of single-layer networks in unsupervised feature learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 215–223.

[74] Krizhevsky, A., Hinton, G., et al. "Learning multiple layers of features from tiny images". In: (2009).

[75] Litwin-Kumar, A., Harris, K. D., Axel, R., Sompolinsky, H., and Abbott, L. F. "Optimal Degrees of Synaptic Connectivity". In: *Neuron* 93.5 (2017), 1153–1164.e7. ISSN: 10974199. DOI: 10.1016/j.neuron.2017.01.030.

[76] Mitchell, M., Muftakhidinov, B., Winchen, T., Schaik, B. van, Wilms, A., et al. "Engauge digitizer software". In: *Webpage: http://markummitchell. github. io/engauge-digitizer. Accessed* 11 (2017).

[77] Zenke, F. and Ganguli, S. "Superspike: Supervised learning in multilayer spiking neural networks". In: *Neural computation* 30.6 (2018), pp. 1514–1541.

[78] Zenke, F., Agnes, E. J., and Gerstner, W. "Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks". en. In: *Nature Communications* 6 (Apr. 2015), p. 6922. ISSN: 2041-1723. DOI: 10.1038/ncomms7922.

[79] Zenke, F. and Gerstner, W. "Limits to high-speed simulations of spiking neural networks using general-purpose computers". In: *Frontiers in Neuroinformatics* 8 (2014), p. 76. DOI: 10.3389/fninf.2014.00076.

Data availability

The deep learning tasks used the STL-10 [73] and CIFAR-10 [74] datasets, typically available through all major machine learning libraries. The original releases for these datasets can be found at `http://ai.stanford.edu/%7Eacoates/stl10/`, and `https://www.cs.toronto.edu/~kriz/cifar.html` respectively.

Code availability

The simulation code to reproduce the key results is publicly available at `https://github.com/fmi-basel/latent-predictive-learning`.

Acknowledgments

We thank all members of the Zenke Group for comments and discussions that shaped this project, and Atul Kumar Sinha for many helpful suggestions. We are particularly grateful to Julian Rossbroich for providing invaluable insights throughout the course of this work. This project was supported by the Swiss National Science Foundation [grant number PCEFP3_202981] and the Novartis Research Foundation.

Author contributions

F.Z. conceived the study. M.S.H. and F.Z. developed the theory. M.S.H. wrote DNN code, performed simulations, and analysis. F.Z. developed SNN code. M.S.H. and F.Z. wrote the manuscript.

Competing interests

The authors declare no competing interests.

Supplementary Figures

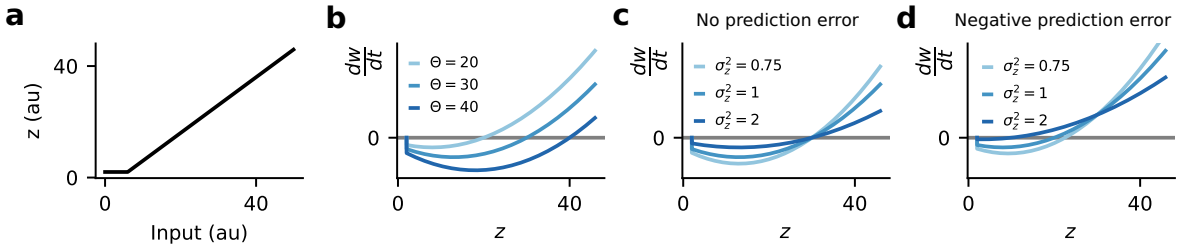

Figure S1: **Same as Figure 2a-d but for the rectified linear unit (ReLU) activation function used for the DNN simulations.** **(a)** The ReLU activation function shifted to approximately fit typical neuronal transfer functions. **(b)** Weight change induced by the LPL rule for co-varying input x and the postsynaptic activity z for different values of the plasticity threshold Θ. **(c)** Same as (b) but for different values of the variance of the postsynaptic activity with zero prediction error $\frac{dz}{dt} = 0$ and fixed mean activity $\bar{z} = 30$. **(c)** Same as (c) but with a negative prediction error $\frac{dz}{dt} = -10$.

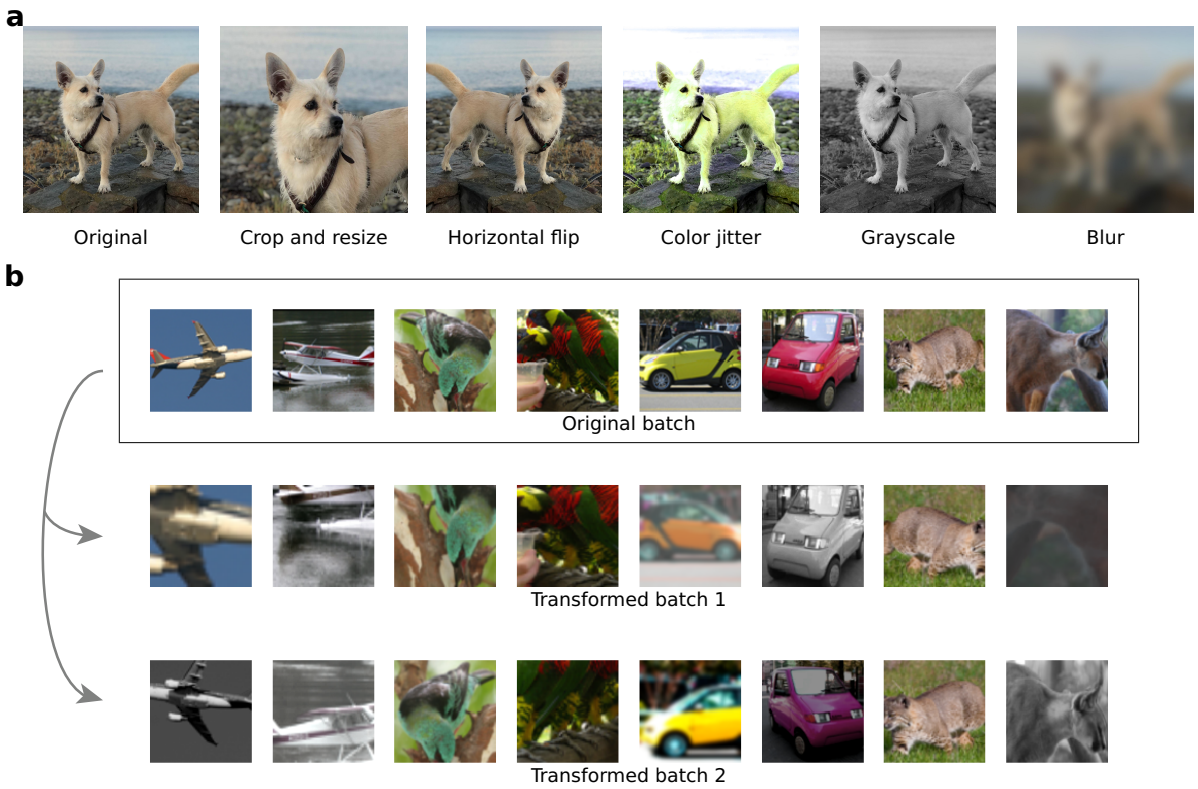

Figure S2: **Image augmentation model.** **(a)** Illustration of the image transformations used to generate natural image sequences as suggested by Chen et al. [14]. **(b)** Sample images from STL-10 and their transformed versions used for training the DNNs.

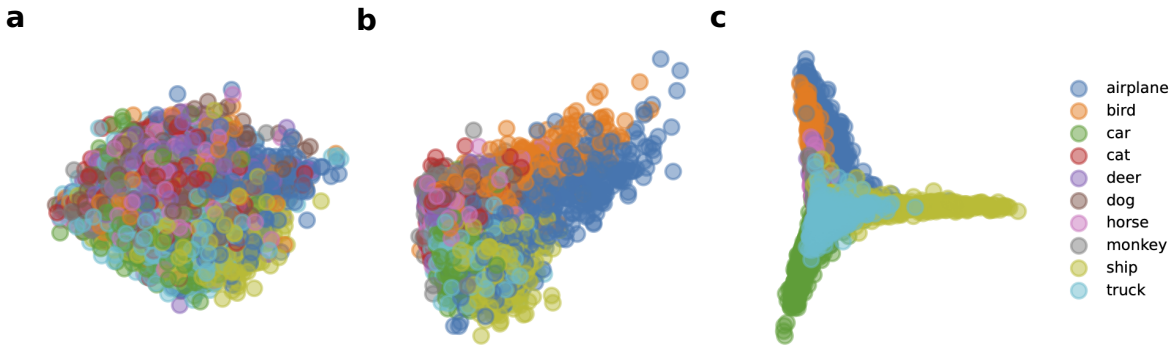

Figure S3: **Disentangling of object representations in the DNN. (a)** Data distribution of the STL-10 validation set along the first two principal components in pixel-space. Data corresponding to different object classes are highly entangled. **(b)** Same as (a) but along the principal components of representations in Layer 3 of the DNN after learning with LPL. Object classes are somewhat disentangled. **(c)** Same as (a) but along the principal components of representations in Layer 8 of the DNN. Object classes are highly disentangled.

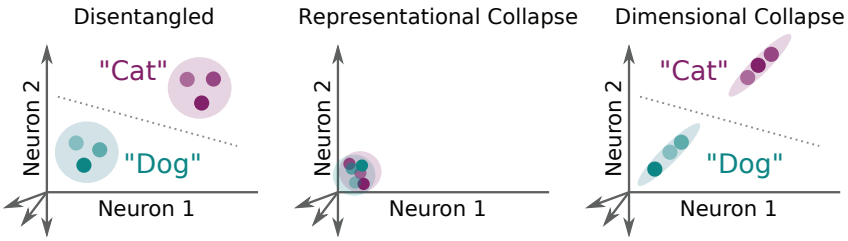

Figure S4: **Illustration of collapse modes typifying poorly disentangled features** Effectively disentangled representations (left) separate categories well with different representational directions encoding different relevant features. Purely predictive learning without counteracting Hebbian plasticity leads to collapsed representations (center), typically to zero activity levels. Dimensional collapse (right) is characterized by highly correlated activity across all neurons, indicating that only one relevant feature is encoded by all neurons, which is unlikely to be conducive to hierarchical feature extraction for non-trivial tasks such as object recognition.

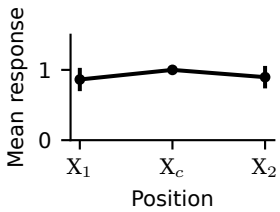

Figure S5: **Learned network representations are invariant to object position on the canvas.** Activity of neurons in the pretrained DNN's output layer in response to images at three positions on the canvas, normalized by each neuron's response to the center position, and averaged over neurons and over images.

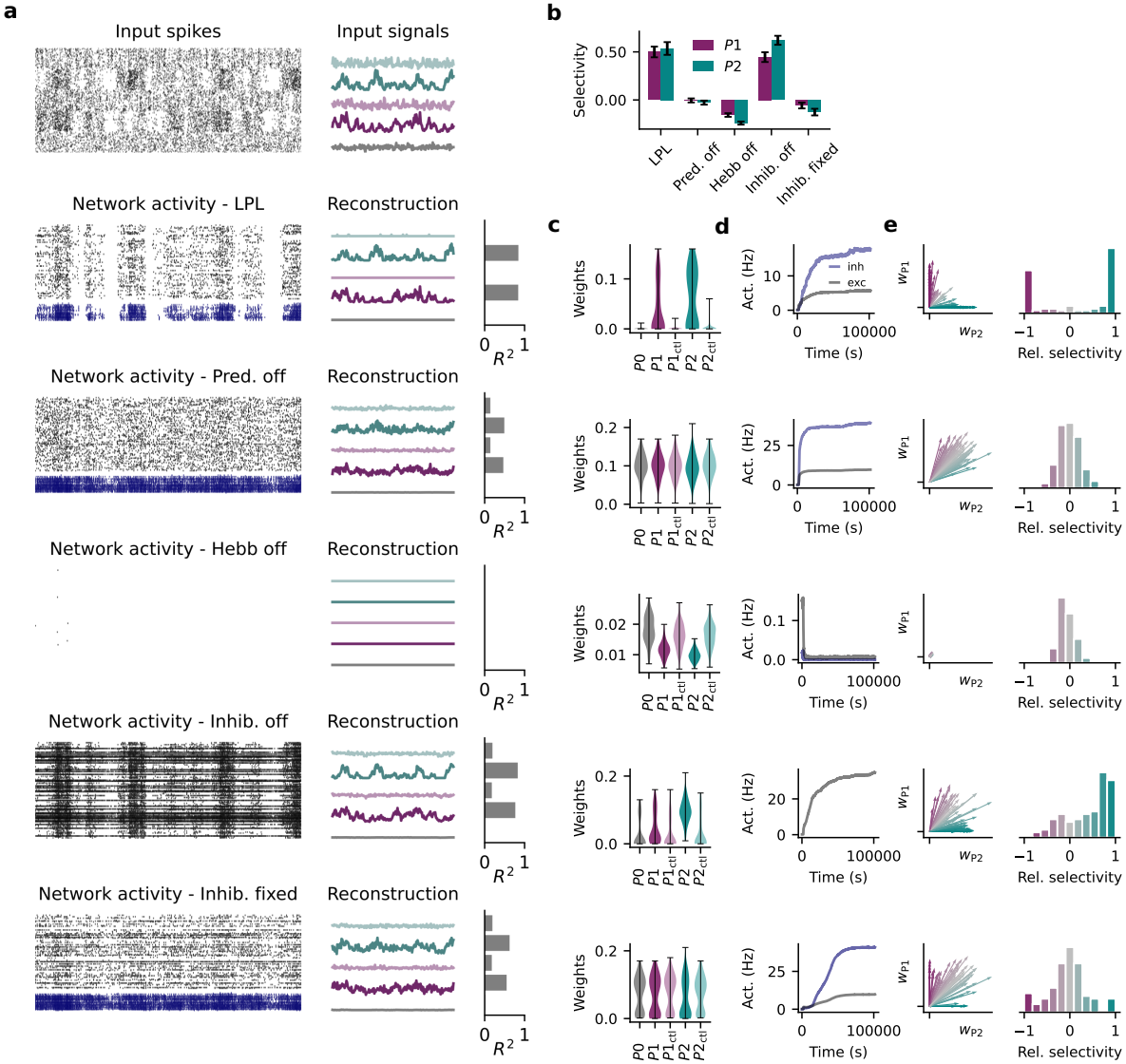

Figure S6: **Same as Figure 5 but with detailed controls.** **(a)** Snapshot of spiking activity (left) and underlying firing rate signals or their reconstructions (right) over 100 ms for the input and network populations. **(b)** Same as Fig. 5d showing signal selectivity learned with the different variations of spiking LPL given in (a). **(c)** Average synaptic connection strength grouped by input population for the different configurations in (a). LPL with plastic inhibition results in higher weights on the slowly varying signals relative to the shuffled controls, but not when the predictive or Hebbian term are disabled. Without inhibition or without inhibitory plasticity, connections from all populations are strong with a small preference for $P2$. **(d)** Average firing rates over 100 s bins throughout training for the configurations in (a). Firing rates saturate with the inhibitory neurons settling at a higher firing rate when learning with spiking LPL with inhibition, even when the predictive term is disabled or the inhibition is not plastic. Activity collapses without the Hebbian term, whereas firing rates diverge without inhibition. **(e)** Averaged weight vectors from populations $P1$ and $P2$ onto each excitatory neuron (left) and distribution of the excitatory neurons' relative selectivity between the two populations (right). Different neurons are exclusively selective to either $P1$ pr $P2$ under spiking LPL with inhibitory plasticity. Without the predictive term, or the Hebbian term, few if any neurons are selective to one population over the other. Moreover, weights collapse to small values without the Hebbian term. When inhibition is removed altogether, a few neurons become exclusively selective to $P2$, but the weight vectors are not well-decorrelated. Without inhibitory plasticity, a few weight vectors are well-decorrelated, but most neurons are not preferentially selective to either signal.

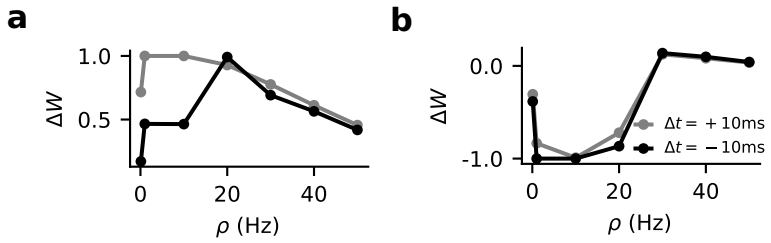

Figure S7: **Learning threshold determines the sign of plasticity (a)** Weight changes as a function of repetition frequency ρ for positive and negative relative spike timings ($\Delta t = \pm 10\,\text{ms}$) with $\sigma^2(t = 0) = 0$ and $\bar{S}_i(t = 0) = 0$. **(b)** Same as (a) but for $\bar{S}_i(t = 0) = 50$.

		Avg	V1	V2	V4	IT	behavior
End-to-end	supervised	.256	.314	.311	.390	.180	.086
	LPL	.246	.300	.293	.361	.202	.074
	untrained	.213	.317	.223	.316	.170	.038
Layer-wise	supervised	.181	.269	.212	.291	.140	-.001
	LPL	.138	.252	.113	.188	.116	.020

Figure S8: **Brain-Scores [66] of the DNN used in this article trained on STL-10 using different objectives.** To match the input image dimension of our model 96×96 to the dimensions required by Brain-Score, we first (linearly) down-sampled the input images. One possible reason for the generally low behavioral scores might be that the network was trained on STL-10, which does not contain several classes that are relevant for the underlying benchmark (e.g., wrench, knife, bear, etc).

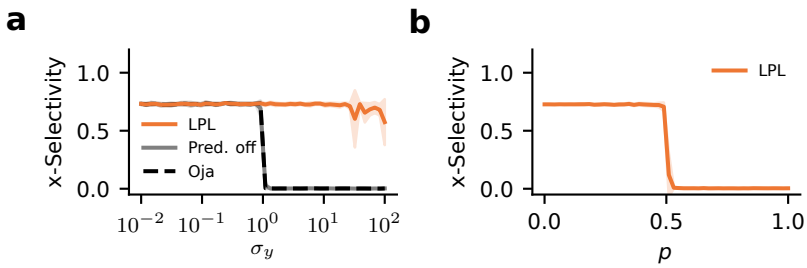

Figure S9: **LPL is robust to noise. (a)** Same as Figure 2f but for high rates of noisy transitions between clusters in the training data sequence with $p = 0.2$ (Methods). A neuron learning with LPL still consistently becomes selective to cluster identity even with noisy transitions. **(b)** Cluster selectivity as a function of the probability of noisy cross-cluster transitions in the data sequence with $\sigma_y = 1$. LPL drives selectivity to cluster identity only below $p = 0.5$, i.e, only as long as cluster identity remains the slow feature.

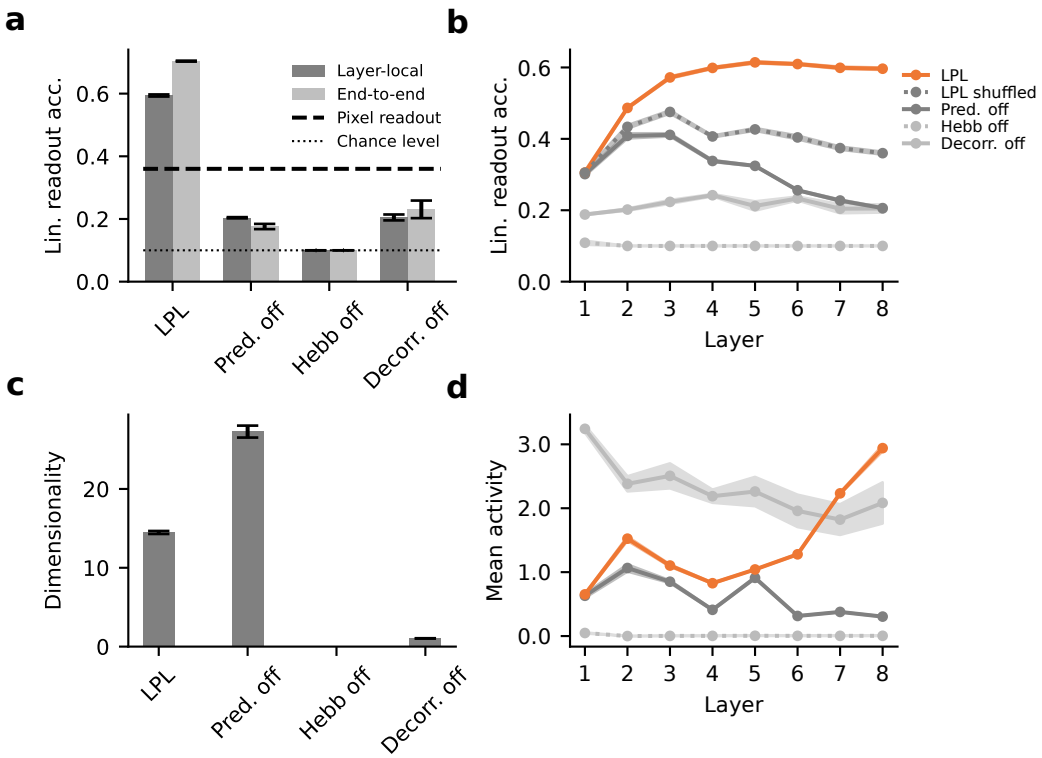

Figure S10: **Same as Figure 3 but for the CIFAR-10 dataset.** **(a)** Linear readout accuracy of object categories decoded from representations at the network output after training it on natural image data for different learning rules in layer-local (dark) as well as the end-to-end configuration (light). **(b)** Linear readout accuracy of the internal representations at different layers of the DNN after layer-local training. **(c)** Dimensionality of the internal representations for the different learning rule configurations shown in (b). **(d)** Mean neuronal activity at different layers of the DNN after training for the different learning rule variants shown in (b).

Supplementary Tables

	STL-10	CIFAR-10
VGG-11	63.2±0.3	59.4±0.4
VGG-11 without bias terms	58.7	54.6
VGG-11 with stale mean and variance estimates	59.3±0.6	47.7±0.6
VGG-11 without GAP in the loss computation	29.2±1.8	47.7±0.3
VGG-11 without GAP (down-sampled inputs)	45.3±0.5	-

Table S1: Linear classification accuracy on the STL-10 and CIFAR-10 datasets for layer-locally trained LPL with the base VGG-11 architecture, and several modified versions. VGG-11 without bias terms refers to the standard architecture including GAP but without any bias terms in the convolutional layers. Stale mean and variance estimates denote calculating the representational mean and variance from the previous batch. VGG-11 without GAP refers to computing and optimizing the LPL objective on the unpooled feature maps. Downsampled inputs correspond to the architecture without GAP, but now using STL-10 images subsampled to a lower resolution of 32×32. Reported error values correspond to SEM over four simulations with different random seeds.

	Single neuron (2D dataset)	Single neuron (digit dataset)	Network simulations
λ_1	1	1	1
λ_2	-	-	10
Optimizer	SGD	SGD	Adam[a]
Learning rate η	$\min(10^{-2}, 10^{-2}/\sigma_y)$	10^{-2}	10^{-3} [b]
Weight decay η_w	0.15	0.15	1.5×10^{-6}
Batch size B	200	200	1024
Training steps	$\max(10000, 100\sigma_y)$	1000	$\sim$78000[c]

[a] With default parameters from Kingma et al. [71].
[b] Reduced to zero during training using a cosine learning rate schedule.
[c] $\sim$35000 for CIFAR-10.

Table S2: Hyperparameter values for DNN simulations.

	STL-10		CIFAR-10	
	Layer-local (%)	End-to-end (%)	Layer-local (%)	End-to-end (%)
LPL	63.2±0.3	72.5±0.1	59.4±0.4	70.4±0.2
Neg. samples	77.0±0.2	81.0±0.3	67.4±0.3	76.5±0.1
Supervised	70.8±0.3	77.8.5±0.3	81.7±0.2	87.1±0.2
Pixel-space decoding	31.6		35.9	

Table S3: Extended version of Table 1 showin linear classification accuracy on the STL-10 and CIFAR-10 datasets for LPL and different baseline models (Methods). Error values correspond to SEM over four simulations with different random seeds.

Parameter	Value	Parameter	Value
τ^{mem}	20 ms	U^{exc}	0 mV
τ^{ampa}	5 ms	U^{inh}	-80 mV
τ^{gaba}	10 ms	U^{leak}	-70 mV
τ^{nmda}	100 ms		
τ^{thr}	5 ms		

Table S4: Table summarizing the neuronal parameters.

Source	Destination	Connection probability	Initial weight value
input	exc	0.1	0.15
exc	inh	local (see Methods)	0.4
inh	exc	0.5	0.1
inh	inh	0.1	0.4

Table S5: Summary of SNN connectivity parameters.

Supplementary Notes

S1 Equivalence of the objective function and learning rule formulations

Here, we show that the objective functions defined in Eqs. (4), (5), and (6) indeed result in the LPL rule (Eq. (7)).

Predictive component. We recall that the predictive objective $\mathcal{L}_{\text{pred}}$ is the mean squared difference between neuronal activity in consecutive time steps.

$$\mathcal{L}_{\text{pred}} = \frac{1}{2MB}\sum_{b=1}^{B}\|z^b - SG(z^b(t-\Delta t))\|^2 = \frac{1}{2MB}\sum_{b=1}^{B}\sum_{i=1}^{M}\left(z_i^b - SG(z_i^b(t-\Delta t))\right)^2$$

Taking the derivative with respect to the network weights results in the following learning rule

$$\frac{\partial \mathcal{L}_{pred}}{\partial W_{ij}} = \frac{1}{MB}\sum_{b=1}^{B}\left(z_i^b - z_i^b(t-\Delta t)\right) f'(a_i^b)x_j^b \tag{28}$$

which does not require backpropagation through time due to the Stopgrad function.

Hebbian component. The Hebbian component minimizes the negative logarithm of the variance of neuronal activity:

$$\mathcal{L}_{Hebb} = \frac{1}{2M}\sum_{i=1}^{M} -\log\left(\sigma_i^2\right)$$

where $\bar{z}_i = \text{SG}(\frac{1}{B}\sum_{b=1}^{B} z_i^b)$ and $\sigma_i^2 = \frac{1}{B-1}\sum_{b=1}^{B}\left(z_i^b - \bar{z}_i\right)^2$ are the mean and variance of the activity of the ith output neuron over the minibatch. The corresponding learning rule is obtained as the negative gradient of this loss function with respect to the weights W. The gradient itself is given by:

$$\frac{\partial \mathcal{L}_{Hebb}}{\partial W_{ij}} = -\frac{1}{M(B-1)\sigma_i^2}\sum_{b=1}^{B}(z_i^b - \bar{z}_i)f'(a_i^b)x_j^b \tag{29}$$

Note that the objective and the resulting gradient is essentially unchanged upto a scaling factor when we use running estimates of the variance with a time constant τ instead of batch estimates.

$$\begin{aligned}
\mathcal{L}_{Hebb}(t) &= \frac{1}{2M}\sum_{i=1}^{M} -\log\left(\sigma_i^2(t)\right) \\
&= \frac{1}{2M}\sum_{i=1}^{M} -\log\left((1-\tau)(z_i(t) - \bar{z}_i(t))^2 + \tau\text{SG}(\sigma_i^2(t-1)) + \epsilon\right) \\
\frac{\partial \mathcal{L}_{Hebb}}{\partial W_{ij}}(t) &= -\frac{(1-\tau)}{M\sigma_i^2(t)}(z_i^b(t) - \bar{z}_i(t))f'(a_i^b(t))x_j^b(t)
\end{aligned}$$

Decorrelation component. Finally, the decorrelation objective is the decorrelation loss function as the sum of the squared off-diagonal terms of the covariance matrix between units.

$$\mathcal{L}_{\text{decorr}} = \frac{1}{2(B-1)(M^2-M)}\sum_{b=1}^{B}\sum_{i=1}^{M}\sum_{k\neq i}(z_i^b - \bar{z}_i)^2(z_k^b - \bar{z}_k)^2$$

which gives the gradient:

$$\frac{\partial \mathcal{L}_{\text{decorr}}}{\partial W_{ij}} = \frac{1}{(B-1)(M^2-M)} \sum_{b=1}^{B} (z_i^b - \bar{z}_i) f'(a_i^b) x_j^b \sum_{k \neq i} (z_k^b - \bar{z}_k)^2 \tag{30}$$

The full learning rule. The combined weight updates for a descent along the sum of the three gradients in Eqs. (28), (29), and (30) in a single-layer network finally yields the LPL rule including the decorrelation component:

$$\begin{aligned} \Delta W_{ij} &= -\eta \left(\frac{\partial \mathcal{L}_{\text{pred}}}{\partial W_{ij}} + \lambda_1 \frac{\partial \mathcal{L}_{\text{Hebb}}}{\partial W_{ij}} + \lambda_2 \frac{\partial \mathcal{L}_{\text{decorr}}}{\partial W_{ij}} \right) \\ &= \eta \frac{1}{MB} \sum_{b=1}^{B} \left(-(z_i^b - z_i^b(t-\Delta t)) + \lambda_1 \frac{\alpha}{\sigma_i^2} (z_i^b - \bar{z}_i) - \lambda_2 \beta (z_i^b - \bar{z}_i) \sum_{k \neq i} (z_k^b - \bar{z}_k)^2 \right) f'(a_i^b) x_j^b \end{aligned} \tag{31}$$

where $\alpha = \frac{B}{B-1}$ and $\beta = \frac{B}{(B-1)(M-1)}$ are the appropriate normalizing constants, and λ_1 and λ_2 are the loss coefficients. Including weight decay in the weight update finally yields the LPL rule for a network given in Eq. (7).

S2 Relating the Hebbian component of LPL to Oja's rule

To see the relation of the Hebbian component of LPL with the classic Oja's rule [1], we consider the case of a single output neuron ($M = 1$), with no nonlinearity ($f'(a) = 1$), along with the assumption that the input is zero-centered ($\bar{x}_j = 0$). Consequently, $\bar{z} = \sum_j W_j \bar{x}_j = 0$ and $\sigma_z^2 = \left\langle (z - \bar{z})^2 \right\rangle = \left\langle z^2 \right\rangle$, which yields a very simple Hebbian learning rule for descending the gradient in Eq. (29):

$$\Delta W_j(t) = -\frac{\partial \mathcal{L}_{Hebb}(t)}{\partial W_{ij}} = \frac{z(t)x_j(t)}{\langle z^2 \rangle}$$

This update rule along with a weight decay (with coefficient η_w) yields a learning rule that, on average, is equivalent to Oja's rule up to a scaling factor, and in fact has exactly the same non-zero fixed point when $\eta_w = 1$, but with different convergence dynamics because of the multiplication by $1/\langle z^2 \rangle$.

$$\begin{aligned} \Delta W_j(t) &= \frac{z(t)x_j(t)}{\langle z^2 \rangle} - \eta_w W_j \\ \langle \Delta W_j \rangle &= \frac{\langle z x_j \rangle}{\langle z^2 \rangle} - \eta_w W_j \\ &= \frac{1}{\langle z^2 \rangle} \left(\langle z x_j \rangle - \eta_w W_j \left\langle z^2 \right\rangle \right) \end{aligned} \tag{32}$$

Oja's rule is presented below for reference

$$\begin{aligned} \Delta W_j^{Oja}(t) &= z(t)x_j(t) - W_j z^2 \\ \left\langle \Delta W_j^{Oja} \right\rangle &= \langle z x_j \rangle - W_j \left\langle z^2 \right\rangle \end{aligned} \tag{33}$$

S3 Importance of the variance-dependent modulation of Hebbian learning

To analytically understand the importance of the variance-dependent scaling of the Hebbian term in the learning rule, we first looked at the synthetic two-dimensional learning task from Fig. 2e, and modeled the behaviour of the LPL loss functions under a particular distribution of the representations. Specifically, we considered the case where the two input clusters map to a mixture of two normal distributions in *representation space* with each Gaussian component corresponding to the representations of one input cluster. Each of the two Gaussians are assumed to have a standard deviation of r, with their means symmetrically located on either side of zero with a distance of $D = 4r$ between their centers. We used this setting to investigate how the predictive and Hebbian loss terms of LPL behave under different values of the representational variance by co-varying r and D.

The predictive loss in this case is proportional to the expected squared difference $\langle (z_1 - z_2)^2 \rangle$ between two independently drawn samples $z_{1/2}$ from the same Gaussian, i.e, $\mathcal{L}_{\text{pred}} = 2r^2$. The overall variance of the representations is $\sigma_z^2 = r^2 + (\frac{D}{2})^2$ (variance of the Gaussian mixture). Under this representational distribution, we studied the overall loss function obtained by combining the predictive loss with different variance-maximising losses, each a different

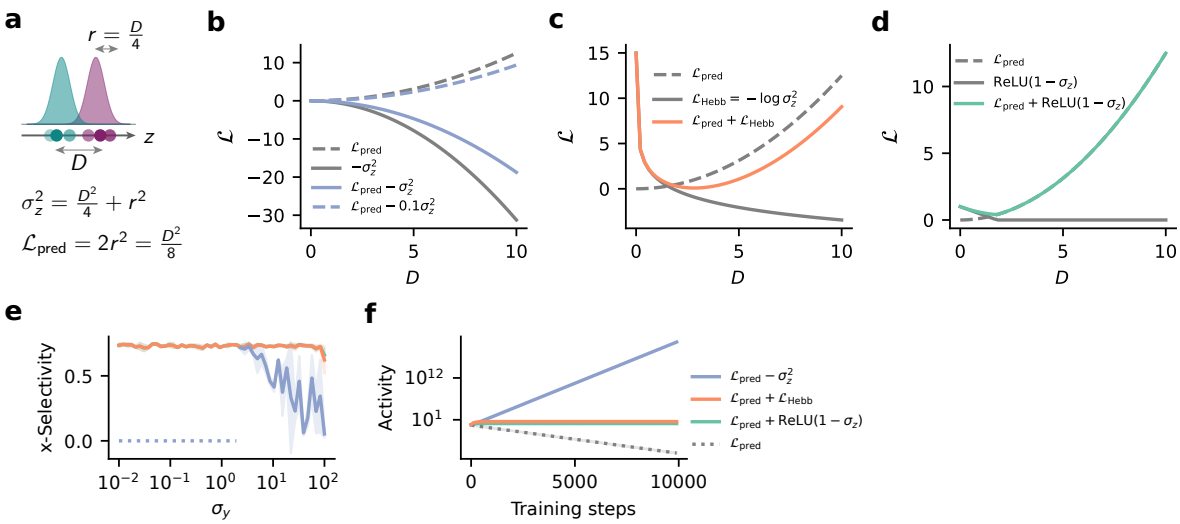

Figure S11: **Variance-dependent modulation of Hebbian learning objective is crucial for stable learning.** **(a)** Example setting with two clusters distance D apart in representation space. The size of each cluster $r = \frac{D}{4}$ is assumed to scale with D. In this case, representational variance is $\frac{D^2}{4} + r^2$. **(b)** Total loss combining the predictive loss with a naive variance maximising loss ($-\sigma_z^2$) as a function of cluster separation D. Depending on the coefficient of the variance loss, the global minimum of the loss is either at $D = \infty$ (solid blue) or at $D = 0$ (dashed blue), implying runaway LTP or collapse respectively. **(c)** Same as (b) but using $\mathcal{L}_{\text{Hebb}} = -\log \sigma_z^2$, the Hebbian objective optimized by LPL instead of the naive variance objective. Here, the predictive loss starts dominating at higher values of D inducing a global minimum at $D \approx 3$. **(d)** Same as (b) but the variance loss is now $\text{ReLU}(1 - \sigma_z)$, the objective that was optimized in the VICReg model [2]. Here as well, the predictive loss dominates at higher values of D, inducing a global minimum at $D \approx 2$. **(e)** Cluster selectivity learned by LPL on the two-dimensional synthetic sequence from Fig. 2e with the standard predictive loss combined with the different variance losses from (a), (b) and (c). Dotted sections indicate simulations where learning diverged. **(f)** Mean output activity over training time for LPL on the two-dimensional synthetic sequence with $\sigma_y = 1$ for the different cases from (a), (b) and (c).

decreasing function of the variance, i.e, $\mathcal{L}_{\text{var}} = f(\sigma_z^2)$. Specifically, we considered the cases $\mathcal{L}_{\text{var}} = -\sigma_z^2$, $\mathcal{L}_{\text{var}} = -\log \sigma_z^2$, and $\mathcal{L}_{\text{var}} = \text{ReLU}(1 - \sigma_z)$. These loss functions are plotted in Fig. S11b-d respectively along with $\mathcal{L}_{\text{pred}}$, and the resulting full LPL objective in each case. With the naive variance maximization objective $\mathcal{L}_{\text{var}} = -\sigma_z^2$, the full objective is dominated by the variance term at large values of D (Fig. S11b), and therefore inherently drives unstable learning. It is not possible to remedy this situation by simply using a smaller weight for the variance loss, for instance, by weighting $\mathcal{L}_{\text{var}}$ with a small weight of 0.1. This is because downweighting the variance objective simply moves the loss minimum at $D = \infty$ to $D = 0$, the exact situation of collapse the variance objective is meant to prevent (Fig. S11b). In contrast, using $\mathcal{L}_{\text{var}} = -\log \sigma_z^2$ (Fig. S11c), or $\mathcal{L}_{\text{var}} = \text{ReLU}(1-\sigma_z)$ (Fig. S11d) along with $\mathcal{L}_{\text{pred}}$ constitute loss landscapes with minima at finite non-zero values of D. This is because these variance objectives only dominate at low values of D, but have diminishing influence with growing D allowing the predictive term to dictate learning, and preventing runaway activity.

We validated these scaling arguments with learning simulations using each of the three proposed learning objectives on the synthetic two-dimensional sequence learning task from Fig. 2e. We found that using the naive variance maximization objective $\mathcal{L}_{\text{var}} = -\sigma_z^2$ results in poor learning of cluster selectivity, whereas $\mathcal{L}_{\text{var}} = -\log \sigma_z^2$ and $\mathcal{L}_{\text{var}} = \text{ReLU}(1 - \sigma_z)$ prove effective (Fig. S11e). Furthermore, the naive variance objective indeed suffers from runaway instability (Fig. S11f).

S4 Predictive feature selected by LPL strictly depends on temporal contiguity properties of the input sequence

The slow or "predictive" feature picked up by a single neuron learning with LPL purely depends on the temporal order of stimuli that it is exposed to. One would expect, then, that it is possible to manipulate the learned feature by altering the temporal sequence of stimuli.

To illustrate that this is indeed the case, we designed a predictive learning task similar to that in Fig. 2e using a subset of images from the MNIST handwritten digit dataset [3]. Specifically, we sampled 2000 images from this dataset corresponding to the digits "five" and "six", equally distributed between the two classes. We generated sample inputs by embedding these 28×28 grayscale images in a 56×56 blank canvas at either the top-left or bottom-right location. Because we sought to demonstrate that changing the temporal transition structure qualitatively changes neuronal selectivity, we considered two types of sequences with distinct temporal contiguity properties (Fig. S12). In the Digit Sequence we preserved digit identity (either five or six) across subsequent input images, while changing their position on the canvas, whereas in the Location Sequence, we presented different digits at the same location in successive inputs. Therefore, the predictive feature is location in the Location Sequence and digit identity in the Digit Sequence. Furthermore, digit identity and digit location were approximately aligned with the first two principal components of the data which account for 30 % and 5 % of the explained variance respectively (Fig. S12).

We again exposed a single rate neuron model to these two sequence types, while allowing the plastic input connections to evolve according to the LPL rule. After convergence, we measured neuronal selectivity to digit identity and location. We measured selectivity to location and digit identity with the same measure defined in Eq. (9), only changing what inputs fall into clusters 1 and 2 in each case. Concretely, we measured selectivity to digit identity by setting $\langle z_1 \rangle$ to be the mean response to the digit five (at any location) and $\langle z_2 \rangle$ the mean response to the digit six. Finally, we set $\langle z_1 \rangle$ and $\langle z_2 \rangle$ to the mean responses to digits at the two locations regardless of digit identity in order to measure location selectivity.

At initialization with random weights, the neuron was partially selective to both location and digit identity (Fig. S12f). However, subsequent training with LPL rendered the neuron purely selective to either location or digit identity depending on which sequence it was exposed to during training. Yet, when the predictive term was turned off, the specific sequence did not matter and the neuron always became selective to location, which coincides with the direction of highest variance in the data (PC1; Supplementary Fig. S12). Finally, we confirmed that Oja's rule showed the same behavior (Fig. S12f). Thus, a neuron learning with LPL finds temporally contiguous features in high-dimensional sequential data rather than the direction of largest variance, and the the temporally contiguous feature that is learned is strictly determined by the temporal sequence of the stimuli the neuron is exposed to.

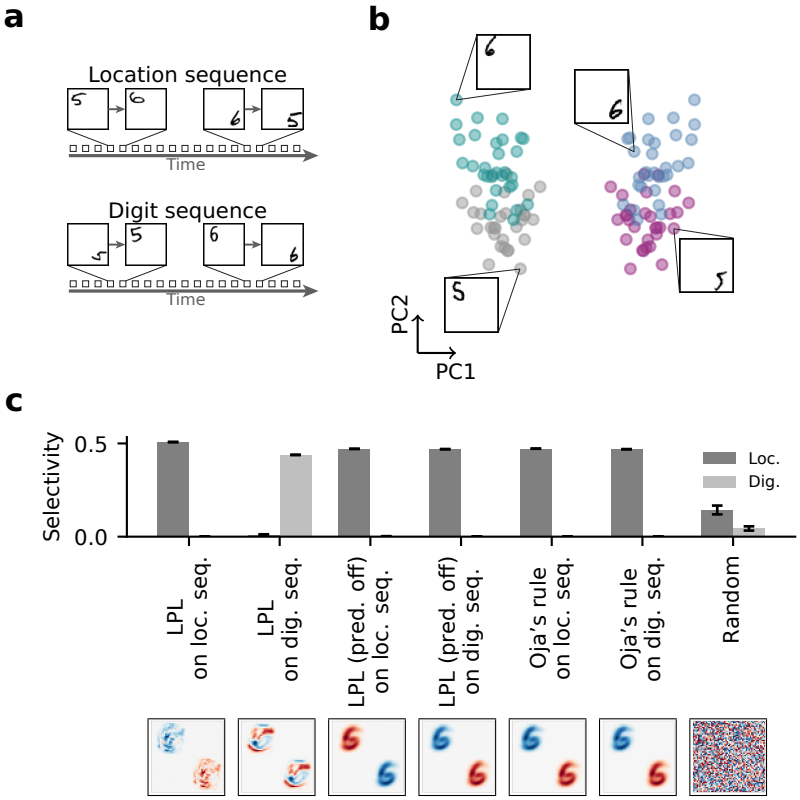

Figure S12: **Temporal contiguities determine features learned under LPL. (a)** Schematic of the two kinds of temporal sequences (Methods) in which subsequent inputs were either different digits shown at the same location ("Location Sequence") or the same digit presented at a different location ("Digit Sequence"). **(b)** Scatter plot of the first two principal components of the synthetic digit dataset consisting of randomly sampled handwritten digits in one of the two locations. The principal components closely correspond to location (PC1) and digit identity (PC2). The four digit-position categories are indicated by color. Insets show representative examples from each category. **(c)** Emergent feature selectivity of a single neuron exposed to the two sequence modalities while learning under different rules (top), and the resulting input weights learned in each case (bottom). Under LPL, the neuron's selectivity mirrors the temporally preserved (predictive) property in each sequence, i.e., location in the Location Sequence and digit identity in the Digit Sequence. However, the specific sequence does not matter for Oja's rule or for LPL without the predictive term. In these cases, the neuron always becomes selective to location, the direction of maximum variance. Error bars indicate SEM over ten random seeds.

S5 Details of the deep neural network architecture

For all DNN simulations, we used the convolutional layers of the VGG-11 architecture consisting of eight blocks each containing 3×3 convolutions, the ReLU activation function followed by a 2×2 max-pool operation in some blocks (detailed architecture description provided below).

```
VGG11Encoder(
  (blocks): ModuleList(
    (0): ConvBlock(
      (module): Sequential(
        (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      )
    )
    (1): ConvBlock(
      (module): Sequential(
        (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      )
    )
    (2): ConvBlock(
      (module): Sequential(
        (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): Identity()
      )
    )
    (3): ConvBlock(
      (module): Sequential(
        (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      )
    )
    (4): ConvBlock(
      (module): Sequential(
        (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): Identity()
      )
    )
    (5): ConvBlock(
      (module): Sequential(
        (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      )
    )
    (6): ConvBlock(
      (module): Sequential(
        (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): Identity()
      )
    )
    (7): ConvBlock(
      (module): Sequential(
        (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
```

```
        (1): ReLU(inplace=True)
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      )
    )
  )
  (pooler): AdaptiveAvgPool2d(output_size=(1, 1))
)
```

Furthermore, for the simulations modeling unsupervised learning in the IT, we used an adaptive average pooling layer with spatial output dimensions of 13×1. This ensured that the final pooling layer preserved spatial separation along the canvas itself so that the final feature map consisted of 13×1 512-dimensional vectors. We added a fully connected layer on top of these feature maps to finally get a single 512-dimensional feature vector per image.

References

[1] Oja, E. "Simplified neuron model as a principal component analyzer". In: *Journal of Mathematical Biology* 15.3 (1982), pp. 267–273. ISSN: 0303-6812. DOI: 10.1007/BF00275687.

[2] Bardes, A., Ponce, J., and LeCun, Y. "VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning". In: (2021), pp. 1–17. arXiv: 2105.04906.

[3] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.