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

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

[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 meaningful internal representations in deep sensory networks with plastic synaptic connections. Experience-dependent plasticity presumably exploits temporal contingencies between sensory inputs to build these internal representations. However, the precise mechanisms underlying plasticity remain elusive. We derive a local synaptic plasticity model inspired by self-supervised machine learning techniques that shares a deep conceptual connection to Bienenstock-Cooper-Munro (BCM) theory and is consistent with experimentally observed plasticity rules. We show that our plasticity model yields disentangled object representations in deep neural networks without the need for supervision and implausible negative examples. In response to altered visual experience, our model qualitatively captures neuronal selectivity changes observed in the monkey inferotemporal cortex in-vivo. Our work suggests a plausible learning rule to drive learning in sensory networks 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 objects or categories that remain invariant to different views and their context. What processing is required to enable 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 a simple linear combination of their inputs. *Disentangling* behaviorally relevant categories requires deep sensory networks that yield invariant linearly separable category representations (Fig. 1b; [3]). Crucially, successful disentangling requires a specific network connectivity which is thought to be shaped through experience-dependent plasticity [4]. However, current data-driven plasticity models are unable to account for the emergence of disentangled representations in deep biological neural networks.

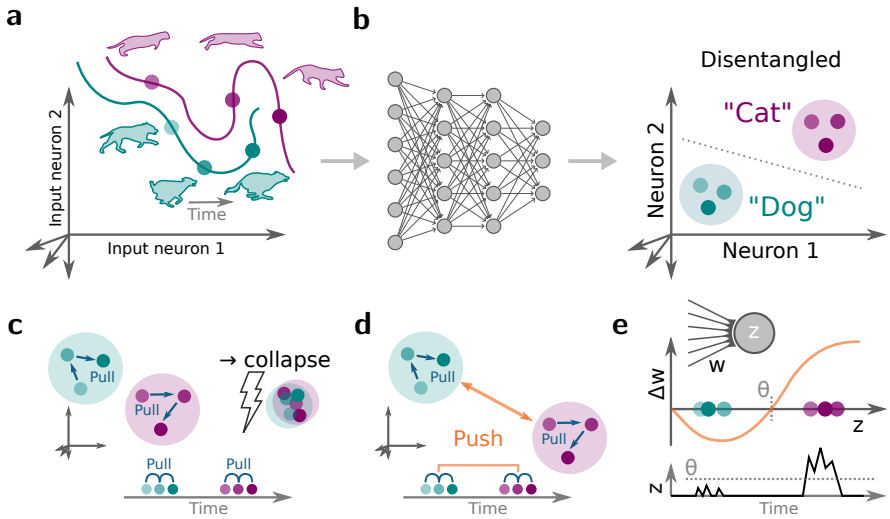

Figure 1: **Disentangling sensory stimuli with plastic neural networks.** **(a)** Schematic of neuronal responses at the sensory input level. The neuronal response patterns of different stimuli correspond to points in a high dimensional space. The response patterns from different stimulus classes, e.g., 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 learning influences latent representations during predictive learning (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).

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 growing datasets and larger network models has matured into powerful machine learning algorithms 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, 8]. This similarity suggests that DNNs are valuable tools to elucidate neural information processing in the brain [9–11]. Yet, two central issues remain concerning their training and 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 [12–14]. Second, the notion of supervised learning is not biologically plausible because it requires neurons to receive specific targets or labels that are not conceivably available to animals and humans.

Self-supervised learning (SSL), a family of unsupervised machine learning algorithms, may offer a remedy for the latter issue. Crucially, SSL does not need labeled data but instead requires that the internal network representations belonging to related inputs predict one another [15–18]. In other words, 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 a network has to 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 [19–21], for instance, by extracting slowly varying features from their sensory inputs as done in slow feature analysis (SFA) [22, 23]. In other words, predictive learning posits that biological networks change their connections with the goal of "pulling" together related internal representations, e.g., representations of input stimuli that frequently occur close together in time (Fig. 1c).

However, one major issue with this strategy is that without any forces opposing the representational pull, such learning 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 [24–26], but it requires constant switching of the sign of plasticity rule depending on whether two successive inputs are related to each other or not. Still, 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 [27]. Interestingly, variance maximization is a known signature of Hebbian plasticity [28, 29], which has been found ubiquitously in the brain [30–34]. 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.

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; [31, 35, 36]), with a predictive component inspired by SFA [22, 23] 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. Finally, we demonstrate that LPL captures central findings of unsupervised visual learning experiments in monkeys. In summary, we argue that LPL constitutes a putative plasticity mechanism for representation learning in sensory networks.

Results

To understand whether and how we can reconcile Hebbian plasticity with the notion of predictive learning, we derived a plasticity model that combines both aspects and optimizes a self-supervised objective function (Methods; Supplementary Material S1). The plasticity model combines a predictive term with a modified variant of the BCM learning rule [1, 37] commonly used to explain receptive field formation and ocular dominance plasticity [38]. Specifically, we propose the following synaptic learning rule

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

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, 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 [22, 23] 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 activity constraints, the predictive term leads to collapsing neuronal activity levels [22]. 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, 37]. Together with $f'(a(t))$, the derivative of the neuronal activation function (Fig. 2a) and the presynaptic term $x_j(t)$ give the learning rule its characteristic BCM-like shape and a moving threshold (Fig. 2b). While the detailed shape of the learning rule depends on the choice of neuronal activation function (Supplementary Fig. S1), its qualitative behavior remains unchanged.

However, our model constitutes a departure from the BCM model because the moving threshold depends only linearly on $\bar{z}(t)$ (Fig. 2b) and the added predictive term $-\frac{dz}{dt}$ provides an additional mechanism acting on the plasticity threshold (Fig. 2c) to oppose sudden changes of the neuron's output. Additionally, Eq. (1) regulates the slope of the learning curve at the plasticity threshold as a function of the postsynaptic variance $\sigma_z^2(t)$ (Fig. 2d), i.e., the strength of the Hebbian term is inversely proportional to an online estimate of the variance $\sigma_z^2(t)$ of the postsynaptic activity. We will see in the following that these modifications are key to representation learning from 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 their input, we refer to Eq. (1) as the latent predictive learning (LPL) rule.

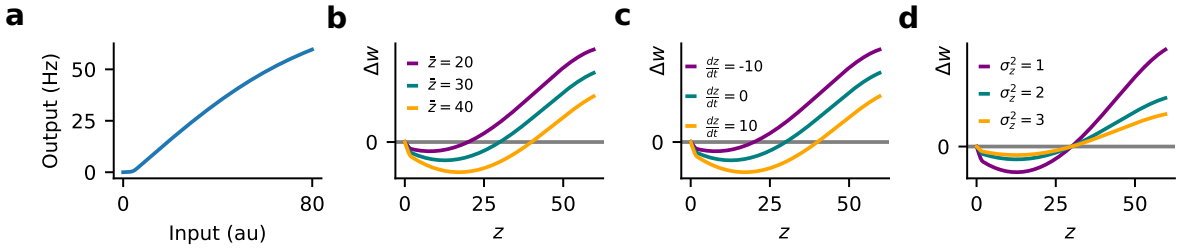

Figure 2: **LPL extends BCM theory by adding a variance- and rate-of-change dependence.** **(a)** Example of a typical neuronal transfer function as used in Rajan et al. [39]. **(b)** Weight change induced by the LPL rule for co-varying input x and the postsynaptic activity z for different values of the mean-dependent threshold $\bar{z}$. The functional shift of the threshold is reminiscent of the BCM rule. **(c)** Same as (b) but for different rates of change $^{dz}/_{dt}$ of the postsynaptic activity. **(d)** Same as (b) but for different values of the variance of the postsynaptic activity.

LPL finds contiguous features in temporal data

We wondered whether LPL would render a single neuron selective to temporally contiguous features, i.e, features that are preserved or slowly varying over successive temporal inputs. To that end, we first designed a synthetic two-dimensional learning task in which we parametrically controlled the proportion of predictable changes between subsequent observations relative to the total fluctuations in the dataset (Fig. 3a; Methods). Specifically, the data sequence consisted of noisy inputs distributed over two clusters separated along the x-axis. Subsequent 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, where 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 trains the neuron to be selective to the cluster identity regardless of the value of σ_y (Fig. 3b). However, without the predictive term in the learning rule, the neuron became selective to cluster identity only at low values of σ_y, but not at higher values which correspond to large unpredictable changes. This is the expected behaviour because without the predictive term, the learning rule becomes purely Hebbian, which drives selectivity to directions of the highest variance. To illustrate this point, we repeated the same simulated learning experiments with Oja's rule, a classic Hebbian learning rule, and found identical qualitative behaviour. In contrast to Oja's rule the Hebbian term in LPL is modulated by the inverse variance of the postsynaptic activity which allows the predictive term to dominate the learning dynamics when there is sufficient postsynaptic variance (Supplementary Material S2). The absence of such variance-dependent modulation, however, leads to run-away LTP and thus precludes predictive learning from establishing cluster selectivity (Supplementary Material S3). Hence, the predictive term of LPL in combination with variance-modulated Hebbian plasticity renders neurons selective to cluster identity independent of the co-variance structure of the data.

Next, we sought to confirm that Hebbian learning is still required to prevent representational collapse. To that end, we simulated learning with $\sigma_y = 1$ under LPL with either of the two terms disabled (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 activity as expected, whereas this behavior was avoided with the Hebbian

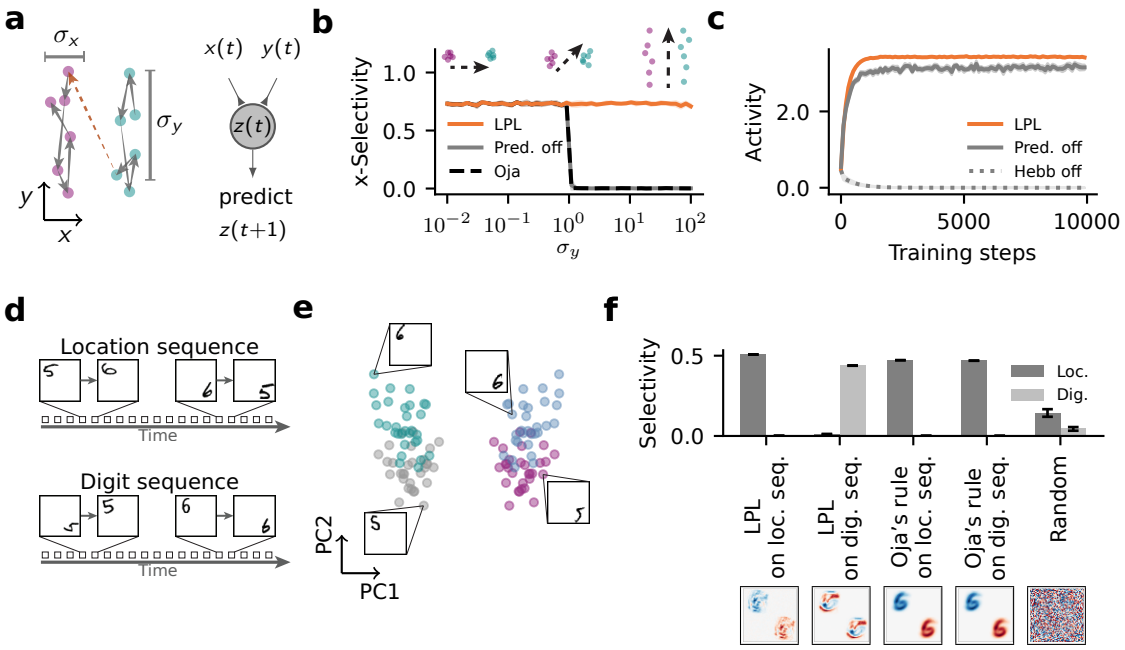

Figure 3: **LPL finds temporal contiguities in sequential data.** **(a)** Illustration of the two-dimensional synthetic data generating process. Consecutive data points stay within the same cluster separated along the x-direction and are drawn independently from the corresponding normal distribution centered in that cluster (left). These data are fed into a linear neuron that learns via LPL (right). **(b)** Cluster selectivity of the features learned by LPL with and without the predictive term 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 $\sigma_y = 1$. The LPL rule without the predictive component shows the same behavior. **(c)** 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. **(d)** 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"). **(e)** 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. **(f)** 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 and the neuron always becomes selective to location, the direction of maximum variance.

term present (Fig. 3c). Therefore, LPL's Hebbian component prevents activity collapse while the predictive component steers neuronal selectivity towards temporally contiguous features.

To further investigate whether LPL can drive learning of temporally contiguous features in high-dimensional inputs, we designed a similar predictive learning task using images of handwritten digits placed randomly at two diagonally opposite locations on a canvas (Methods). We sought to demonstrate that changing the temporal transition structure qualitatively changes neuronal selectivity. To that end, we considered two types of sequences with distinct temporal contiguity properties (Fig. 3d). 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 (Fig. 3e).

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 quantified Digit Selectivity as the normalized difference between average responses to the two digit classes. Correspondingly, we measured Location Selectivity as the normalized average difference between the two locations. At initialization with random weights, the neuron was partially selective to both location and digit identity (Fig. 3f). 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. S2). Finally, we confirmed that Oja's rule showed the same behavior (Fig. 3f). Thus, a neuron learning with LPL finds temporally contiguous features in high-dimensional sequential data rather than the direction of largest variance.

LPL disentangles representations in deep hierarchical networks

In vision, objects present in a scene induce temporally contiguous features. For instance, we may look at an object from different angles or the object may move, meanwhile the projections cast on our retina change as our visual system performs multiple saccades to scan the scene (Fig. 4a; [40]). Hence, we wondered whether training a DNN with LPL on sequences of images in which object identity is the temporally contiguous feature would result 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 in a layer-local manner, meaning that no error signals were backpropagated between layers. Additionally we included a layer-local decorrelation term, that was necessary to prevent neurons within a given layer from becoming correlated (Methods). In biological neural networks this role could plausibly be achieved through plasticity at inhibitory synapses [41–43]. 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. S3; Methods). We exposed our network model to these visual data until learning converged and evaluated the linear decodability of object category from the learned representations using a linear classifier.

We found that in networks trained with LPL, object categories could be linearly decoded with an accuracy of $(62 \pm 0.2)\%$ (Fig. 4b; Table 1), suggesting that the network has formed partially disentangled representations at its output. To elucidate the contributions of the dif-

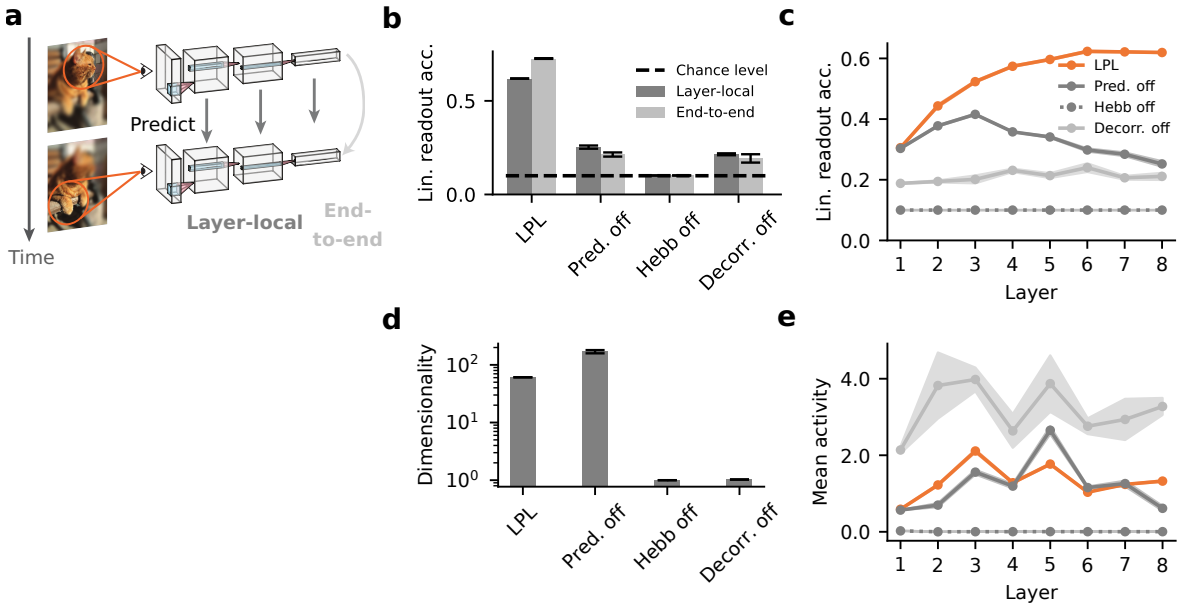

Figure 4: **LPL disentangles representations in DNNs.** **(a)** Schematic of the DNN trained using LPL. We distinguish two learning strategies: Layer-local and end-to-end learning. In layer-local LPL, each layer's learning objective ($\mathcal{L}_i$) is to predict representations within the same layer, 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) for 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. 6). "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 close to chance level at 10% (dotted line). **(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 rapidly falls toward chance level when turning off either the Hebbian or decorrelation component. **(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.

ferent learning rule components to disentangling, we conducted several ablation experiments. First, we repeated the same simulation but now excluding the predictive term, which resulted in an accuracy of $(21.3 \pm 2.2)\%$, which is *lower* than the linear readout accuracy of a classifier trained on the pixels of the raw 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 struggles with learning disentangled representations in deep networks [10, 44–47]. We measured a similar drop in accuracy when we disabled either the Hebbian or the decorrelation component during learning (Fig. 4b). Convolutional DNNs trained through supervised learning use depth to progressively separate representations as activity propagates through their layers [5, 6, 48]. We sought to understand whether this also applies to networks trained with LPL. To answer this question, we measured the linear readout accuracy of the 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. 4c), whereas this was not the case when any of the three components of LPL was disabled. Together, these results suggest that each of the three components of LPL is crucial for learning disentangled representations in hierarchical DNNs.

	STL-10		CIFAR-10	
	Layer-local (%)	End-to-end (%)	Layer-local (%)	End-to-end (%)
LPL	61.9±0.2	72.6±0.4	53.5±0.5	70.4±1.1
Neg. samples	78.5±0.2	84.4±0.3	74.4±0.5	82.5±0.5
Supervised	57.0±1.0	50.5±2.7	76.0±0.9	89.4±0.4
Pixel-space decoding	29.4		31.8	

Table 1: Linear classification accuracy on the STL-10 and CIFAR-10 datasets for LPL along with different baseline models. Error values correspond to the standard deviation over four simulations with different random seeds.

Two common causes for failure to disentangle representations are representational collapse and dimensional collapse, which results from excessively high correlations between neurons [49, 50]. To disambiguate between these two possibilities, we computed the dimensionality of the output 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 ≈ 20 or ≈ 100 respectively (Fig. 4d). When we disabled the Hebbian term, this resulted in zero activity across all layers (Fig. 4e), 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. 4e). Finally, excluding the predictive component of LPL did not cause collapse of either activity levels or in dimensionality. However, it still led to a rapid decrease in linear readout accuracy in the higher layers, showing that the predictive term is crucial for disentangling object representations in DNNs (Fig. 4) 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 notably different from typical DNN training which is end-to-end, i.e., training optimizes an objective at the network's output layer. End-to-end training is typically achieved with backpropagation [51], an algorithm that assigns credit or blame for output errors to neurons at 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, 14, 52, 53]. In any case, we wanted to know whether optimizing an LPL learning objective only at the output layer, while allowing gradient information to flow into intermediate layers would also yield disentangled representations. 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.4b; 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. Such optimization requires dedicated neural circuitry for solving the credit assignment problem which may be implemented by dedicated neuronal or circuit elements in neurobiology [53–55].

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) which has been interpreted as a consequence of unsupervised temporal slowness learning (UTL) [20], a principle directly captured by the predictive term in our learning rule. In Li et al. [20], 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. 5a). The experimenters differentiated between normal exposures in which the object does not change during the saccade and "swap exposures" whereby the initially presented object was consistently swapped out for a different one while the monkey saccaded to a specific target location $\mathrm{X_{swap}}$. Hence, such swap exposures created an "incorrect" temporal association between one object at position $\mathrm{X_{swap}}$ and a different one at the animal's center of gaze $\mathrm{X_c}$. For each pair of swap objects, either the location above or below the center of gaze was chosen as $\mathrm{X_{swap}}$, and transitions from the opposite peripheral position $\mathrm{X_{nonswap}}$ to the center $\mathrm{X_c}$ 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 $\mathrm{X_{swap}}$ while preserving its selectivity at the non-swap position $\mathrm{X_{nonswap}}$ (Fig. 5b).

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. [20] based on the DNN introduced in the previous section. Although DNNs have several important architectural differences to cortical networks, they are currently among the best models of visual processing in the ventral stream [7]. 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 image (object) position on the canvas (Supplementary Fig. S4), 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. Specifically, for a given pair of images of 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. Importantly, the weights in the network model evolved according to the LPL rule during exposure to these perturbed inputs and we recorded neuronal responses in the network's output layer.

We observed that the neuronal selectivity between preferred inputs P (initial preference de-

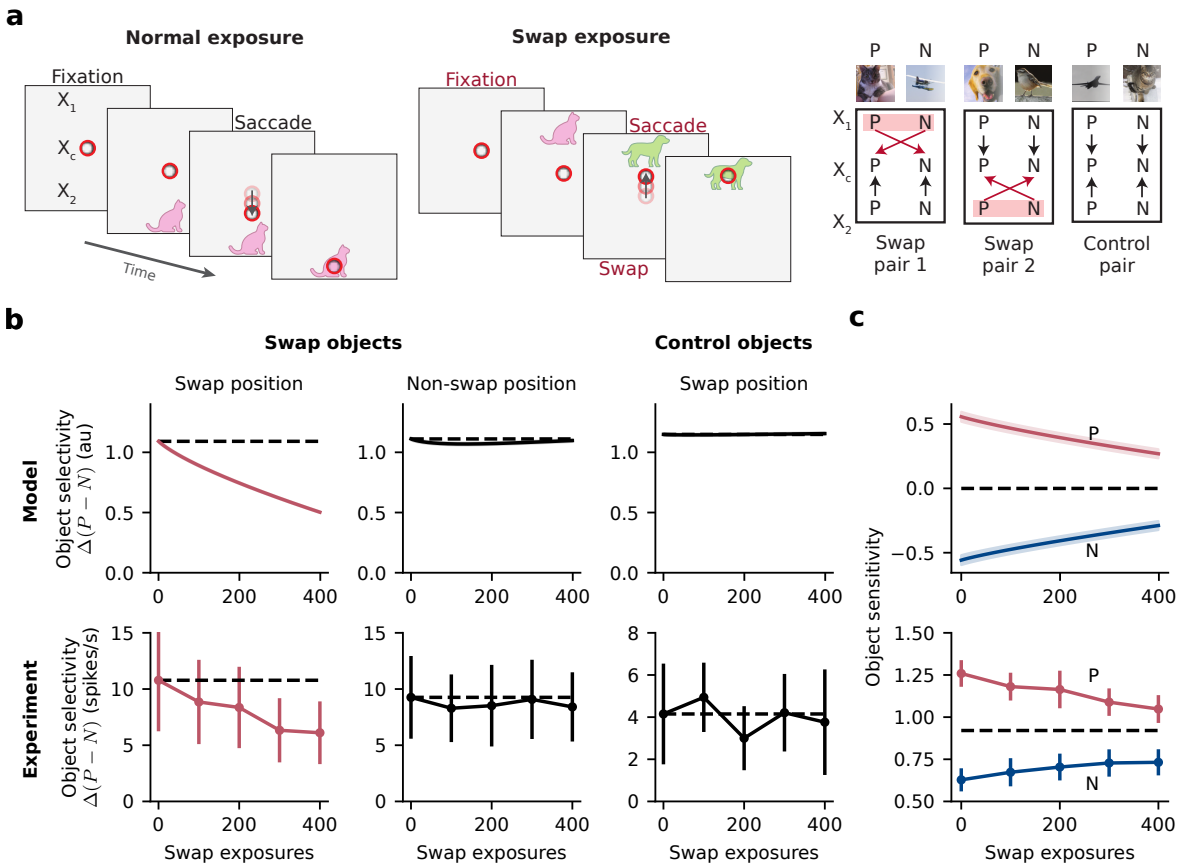

Figure 5: **LPL captures invariance learning in the primate inferotemporal cortex (IT)** **(a)** Schematic of the simulation setup modeled after Li et al. [20]. The inputs to the model consist of images of objects presented at three different positions X_1, X_c, and X_2 on a blank canvas. Following the 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. **(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 [20], 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).

fined on a per-neuron basis) in comparison to non-preferred stimuli N in the model qualitatively reproduced the evolution of object selectivity reported in the experiments (Fig. 5b). 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, consistent with the experiment, we observed that object selectivity for pairs of control objects not used during the swap training protocol showed no changes (Fig. 5b). Finally, further analysis revealed that the origin of these selectivity changes between objects P and N was due to both increases in responses to N and decreases in responses to P which was also observed in the experiments (Fig. 5c). Thus, LPL can account for neuronal selectivity changes observed in monkey IT during in-vivo unsupervised visual learning experiments.

Discussion

In this article, we have shown that a combination of predictive and Hebbian learning rules can disentangle object representations through mere exposure to temporal data in which object identity is preserved across successive inputs. We first demonstrated that LPL's predictive component renders individual neurons selective to temporally contiguous features while the Hebbian component prevents representational collapse. Second, we showed that when LPL is applied locally to all the neurons in a DNN while also ensuring decorrelated neuronal activity, the network layers learn to gradually disentangle object representations when successive inputs with high probability provide different views of the same object. Moreover, optimizing networks in an end-to-end manner with the LPL objective (cf. Eq. 2; Table 1) leads to further disentangled representations, suggesting that biological credit assignment mechanisms could improve learning in deep sensory networks. Crucially, we verified that neither predictive nor Hebbian learning on their own are able to learn disentangled representations. Finally, we demonstrated that LPL in a DNN qualitatively captures the representational changes observed in unsupervised learning experiments in monkey IT [20].

Importantly, LPL does not require negative samples. Instead, it only relies on three distinct plasticity mechanisms that use information that is locally available to them. First, the predictive plasticity term depends on temporal changes in firing activity induced by successive stimuli. Second, the plasticity rule depends on changes to mean neuronal activity and its temporal fluctuations, both of which a synapse could extract through some form of temporal averaging. Finally, LPL in networks requires a circuit mechanism that decorrelates neuronal activity, which could be achieved through inhibitory circuits and their plasticity [41–43]. Because of these properties, we argue that LPL constitutes a plausible learning mechanism that could serve as a critical driver of neural plasticity in sensory networks.

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. First, our LPL learning objective is a direct extension of work by Bardes et al. [27], a recent SSL approach that relies on variance regularization instead of negative samples to prevent representational collapse. Our model uses virtually the same mechanism while building a conceptual bridge from variance regularization to BCM theory. To that end, we used a logarithmic dependence on the variance instead of a piece-wise linear inactivation function (Supplementary Fig S7), which yields a smooth derivative and immediately exposes the relationship to Hebbian learning. However, unlike Bardes et al. [27] who used an end-to-end learning approach in which the objective function was formulated on the embeddings at the network output, 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, allows formulating the weight updates as local learning rules. Such layer-local learning is similar to works combining a contrastive predictive coding (CPC) objective with greedy layer-wise training thereby reducing or avoiding the need for backpropagation [46, 56]. In a similar vein, recent work by Illing et al. [24] showed that such greedy layer-wise learning is directly linked to local Hebbian learning rules. However, these previous studies used contrastive loss functions which rely on an additional predictor network and negative samples to avoid representational collapse. In contrast, LPL does neither require negative samples nor a predictor network while also dispensing with the need for end-to-end training.

Moreover, the reliance on the log-variance term in LPL inherently induces a postsynaptic plasticity threshold that switches between LTD and long-term potentiation (LTP) and naturally relates the model to BCM theory. Still, there are also important differences. First, LPL adds a predictive term which is missing in classic BCM theory. Second, the strength of the BCM-like contribution in our learning rule is scaled by the inverse variance of the neuronal activity. This scaling effectively switches off Hebbian learning when output fluctuations grow sufficiently large which in combination with the predictive loss provides a natural stabilizing mechanism for the learning which works even in the absence of a nonlinear firing rate dependence of the moving threshold (Supplementary Fig. S7). The link to BCM learning is important because several experimental studies have confirmed the existence of a BCM-like plasticity threshold both in-vitro [31, 35, 36] and in-vivo [57], and subsequently integrated into computational plasticity models [58–62]. However, unequivocal evidence for a moving threshold as predicted by BCM theory remains scant, specifically with regard to its nonlinear dependence on the postsynaptic activity [36] and the timescales involved [38, 63–65]. It will be important for future studies to further investigate whether and how neural circuit mechanisms could provided the suitable threshold adaptation. While purely neuronal mechanisms are certainly a conceivable option to implement such a moving threshold mechanism [36], alternative mechanisms could rely on inhibitory circuits and associated inhibitory plasticity [66] which may further provide conceptual links to experiments demonstrating gating of LTP through the pervasive circuit motif of disinhibition [67, 68].

The notion of predictive learning is at the heart of SFA [22]. However, a central distinguishing feature of LPL is that it does not impose a hard constraint on neuronal variance. While SFA explicitly restricts neuronal variance to be one, LPL merely enforces a soft variance constraint similar to VICReg [27] which prevents the variance from collapsing to zero. Additionally, the variance constraint is not an explicit constraint, but is part of the overall optimization objective which due to its logarithmic dependence does not necessarily enforce high variance. But how does SFA relate to plasticity mechanism observed in the brain? One biological plasticity mechanism that has been studied extensively is spike-timing-dependent plasticity (STDP) [33] and effective models exist that link the spike-timing-dependence with the typical activity dependence of BCM theory. Interestingly, some work has linked SFA conceptually to STDP [23] thereby creating a possible entry point to relate our plasticity model to biologically plausible spike-timing-dependent plasticity models.

Our study has several limitations which we aim to address in future work. First, our study is limited to simple toy problems and 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 limited 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 reasons 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 visual 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 world evolves around us. It is left as future work to show how LPL responds to more ethologically realistic sensory modalities [69] and video input while further combining them with plausible models of saccadic eye movement [70, 71].

Another limitation arises from the model architectures we studied. In this article, we limited ourselves to strictly feedforward network models without recurrence and dedicated excitatory and inhibitory cell types. This was primarily done for simplicity and because it allowed us to rely on established convolutional DNN architectures and to leverage powerful deep learning software libraries to train our models on extensive amounts of data. A direct consequence of this choice is that neuronal decorrelation in our model is not achieved through recurrent interactions with inhibitory interneurons as is presumably the case in cortical networks, but rather by introducing a redundancy reduction objective commonly used in SSL [27, 72]. While this objective function adds a non-local component to the learning rule, previous work suggests that a similar circuit-level effect can be achieved through more realistic models with recurrent plastic inhibition [41–43]. We plan on verifying this point in future work.

Moreover, we have evaluated our model on its ability to disentangle object representations in DNNs and to reproduce neuronal activity signatures of unsupervised visual learning in experiments performed in monkey IT [20]. An insightful next step will be to directly compare the representational similarity between our models and monkey visual areas along the hierarchy similar to previous work [7]. Further, for the linear readout accuracy that we employed as a metric, there remains a performance gap between LPL and negative sample-based or supervised methods (Table 1). Addressing these points may necessitate further conceptual work and will presumably require up-scaling our models to larger networks and datasets.

Despite these limitations, our model makes some concrete predictions about the underlying plasticity rules. As we have shown, modulating the strength of Hebbian plasticity as a function of the variance of the postsynaptic activity is essential to LPL (Supplementary Fig. S7) since it facilitates self-tuning the interaction strength between predictive and Hebbian learning which ultimately results in disentangled representations. A direct prediction of our model is that the predictive contribution to the learning rule should be best observable when postsynaptic activity is highly variable, while it should barely be 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 not been active in a while, as it may be the case in slice electrophysiology experiments, shows stronger Hebbian learning, than neurons which take part in in-vivo activity. Since predictive learning should manifest essentially in anti-Hebbian plasticity signatures that depend on changes of the postsynaptic firing activity on short timescales of tens to hundreds of milliseconds, such changes should be reflected in nontrivial modifications of the STDP window. One can further speculate that such differences in the postsynaptic firing statistics between in-vitro and in-vivo plasticity experiments could also explain why classic STDP does not seem to be the full story [34, 73, 74] and why the STDP window can be altered or difficult to observe in-vivo [75, 76].

In summary, to fathom how we can reconcile established data-driven plasticity models with experimental support with theoretically motivated learning rules such as SFA and SSL is a promising research direction where much remains to be learned. A central open question is how the different components of such learning rules can conspire with the rich structure of local microcircuits found in neurobiology to yield useful representations at the network level. Crucially, we should expect to discover signatures of predictive learning, as we discussed it here, across the entire hierarchy from the neuronal level to the circuit- and the network level. We have made one step in this direction which opens the door for several exciting future research avenues.

Methods

Plasticity model

The LPL rule consists of three distinct parts. First, a predictive component which minimizes neuronal output fluctuations for inputs that occur close in time. Second, a Hebbian component which prevents representational collapse. Finally, we use a decorrelation term in all network simulations to prevent excessive correlations between neurons within the same layer in a network. Each part of the learning rule minimizes a specific objective function:

$$\mathcal{L}_{\text{LPL}} = \mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{Hebb}} + \mathcal{L}_{\text{decorr}} \tag{2}$$

We obtain the equivalent learning rule as the negative gradient of the total objective $\mathcal{L}_{\text{LPL}}$.

In the following, we consider a neural 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 two-time step time horizon 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.

$$\mathcal{L}_{\text{pred}}(t) = \frac{1}{2}\sum_{b=1}^{B} \|z^b(t) - \text{SG}(z^b(t - \Delta t))\|^2 = \frac{1}{2}\sum_{b=1}^{B}\sum_{i=1}^{M} \Big(z_i^b(t) - \text{SG}(z_i^b(t - \Delta t))\Big)^2 \tag{3}$$

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:

$$\mathcal{L}_{\text{Hebb}}(t) = \sum_{i=1}^{M} -\log\left(\sigma_i^2(t)\right) \tag{4}$$

where $\bar{z}_i(t) = \text{SG}(\frac{1}{B}\sum_{b=1}^{B} z_i^b(t))$ and $\sigma_i^2(t) = \alpha \sum_{b=1}^{B} \left(z_i^b(t) - \bar{z}_i(t)\right)^2$ are the current estimates of the mean and variance of the activity of the ith output neuron over the minibatch with $\alpha = \frac{1}{B-1}$. Note that we must also detach the gradient with respect to the mean activity in the estimate of the variance in order to obtain exact equivalence to the learning rule in Eq. (2). Although we here estimate mean and variance of neuronal activity over a minibatch of training examples, these quantities can also be estimated as running averages resulting in effectively the same learning rule (see Supplementary Material S1).

Decorrelation component. Finally, we use a decorrelation objective to prevent excessive covariance between different neurons in the same layer as suggested previously [27, 41, 72, 77]. 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

$$\mathcal{L}_{decorr}(t) = \beta \sum_{b=1}^{B} \sum_{i=1}^{M} \sum_{k \neq i} (z_i^b(t) - \bar{z}_i(t))^2 (z_k^b(t) - \bar{z}_k(t))^2 \tag{5}$$

Here, $\beta = \frac{1}{B(M-1)}$ is a scaling factor that keeps the objective invariant to the number of units in the population.

The full learning rule. We compute the weight update as the negative gradient of the LPL objective (Eq. (2)) along with an added weight decay so that learning is less sensitive to initial conditions. For a single network layer this finally yields the network LPL rule:

$$\begin{aligned} \Delta W_{ij} &= -\eta \left(\frac{\partial \mathcal{L}_{\text{pred}}}{\partial W_{ij}} + \lambda_1 \frac{\partial \mathcal{L}_{\text{Hebb}}}{\partial W_{ij}} + \lambda_2 \frac{\partial \mathcal{L}_{\text{decorr}}}{\partial W_{ij}} \right) - \eta \eta_w W_{ij} \\ &= \eta \sum_{b=1}^{B} \left(-(z_i^b - z_i^b(t - \Delta t)) + \lambda_1 \frac{\alpha}{\sigma_i^2} (z_i^b - \bar{z}_i) - \lambda_2 \beta (z_i^b - \bar{z}_i) \sum_{k \neq i} (z_k^b - \bar{z}_k)^2 \right) f'(a_i^b) x_j^b \\ &\quad - \eta \eta_w W_{ij} \end{aligned} \tag{6}$$

where λ_1 and λ_2 are parameters which control the relative strengths of each objective, and η_w is a parameter controlling the strength of the weight decay. Note, we omitted the time argument t from all quantities corresponding to the present for ease of notation.

Numerical optimization methods. We implemented all network models and LPL learning as gradient descent on the equivalent objective function using PyTorch [78] with the Lightning framework [79]. 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 GPUs. In case of the DNNs, we used the Adam optimizer [80] to accelerate learning. Parameter values used in all simulations are summarized in Table 2.

	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	200	200	1024
Training steps	$\max(10000, 100\sigma_y)$	1000	∼78000

[a] With default parameters from Kingma et al. [80].
[b] Reduced to zero during training using a cosine learning rate schedule.
[c] ∼35000 for CIFAR-10.

Table 2: Hyperparameter values

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:

$$\mathcal{L} = (z(t) - SG(z(t - \Delta t)))^2 - \log\left(\sigma_z^2(t) + \epsilon\right) - \eta_w \sum_j W_j^2 \tag{7}$$

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. (6)) simplifies to Eq. (1) without the decorrelation term.

Synthetic two-dimensional dataset generation. The two-dimensional synthetic data sequence (Fig. 3a) consists of two clusters of inputs, one centered at $x = 1$, and the other at $x = -1$, and $B = 2$ is the distance separating the two clusters along the x-axis. 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 $\sigma_x \ll B$ fixed ($\sigma_x = 0.1$) and varied σ_y. Additionally, to account for occasional transitions between clusters with a probability p, we included a corresponding fraction of such "noisy 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 receives the x and y components of the data as its two inputs, and learns according to LPL. In our simulations in Fig. 3 we assumed $p \to 0$, however, the qualitative behavior remains unchanged for higher noise levels of up to at least $p = 0.2$, meaning one-fifth of the samples in every training batch were "noisy" pairs of points from different clusters (Supplementary Fig. S5), whereas x-Selectivity gracefully declines for higher values of p and σ_y.

Synthetic handwritten digit sequences. The images making up the high-dimensional visual digit sequences (Fig. 3d) were generated using a subset of the MNIST handwritten digit dataset [81]. 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 space at either the top-left or bottom-right location. To simulate the two kinds of temporal transitions in the Digit and Location sequences, we changed the pairs of stimuli that were given as paired consecutive inputs to the neuron.

Neuronal selectivity measure. After training until convergence, we measured the neuron's selectivity to a relevant feature as the normalized difference between mean responses to inputs from two clusters that differ in that feature. For instance, in the two-dimensional data sequence, if $\langle z_1 \rangle$ is the mean output for inputs from the cluster at $x = 1$, $\langle z_2 \rangle$ is the mean output for inputs from the cluster at $x = -1$, and $z_{\max}$ and $z_{\min}$ are the maximum and minimum responses across all inputs, then the selectivity χ is defined as:

$$\chi = \frac{|\langle z_1 \rangle - \langle z_2 \rangle|}{z_{\max} - z_{\min}} \tag{8}$$

Similarly, 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.

Learning in deep convolutional neural networks

For all network simulations, we used a convolutional DNN based on the VGG-11 architecture [82] (see Supplementary Material S4 for details). We trained this network on CIFAR-10 [83] and STL-10 [84], two natural image datasets (see Table 2 for hyperparameters). To simulate related consecutive inputs, we used two differently augmented versions of the same underlying image, an approach typically followed in vision-based SSL methods. Specifically, we used the set of augmentations originally suggested in [16], which includes random crops, blurring, color jitter and random horizontal flips (see Supplementary Fig. S3 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 [80] 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 $\mathcal{L}_{\text{LPL}}^{(out)}$ at the network's output using backpropagation to train the hidden layer weights. This is the standard approach used in deep learning. In contrast, in layer-local learning we minimize the LPL objective, i.e, $\mathcal{L}_{\text{LPL}}$ at each layer in the network independently without backpropagating loss gradients between layers similar to previous work [24, 46]. 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 $\sum_l \mathcal{L}_{\text{LPL}}^{(l)}$.

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 learning. Although pooling was not strictly necessary and LPL could be directly applied on the feature maps, pooling substantially sped up training and led to an overall improved linear readout accuracy on CIFAR-10 (Supplementary Table S1). On the STL-10 dataset, however, GAP was essential in most training runs for achieving linear readout accuracy levels above the pixel-level baseline (cf. Table 1). This discrepancy may be due to the larger pixel dimensions of the datasets and the resulting possibility for more drastic augmentations, for instance, in the case of random cropping.

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 readout neurons' outputs in response to a set of training images. Crucially, in 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. 4b) on a held-out test set of images. Furthermore, we used the same procedure to evaluate the representations at intermediate layers (Fig. 4c).

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 [85]. Concretely, if $Z \in \mathcal{R}^{M \times N}$ are N-dimensional representations of M 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:

$$\text{Dim.} = \frac{\left(\sum_{i=1}^{N} \lambda_i\right)^2}{\sum_{i=1}^{N} \lambda_i^2} \tag{9}$$

Model of unsupervised learning in inferotemporal cortex

Network model and pretraining dataset. To simulate the experimental setup of [20], we modeled the animal's ventral 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 then proceeded as follows. First, we performed unsupervised pre-training for 800 epochs on STL-10 using augmented image views as described above. Second, 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 further expanded to to account for the added spatial dimension of different canvas positions in the experiment Li et al. [20]. 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 Material 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.

Experimental data for comparison. To compare to experiments, we replotted experimental data from Li et al. [20] in Fig. 5b. To that end, we extracted the data from the published plots using the Engauge Digitizer software [86].

Temporal data generation. To simulate the experiment by [20], 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. Like 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 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 of the cosine learning rate schedule using 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 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 how the object selectivity $P - N$ changes 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.

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 work was supported by the Novartis Research Foundation.

Competing interests

The authors declare no competing interests.

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] Yamins, D. L. K. and DiCarlo, J. J. "Using goal-driven deep learning models to understand sensory cortex". en. In: *Nature Neuroscience* 19.3 (Mar. 2016), pp. 356–365. ISSN: 1097-6256. DOI: 10.1038/nn.4244.

[9] Kriegeskorte, N. "Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing". In: *Annual Review of Vision Science* 1.1 (2015). _eprint: https://doi.org/10.1146/annurev-vision-082114-035447, pp. 417–446. DOI: 10.1146/annurev-vision-082114-035447.

[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] Grossberg, S. "Competitive Learning: From Interactive Activation to Adaptive Resonance". en. In: *Cognitive Science* 11.1 (Jan. 1987), pp. 23–63. ISSN: 1551-6709. DOI: 10.1111/j.1551-6708.1987.tb00862.x.

[13] Crick, F. "The recent excitement about neural networks". en. In: *Nature* 337.6203 (Jan. 1989), pp. 129–132. ISSN: 0028-0836. DOI: 10.1038/337129a0.

[14] 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.

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

[16] 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.

[17] Tian, Y., Chen, X., and Ganguli, S. "Understanding self-supervised Learning Dynamics without Contrastive Pairs". In: *arXiv:2102.06810 [cs]* (Feb. 2021). arXiv: 2102.06810.

[18] Tomasev, N., Bica, I., McWilliams, B., Buesing, L., Pascanu, R., Blundell, C., and Mitrovic, J. "Pushing the limits of self-supervised ResNets: Can we outperform supervised learning without labels on ImageNet?" In: *arXiv preprint arXiv:2201.05119* (2022).

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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).

[25] 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.

[26] 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.

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

[28] 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.

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

[30] Bliss, T. V. P. and Lømo, T. "Long-Lasting Potentiation of Synaptic Transmission in the Dentate Area of the Anaesthetized Rabbit Following Stimulation of the Perforant Path". en. In: *The Journal of Physiology* 232.2 (Jan. 1973), pp. 331–356. ISSN: 0022-3751, 1469-7793.

[31] 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.

[32] 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.

[33] 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.

[34] Suvrathan, A. "Beyond STDP—towards diverse and functionally relevant plasticity rules". In: *Current Opinion in Neurobiology.* Neurobiology of Learning and Plasticity 54 (Feb. 2019), pp. 12–19. ISSN: 0959-4388. DOI: 10.1016/j.conb.2018.06.011.

[35] 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.

[36] 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.

[37] Cooper, L. N., Intrator, N., Blais, B. S., and Shouval, H. Z. *Theory of Cortical Plasticity.* New Jersey: World Scientific, Apr. 2004. ISBN: 978-981-238-746-2.

[38] 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.

[39] Rajan, K., Abbott, L. F., and Sompolinsky, H. "Stimulus-dependent suppression of chaos in recurrent neural networks". In: *Physical Review E* 82.1 (July 2010), pp. 1–5. ISSN: 1539-3755. DOI: 10.1103/PhysRevE.82.011903.

[40] 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.

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

[42] 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.

[43] 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.

[44] Ngiam, J., Chen, Z., Bhaskar, S. A., Koh, P. W., and Ng, A. Y. "Sparse Filtering". In: *Advances in Neural Information Processing Systems 24*. Ed. by Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q. Curran Associates, Inc., 2011, pp. 1125–1133.

[45] Illing, B., Gerstner, W., and Brea, J. "Biologically plausible deep learning - But how far can we go with shallow networks?" eng. In: *Neural Networks: The Official Journal of the International Neural Network Society* 118 (Oct. 2019), pp. 90–101. ISSN: 1879-2782. DOI: 10.1016/j.neunet.2019.06.001.

[46] 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.

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

[48] Fukushima, K. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". en. In: *Biological Cybernetics* 36.4 (Apr. 1980), pp. 193–202. ISSN: 0340-1200, 1432-0770. DOI: 10.1007/BF00344251.

[49] Jing, L., Vincent, P., LeCun, Y., and Tian, Y. "Understanding Dimensional Collapse in Contrastive Self-supervised Learning". In: (2021), pp. 1–15. arXiv: 2110.09348.

[50] Hua, T., Wang, W., Xue, Z., Ren, S., Wang, Y., and Zhao, H. "On Feature Decorrelation in Self-Supervised Learning". In: (2021). arXiv: 2105.00470.

[51] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. "Learning representations by back-propagating errors". en. In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 0028-0836. DOI: 10.1038/323533a0.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] Yeung, L. C., Blais, B. S., Cooper, L. N., and Shouval, H. Z. "Calcium as the associative signal for a model of Hebbian plasticity: application to multi-input environments". In: *Neurocomputing*. Computational Neuroscience: Trends in Research 2003 52–54 (June 2003), pp. 437–440. ISSN: 0925-2312. DOI: 10.1016/S0925-2312(02)00866-4.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] 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.

[64] 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.

[65] Jedlicka, P., Benuskova, L., and Abraham, W. C. "A Voltage-Based STDP Rule Combined with Fast BCM-Like Metaplasticity Accounts for LTP and Concurrent "Heterosynaptic" LTD in the Dentate Gyrus In Vivo". In: *PLoS Comput Biol* 11.11 (Nov. 2015), e1004588. DOI: 10.1371/journal.pcbi.1004588.

[66] Clopath, C., Vogels, T. P., Froemke, R. C., and Sprekeler, H. *Receptive field formation by interacting excitatory and inhibitory synaptic plasticity.* en. preprint. Neuroscience, July 2016. DOI: 10.1101/066589.

[67] Letzkus, J. J., Wolff, S. B. E., and Lüthi, A. "Disinhibition, a Circuit Mechanism for Associative Learning and Memory". English. In: *Neuron* 88.2 (Oct. 2015), pp. 264–276. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2015.09.024.

[68] 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.

[69] 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.

[70] Le Meur, O. and Liu, Z. "Saccadic model of eye movements for free-viewing condition". en. In: *Vision Research*. Computational Models of Visual Attention 116 (Nov. 2015), pp. 152–164. ISSN: 0042-6989. DOI: 10.1016/j.visres.2014.12.026.

[71] 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.

[72] 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.

[73] 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.

[74] González-Rueda, A., Pedrosa, V., Feord, R. C., Clopath, C., and Paulsen, O. "Activity-Dependent Downscaling of Subthreshold Synaptic Inputs during Slow-Wave-Sleep-like Activity In Vivo". English. In: *Neuron* 0.0 (Mar. 2018). ISSN: 0896-6273. DOI: 10.1016/j.neuron.2018.01.047.

[75] Pawlak, V., Greenberg, D. S., Sprekeler, H., Gerstner, W., and Kerr, J. N. "Changing the responses of cortical neurons from sub- to suprathreshold using single spikes in vivo". In: *eLife* 2 (2013), e00012. ISSN: 2050-084X. DOI: 10.7554/eLife.00012.

[76] Inglebert, Y., Aljadeff, J., Brunel, N., and Debanne, D. "Synaptic plasticity rules with physiological calcium levels". en. In: *Proceedings of the National Academy of Sciences* (Dec. 2020). Publisher: National Academy of Sciences Section: Biological Sciences. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.2013663117.

[77] Zylberberg, J., Murphy, J. T., and DeWeese, M. R. "A Sparse Coding Model with Synaptically Local Plasticity and Spiking Neurons Can Account for the Diverse Shapes of V1 Simple Cell Receptive Fields". In: *PLoS Comput Biol* 7.10 (Oct. 2011), e1002250. DOI: 10.1371/journal.pcbi.1002250.

[78] 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.

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

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

[81] Deng, L. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

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

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

[84] 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.

[85] 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.

[86] 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).

Supplementary Material

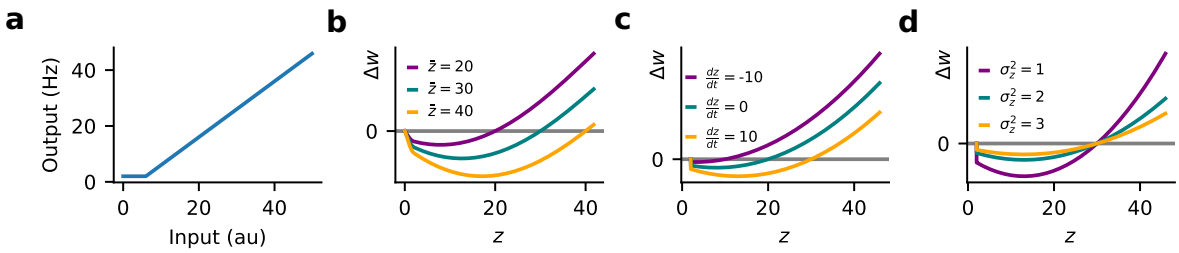

Figure S1: **Same as Figure 2 but for the rectified linear unit (ReLU) activation function used for the DNN simulations.** **(a)** The ReLU activation function shifted to approximately fit typical neuronal transfer functions. **(b)** Weight change induced by the LPL rule for co-varying input x and the postsynaptic activity z for different values of the mean-dependent threshold $\bar{z}$. **(c)** Same as (b) but for different values of the variance of the postsynaptic activity. **(d)** Same as (b) but for different rates of change $^{dz}/_{dt}$ of the postsynaptic activity of the learning rule.

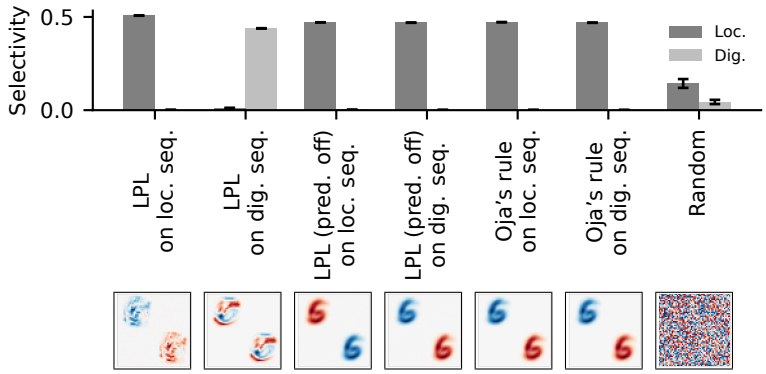

Figure S2: **Same as Figure 3e but with additional controls.** Emergent feature selectivity of a single neuron exposed to the two sequence modalities while learning under different rules (top), and the resulting input weights learned in each case (bottom). Under LPL, the neuron's selectivity mirrors the temporally preserved (predictive) property in each sequence, i.e., location in the Location Sequence and digit identity in the Digit Sequence. With the predictive term off, the sequence ceases to matter and the neuron always becomes selective to location, the direction of maximum variance, which is the same as for Oja's rule.

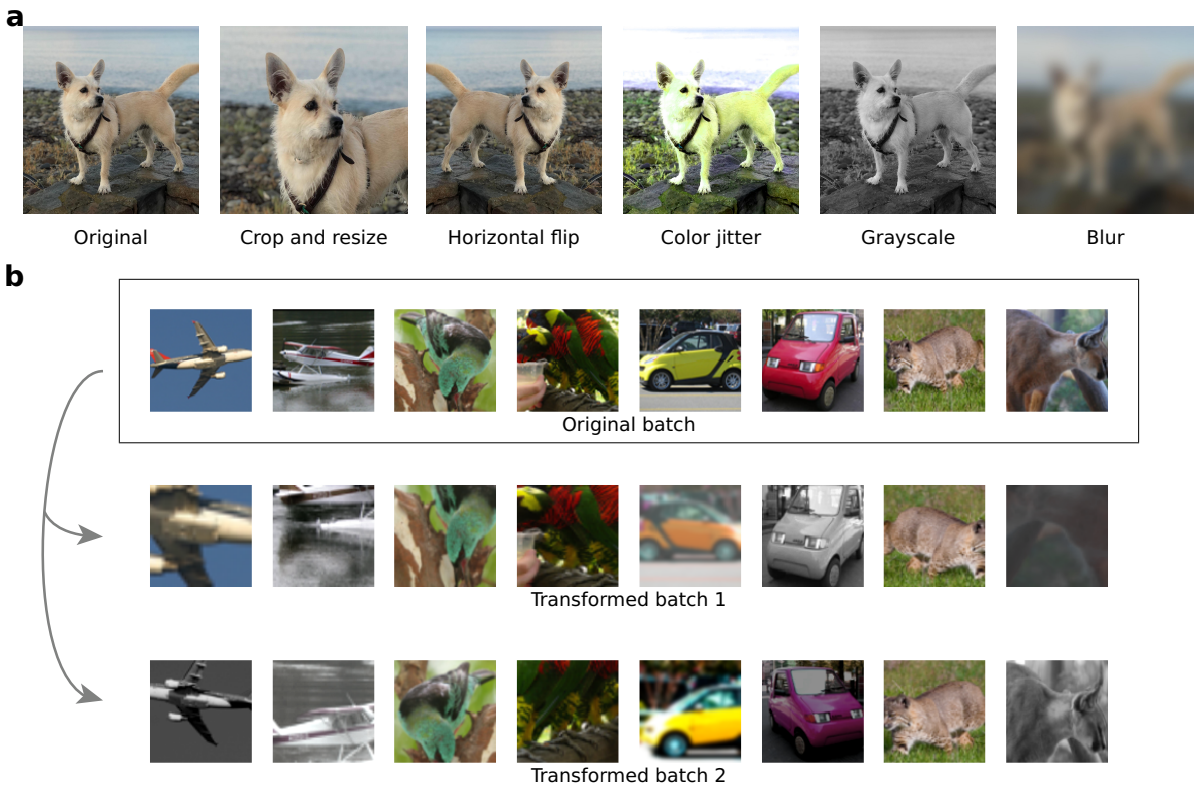

Figure S3: **Image augmentation model. (a)** Illustration of the image transformations used to generate natural image sequences as suggested by Chen et al. [16]. **(b)** Sample images from STL-10 and their transformed versions used for training the DNNs.

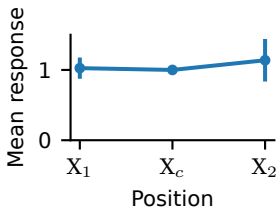

Figure S4: **Learned network representations are invariant to object position on the canvas.** Activity of neurons in the pretrained DNN's output layer in response to images at three positions on the canvas, normalized by each neuron's response to the center position, and averaged over neurons and over images.

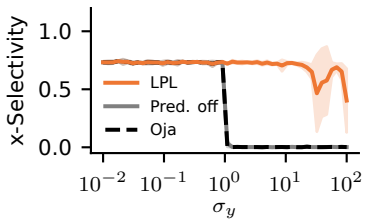

Figure S5: **Same as Figure 3b but for high rates of noisy transitions between clusters in the training data with** $p = 0.2$ (Methods). Cluster selectivity of the features learned by LPL, without the predictive term, and by Oja's rule for different values of σ_y.

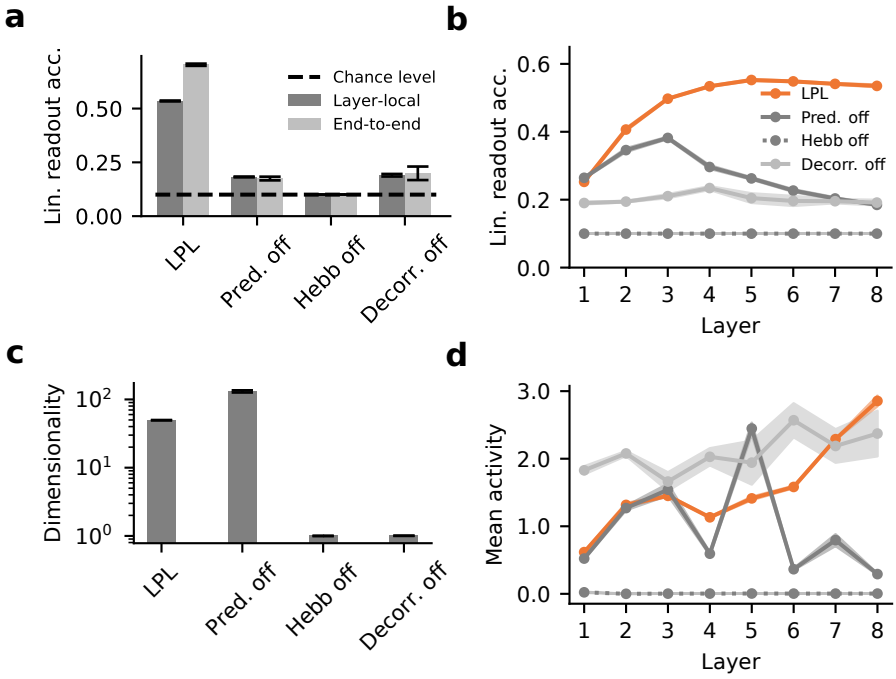

Figure S6: **Same as Figure 4 but for the CIFAR-10 dataset.** **(b)** Linear readout accuracy of object categories decoded from representations at the network output after training it on natural image data for different learning rules in layer-local (dark) as well as the end-to-end configuration (light). **(c)** Linear readout accuracy of the internal representations at different layers of the DNN after training. **(d)** Dimensionality of the internal representations for the different learning rule configurations shown in (b). **(e)** Mean neuronal activity at different layers of the DNN after training for the different learning rule variants shown in (c).

	STL-10	CIFAR-10
VGG-11	61.9±0.2	53.5±0.5
VGG-11 without GAP in the loss computation	23.8±11.8	48.0±0.7
VGG-11 without bias terms	53.3	48.1

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 two modified versions. VGG-11 without GAP refers to computing and optimizing the LPL objective on the unpooled feature maps. VGG-11 without bias terms refers to the same architecture including GAP but without any bias terms in the convolutional layers.

S1 Equivalence of the objective function and learning rule formulations

Here, we show that the objective functions defined in Eqs. (3), (4), and (5) indeed result in the LPL rule (Eq. (6)).

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

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

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

$$\frac{\partial \mathcal{L}_{pred}}{\partial W_{ij}} = \sum_{b=1}^{B}\Big(z_i^b - z_i^b(t-\Delta t)\Big)\, f'(a_i^b)x_j^b \tag{10}$$

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:

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

is obtained as the negative gradient of this loss function with respect to the weights W. The gradient itself is given by:

$$\frac{\partial \mathcal{L}_{Hebb}}{\partial W_{ij}} = -\frac{\alpha}{\sigma_i^2}\sum_{b=1}^{B}(z_i^b - \bar{z}_i)f'(a_i^b)x_j^b \tag{11}$$

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 a batch estimate.

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

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.

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

which gives the gradient:

$$\frac{\partial \mathcal{L}_{\text{decorr}}}{\partial W_{ij}} = \beta\sum_{b=1}^{B}(z_i^b - \bar{z}_i)f'(a_i^b)x_j^b\sum_{k\neq i}(z_k^b - \bar{z}_k)^2 \tag{12}$$

The full learning rule. The combined weight updates for a descent along the sum of the three gradients in Eqs. (10), (11), and (12) in a single-layer network finally yields the LPL rule including the decorrelation component:

$$\begin{aligned}\Delta W_{ij} &= -\eta\left(\frac{\partial\mathcal{L}_{\text{pred}}}{\partial W_{ij}} + \lambda_1\frac{\partial\mathcal{L}_{\text{Hebb}}}{\partial W_{ij}} + \lambda_2\frac{\partial\mathcal{L}_{\text{decorr}}}{\partial W_{ij}}\right) \\ &= \eta\sum_{b=1}^{B}\left(-(z_i^b - z_i^b(t-\Delta t)) + \lambda_1\frac{\alpha}{\sigma_i^2}(z_i^b - \bar{z}_i) - \lambda_2\beta(z_i^b - \bar{z}_i)\sum_{k\neq i}(z_k^b - \bar{z}_k)^2\right) f'(a_i^b)x_j^b\end{aligned} \tag{13}$$

Including weight decay in the update finally yields the LPL rule for a network given in Eq. (6).

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 [28], 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. (11):

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

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 due to multiplication by $1/\langle z^2\rangle$.

$$\begin{aligned}\Delta W_j(t) &= \frac{z(t)x_j(t)}{\langle z^2\rangle} - \eta_w W_j \\ \langle\Delta W_j\rangle &= \frac{\langle zx_j\rangle}{\langle z^2\rangle} - \eta_w W_j \\ &= \frac{1}{\langle z^2\rangle}\left(\langle zx_j\rangle - \eta_w W_j\left\langle z^2\right\rangle\right)\end{aligned} \tag{14}$$

Oja's rule is presented below for reference

$$\begin{aligned}\Delta W_j^{Oja}(t) &= z(t)x_j(t) - W_j z^2 \\ \left\langle\Delta W_j^{Oja}\right\rangle &= \langle zx_j\rangle - W_j\left\langle z^2\right\rangle\end{aligned} \tag{15}$$

S3 Importance of the variance-dependent suppression of Hebbian learning

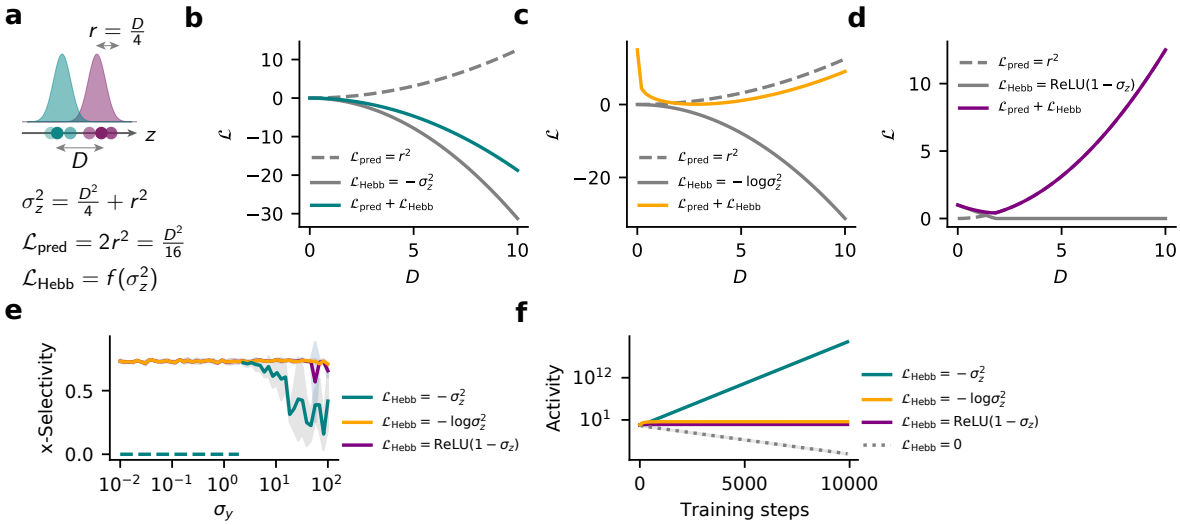

Figure S7: **Suppression of Hebbian learning at high variance is crucial for stable learning.** **(a)** Example setting with two clusters distance D apart in representation space. The size of each cluster $r = \frac{D}{4}$ is assumed to scale with D. In this case, representational variance is $\frac{D^2}{4} + r^2$. **(b)** Hebbian and predictive losses as a function of cluster separation D for $\mathcal{L}_{\text{Hebb}} = -\sigma_z^2$. The loss function is dominated by the Hebbian term at large D, and drives runaway LTP. **(c)** Same as (b) but for $\mathcal{L}_{\text{Hebb}} = -\log\sigma_z^2$, the Hebbian objective optimized by LPL. Here, the predictive loss takes over at higher values of D. **(d)** Same as (b) but for $\mathcal{L}_{\text{Hebb}} = -\text{ReLU}(1-\sigma_z)$, the analogous Hebbian objective that was optimized in the VICReg model [27]. Here as well, the predictive loss takes over at higher values of D. **(e)** Cluster selectivity learned by LPL on the two-dimensional synthetic sequence from Fig. 3a with the standard predictive loss combined with the different Hebbian losses from (a), (b) and (c). Dotted sections indicate simulations where learning diverged. **(e)** Mean output activity over training time for LPL on the two-dimensional synthetic sequence with $\sigma_y = 1$ for the different Hebbian losses from (a), (b) and (c).

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. 3a, and studied 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 LPL with different forms of the Hebbian learning objective, each a different decreasing function of the variance, i.e, $\mathcal{L}_{\text{Hebb}} = f(\sigma_z^2)$. Specifically, we considered the cases $\mathcal{L}_{\text{Hebb}} = -\sigma_z^2$, $\mathcal{L}_{\text{Hebb}} = -\log\sigma_z^2$, and $\mathcal{L}_{\text{Hebb}} = \text{ReLU}(1-\sigma_z)$. These objectives are plotted in Fig. S7b-d respectively along with $\mathcal{L}_{\text{pred}}$, and the resulting full

LPL objective in each case. With the naive variance maximization objective $\mathcal{L}_{\text{Hebb}} = -\sigma_z^2$, the full objective is dominated by the Hebbian term at large values of D (Fig. S7b), and therefore inherently drives unstable learning. In contrast, $\mathcal{L}_{\text{Hebb}} = -\log \sigma_z^2$ (Fig. S7c), and $\mathcal{L}_{\text{Hebb}} = \text{ReLU}(1 - \sigma_z)$ (Fig. S7d) favor the predictive objective at higher values of D, and therefore constitute stable LPL objectives with minima at finite values of D.

We validated these scaling arguments with simulations of LPL using each of the three versions of the Hebbian objective on the synthetic two-dimensional sequence learning task from Fig. 3a. We found that using the naive variance maximization objective $\mathcal{L}_{\text{Hebb}} = -\sigma_z^2$ results in poor learning of cluster selectivity, whereas $\mathcal{L}_{\text{Hebb}} = -\log \sigma_z^2$ and $\mathcal{L}_{\text{Hebb}} = \text{ReLU}(1 - \sigma_z)$ remain effective (Fig. S7e). Furthermore, the poor performance of the naive variance objective was indeed due to runaway instability (Fig. S7f).

S4 Details of 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): Identity()
        (2): ReLU(inplace=True)
        (3): 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): Identity()
        (2): ReLU(inplace=True)
        (3): 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): Identity()
        (2): ReLU(inplace=True)
        (3): Identity()
      )
    )
    (3): ConvBlock(
      (module): Sequential(
        (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): Identity()
        (2): ReLU(inplace=True)
        (3): 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): Identity()
```

```
        (2): ReLU(inplace=True)
        (3): Identity()
      )
    )
    (5): ConvBlock(
      (module): Sequential(
        (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): Identity()
        (2): ReLU(inplace=True)
        (3): 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): Identity()
        (2): ReLU(inplace=True)
        (3): Identity()
      )
    )
    (7): ConvBlock(
      (module): Sequential(
        (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): Identity()
        (2): ReLU(inplace=True)
        (3): 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.