

## Contrastive Similarity Matching for Supervised Learning

**Shanshan Qin**

*ssqin@g.harvard.edu*

*John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, U.S.A.*

**Nayantara Mudur**

*nmudur@g.harvard.edu*

*Department of Physics, Harvard University, Cambridge, MA 02138, U.S.A.*

**Cengiz Pehlevan**

*cpehlevan@seas.harvard.edu*

*John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, U.S.A.*

**We propose a novel biologically plausible solution to the credit assignment problem motivated by observations in the ventral visual pathway and trained deep neural networks. In both, representations of objects in the same category become progressively more similar, while objects belonging to different categories become less similar. We use this observation to motivate a layer-specific learning goal in a deep network: each layer aims to learn a representational similarity matrix that interpolates between previous and later layers. We formulate this idea using a contrastive similarity matching objective function and derive from it deep neural networks with feedforward, lateral, and feedback connections and neurons that exhibit biologically plausible Hebbian and anti-Hebbian plasticity. Contrastive similarity matching can be interpreted as an energy-based learning algorithm, but with significant differences from others in how a contrastive function is constructed.**

### 1 Introduction ---

Synaptic plasticity is generally accepted as the underlying mechanism of learning in the brain, which almost always involves a large population of neurons and synapses across many different brain regions. How the brain modifies and coordinates individual synapses in the face of limited information available to each synapse in order to achieve a global learning task, the credit assignment problem, has puzzled scientists for decades. A major effort in this domain has been to look for a biologically plausible implementation of the backpropagation of error algorithm (BP) (Rumelhart, Hinton,

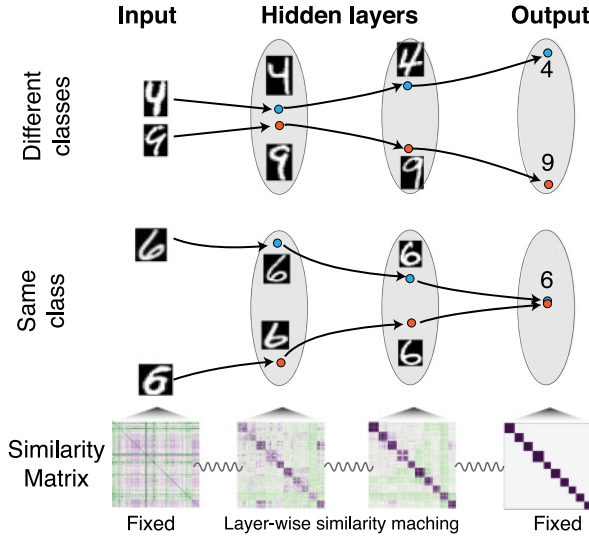


Figure 1: Supervised learning via layer-wise similarity matching. For inputs of different categories, similarity matching differentiates the representations progressively (top), while for objects of the same category, representations become more and more similar (middle). For a given set of training data and their corresponding labels, the training process can be regarded as learning hidden representations whose similarity matrices match that of both input and output (bottom). The tuning of representational similarity is indicated by the springs with the constraints that input and output similarity matrices are fixed.

& Williams, 1986), which has long been disputed due to its biological implausibility (Crick, 1989), although recent studies have made progress in resolving some of these concerns (Xie & Seung, 2003; Lee, Zhang, Fischer, & Bengio, 2015; Lillicrap, Cownden, Tweed, & Akerman, 2016; Nøkland, 2016; Scellier & Bengio, 2017; Guerguiev, Lillicrap, & Richards, 2017; Whittington & Bogacz, 2017, 2019; Sacramento, Costa, Bengio, & Senn, 2018; Richards & Lillicrap, 2019; Belilovsky, Eickenberg, & Oyallon, 2018; Ororbia & Mali, 2019; Lillicrap, Santoro, Marris, Akerman, & Hinton, 2020).

In this letter, we present a novel approach to the credit assignment problem, motivated by observations on the nature of hidden-layer representations in the ventral visual pathway of the brain and deep neural networks. In both, representations of objects belonging to different categories become less similar, while representations of objects belonging to the same category become more similar (Grill-Spector & Weiner, 2014; Kriegeskorte et al., 2008; Yamins & DiCarlo, 2016). In other words, categorical clustering of representations becomes more and more explicit in the later layers (see Figure 1). These results suggest a new approach to the credit assignment problem.

By assigning each layer a layer-local similarity matching task (Pehlevan & Chklovskii, 2019; Obeid, Ramambason, & Pehlevan, 2019), whose goal is to learn an intermediate representational similarity matrix between previous and later layers, we may be able to get away from the need of backward propagation of errors (see Figure 1). Motivated by this idea and previous observations that error signal can be implicitly propagated via the change of neural activities (Hinton & McClelland, 1988; Scellier & Bengio, 2017), we propose a biologically plausible supervised learning algorithm, the contrastive similarity matching (CSM) algorithm.

Contrastive training (Anderson & Peterson, 1987; Movellan, 1991; Baldi & Pineda, 1991) has been used to learn the energy landscapes of neural networks (NNs) whose dynamics minimize an energy function. Examples include influential algorithms like the contrastive Hebbian learning (CHL) (Movellan, 1991) and equilibrium propagation (EP) (Scellier & Bengio, 2017), where weight updates rely on the difference of the neural activity between a free phase and a clamped (CHL) or nudged (EP) phase to locally approximate the gradient of an error signal. The learning process can be interpreted as minimizing a contrastive function, which reshapes the energy landscape to eliminate spurious fixed points and makes the desired fixed point more stable.

The CSM algorithm applies this idea to a contrastive function formulated by nudging the output neurons of a multilayer similarity matching objective function (Obeid et al., 2019). As a consequence, the hidden layers learn intermediate representations between their previous and later layers. From the CSM contrastive function, we derive deep neural networks with feedforward, lateral, and feedback connections and neurons that exhibit biologically plausible Hebbian and anti-Hebbian plasticity.

The nudged phase of the CSM algorithm is analogous to the nudged phase of EP but different. It performs Hebbian feedforward and anti-Hebbian lateral updates. CSM has the opposite sign for the lateral connection updates compared with EP and CHL. This is because our weight updates solve a minimax problem. Anti-Hebbian learning pushes neurons within a layer to learn different representations. The free phase of CSM is also different, where only feedforward weights are updated by an anti-Hebbian rule. In EP and CHL, all weights are updated.

Our main contributions and results follow:

- We provide a novel approach to the credit assignment problem using biologically plausible learning rules by generalizing the similarity matching principle (Pehlevan & Chklovskii, 2019) to supervised learning tasks and introducing the CSM algorithm.
- The proposed supervised learning algorithm can be related to other energy-based algorithms, but with a distinct underlying mechanism.
- We present a version of our neural network algorithm with structured connectivity.

- We show that the performance of our algorithm is on par with other energy-based algorithms using numerical simulations. The learned representations of our Hebbian/anti-Hebbian network are sparser.

The rest of this letter is organized as follows. In section 2, to illustrate our main ideas, we introduce and discuss supervised similarity matching. We then introduce the nudged deep similarity matching objective, from which we derive the CSM algorithm for deep neural networks with nonlinear activation functions and structured connectivity. We discuss the relation of CSM to other energy-based learning algorithms. In section 3, we report the performance of CSM and compare it with EP, highlighting the differences between them. Finally, we discuss our results, possible biological mechanisms, and relate them to other work in section 4.

## 2 Contrastive Similarity Matching for Deep Nonlinear Networks

**2.1 Warm-Up: Supervised Similarity Matching Objective.** Here we illustrate our main idea in a simple setting. Let  $\mathbf{x}_t \in \mathbb{R}^n, t = 1, \dots, T$  be a set of data points and  $\mathbf{z}^l_t \in \mathbb{R}^k$  be their corresponding desired output or labels. Our idea is that the representation learned by the hidden layer,  $\mathbf{y}_t \in \mathbb{R}^m$ , should be halfway between the input  $\mathbf{x}$  and the desired output  $\mathbf{z}^l$ . We formulate this idea using representational similarities, quantified by the dot product of representational vectors within a layer. Our proposal can be formulated as the following optimization problem, which we name *supervised similarity matching*:

$$\min_{\{\mathbf{y}_t\}_{t=1}^T} \frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T [(\mathbf{x}_t^\top \mathbf{x}_{t'} - \mathbf{y}_t^\top \mathbf{y}_{t'})^2 + (\mathbf{y}_t^\top \mathbf{y}_{t'} - \mathbf{z}_t^{l\top} \mathbf{z}_{t'}^l)^2]. \tag{2.1}$$

To get an intuition about what this cost function achieves, consider the case where only one training datum exists. Then,  $\mathbf{y}_1^\top \mathbf{y}_1 = \frac{1}{2}(\mathbf{x}_1^\top \mathbf{x}_1 + \mathbf{z}_1^{l\top} \mathbf{z}_1^l)$ , satisfying our condition. When multiple training data are involved, interactions between different data points lead to a nontrivial solution, but the fact that the hidden-layer representations are in between the input and output layers stays.

The optimization problem, equation 2.1, can be analytically solved, making our intuition precise. Let the representational similarity matrix of the input layer be  $R_{tt'}^x \equiv \mathbf{x}_t^\top \mathbf{x}_{t'}$ , the hidden layer be  $R_{tt'}^y \equiv \mathbf{y}_t^\top \mathbf{y}_{t'}$ , and the output layer be  $R_{tt'}^z \equiv \mathbf{z}_t^{l\top} \mathbf{z}_{t'}^l$ . Instead of solving  $\mathbf{y}$  directly, we can reformulate and solve the supervised similarity matching problem, equation 2.1, for  $\mathbf{R}^y$ , and then obtain  $\mathbf{y}$ s by a matrix factorization through an eigenvalue decomposition. By completing the square, problem 2.1 becomes an optimization problem for  $\mathbf{R}^y$ :

$$\min_{\mathbf{R}^y \in S^m} \frac{1}{T^2} \left\| \frac{1}{2} (\mathbf{R}^x + \mathbf{R}^z) - \mathbf{R}^y \right\|_F^2, \quad (2.2)$$

where  $S^m$  is the set of symmetric matrices with rank  $m$  and  $F$  denotes the Frobenius norm. Optimal  $\mathbf{R}^y$  is given by keeping the top  $m$  modes in the eigenvalue decomposition of  $\frac{1}{2} (\mathbf{R}^x + \mathbf{R}^z)$  and setting the rest to zero. If  $m \geq \text{rank}(\mathbf{R}^x + \mathbf{R}^z)$ , then optimal  $\mathbf{R}^y$  exactly equals  $\frac{1}{2} (\mathbf{R}^x + \mathbf{R}^z)$ , achieving a representational similarity matrix that is the average of input and output layers.

The supervised similarity matching problem, equation 2.1, can be solved by an online algorithm that can be mapped onto the operation of a biologically plausible network with a single hidden layer, which runs an attractor dynamics minimizing an energy function (see appendix A for details). This approach can be generalized to multilayer and nonlinear networks. We do not pursue it further because the resulting algorithm does not perform as well due to spurious fixed points of nonlinear dynamics for a given input  $\mathbf{x}_t$ . The CSM algorithm overcomes this problem.

**2.2 Nudged Deep Similarity Matching Objective and Its Dual Formulation.** Our goal is to combine the ideas of supervised similarity matching and contrastive learning to derive a biologically plausible supervised learning algorithm. To do so, we define the nudged similarity matching problem first.

In energy-based learning algorithms like CHL and EP, weight updates rely on the difference of neural activity between a free phase and a clamped or nudged phase to locally approximate the gradient of an error signal. This process can be interpreted as minimizing a contrastive function, which reshapes the energy landscape to eliminate the spurious fixed points and make the fixed point corresponding to the desired output more stable. We adopt this idea to introduce what we call the nudged similarity matching cost function and derive its dual formulation, which will be the energy function used in our contrastive formulation.

We consider a  $P$ -layer ( $P - 1$  hidden layers) NN with nonlinear activation functions,  $f$ . For notational convenience, we denote inputs to the network by  $\mathbf{r}^{(0)}$ , outputs by  $\mathbf{r}^{(P)}$ , and activities of hidden layers by  $\mathbf{r}^{(p)}$ ,  $p = 1, \dots, P - 1$ . We propose the following objective function for the training phase where outputs are nudged toward the desired labels  $\mathbf{z}_t^l$ :

$$\begin{aligned} \min_{\substack{a_1 \leq r_t^p \leq a_2 \\ t=1, \dots, T \\ p=1, \dots, P}} & \sum_{p=1}^P \frac{\gamma^{p-P}}{2T^2} \sum_{t=1}^T \sum_{t'=1}^T \|\mathbf{r}_t^{(p-1)\top} \mathbf{r}_{t'}^{(p-1)} - \mathbf{r}_t^{(p)\top} \mathbf{r}_{t'}^{(p)}\|_2^2 \\ & + \sum_{p=1}^P \frac{2\gamma^{p-P}}{T} \sum_{t=1}^T \mathbf{F}(\mathbf{r}_t^{(p)})^\top \mathbf{1} + \frac{2\beta}{T} \sum_{t=1}^T \|\mathbf{r}_t^{(P)} - \mathbf{z}_t^l\|_2^2. \end{aligned} \quad (2.3)$$

Here,  $\beta$  is a control parameter that specifies how strong the nudge is.  $\beta \rightarrow \infty$  limit corresponds to clamping the output layer to the desired output.  $\gamma \geq 0$  is a parameter that controls the influence of the later layers to the previous layers.  $F(\mathbf{r}_t^{(p)})$  is a regularizer defined and related to the activation function by  $dF(\mathbf{r}_t^{(p)})/d\mathbf{r}_t^{(p)} = \mathbf{u}_t^{(p)} - \mathbf{b}_t^{(p)}$ , where  $\mathbf{r}_t^{(p)} = f(\mathbf{u}_t^{(p)})$ ,  $\mathbf{u}_t^{(p)}$ , and  $\mathbf{r}_t^{(p)}$  are the total input and output of the  $p$ th layer, respectively, and  $\mathbf{b}_t^{(p)}$  is the the threshold of neurons in layer  $p$ . The reason for the inclusion of this regularizer will be apparent below. We assume  $f$  to be a monotonic and bounded function, whose bounds are given by  $a_1$  and  $a_2$ .

The objective function, equation 2.3, is almost identical to the deep similarity matching objective introduced in Obeid et al. (2019), except the nudging term. They used the  $\beta = 0$  version as an unsupervised algorithm. Here, we use a nonzero  $\beta$  for supervised learning.

We note that we have not made a reference to a particular neural network yet. This is because the neural network that optimizes equation 2.3 will be fully derived from the nudged similarity matching problem. It will not be prescribed as in traditional approaches to deep learning. We next describe how to do this derivation.

Using the duality transforms introduced in Pehlevan, Sengupta, and Chklovskii (2018) and Obeid et al. (2019), the above nudged supervised deep similarity matching problem, equation 2.3, can be turned into a dual minimax problem:

$$\min_{\{\mathbf{W}^{(p)}\}} \max_{\{\mathbf{L}^{(p)}\}} \frac{1}{T} \sum_{t=1}^T l_t \left( \{\mathbf{W}^{(p)}\}, \{\mathbf{L}^{(p)}\}, \mathbf{r}_t^{(0)}, \mathbf{z}_t^l, \beta \right), \tag{2.4}$$

where

$$\begin{aligned} l_t := & \min_{\substack{a_1 \leq \mathbf{r}_t^{(p)} \leq a_2 \\ p=1, \dots, P}} \sum_{p=1}^P \gamma^{p-P} \left[ \text{Tr} \mathbf{W}^{(p)\top} \mathbf{W}^{(p)} - 2\mathbf{r}_t^{(p)\top} \mathbf{W}^{(p)} \mathbf{r}_t^{(p-1)} \right. \\ & \left. + \frac{1 + \gamma(1 - \delta_{pp})}{2} c^{(p)} \left( 2\mathbf{r}_t^{(p)\top} \mathbf{L}^{(p)} \mathbf{r}_t^{(p)} - \text{Tr} \mathbf{L}^{(p)\top} \mathbf{L}^{(p)} \right) + 2\mathbf{F} \left( \mathbf{r}_t^{(p)} \right)^\top \mathbf{1} \right] \\ & + 2\beta \left\| \mathbf{r}_t^{(p)} - \mathbf{z}_t^l \right\|_2^2. \end{aligned} \tag{2.5}$$

Here, we introduced  $c^{(p)}$  as a parameter that governs the relative importance of forward versus recurrent inputs and  $c^{(p)} = 1$  corresponds to the exact transformation (the details are given in appendix B).

In appendix C, we show that the objective of the min in  $l_t$  defines an energy function for a deep neural network with feedforward, lateral, and feedback connections (see Figure 2). It has the following neural dynamics:

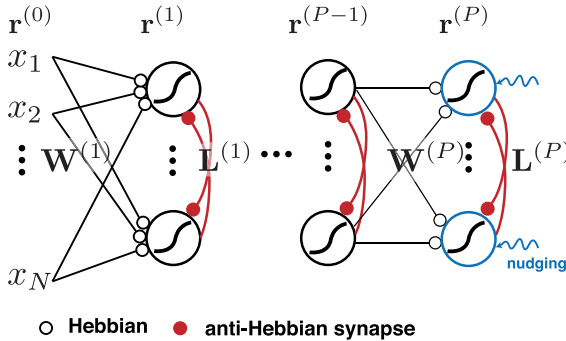


Figure 2: Illustration of the Hebbian/anti-Hebbian network with  $P$  layers that implements the contrastive similarity matching algorithm. The output layer neurons alternate between the free phase and nudged phase.

$$\begin{aligned}
 \tau_p \frac{d\mathbf{u}^{(p)}}{dt} &= -\mathbf{u}^{(p)} + \mathbf{W}^{(p)} \mathbf{r}_t^{(p-1)} - c^{(p)} [1 + \gamma(1 - \delta_{pp})] \mathbf{L}^{(p)} \mathbf{r}_t^{(p)} + \mathbf{b}_t^{(p)} \\
 &\quad + \gamma(1 - \delta_{pp}) \mathbf{W}^{(p+1)\top} \mathbf{r}_t^{(p+1)} - 2\beta \delta_{pp} (\mathbf{r}_t^{(p)} - \mathbf{z}_t^l), \\
 \mathbf{r}_t^{(p)} &= \mathbf{f}(\mathbf{u}^{(p)}),
 \end{aligned} \tag{2.6}$$

where  $\delta_{pp}$  is the Kronecker delta,  $p = 1, \dots, P$ ,  $\tau_p$  is a time constant, and  $\mathbf{W}^{(p+1)} = \mathbf{0}$ ,  $\mathbf{r}_t^{(p+1)} = \mathbf{0}$ . Therefore, the minimization can be performed by running the dynamics until convergence. This observation is the building block of our CSM algorithm, which we present below. Finally, we note that the introduction of the regularizer in equation 2.3 is necessary for the energy interpretation and for proving the convergence of the neural dynamics (Obeid et al., 2019).

**2.3 Contrastive Similarity Matching.** We first state our contrastive function and then discuss its implications. We suppress the dependence on training data in  $l_t$  and define

$$\{\mathbf{L}^{*(p)}\} \equiv \arg \max_{\{\mathbf{L}^{(p)}\}} \frac{1}{T} \sum_{t=1}^T l_t \left( \{\mathbf{W}^{(p)}\}, \{\mathbf{L}^{(p)}\}, \{\mathbf{b}^{(p)}\}, \beta \right) \tag{2.7}$$

and

$$E(\{\mathbf{W}^{(p)}\}, \{\mathbf{b}^{(p)}\}, \beta) = \frac{1}{T} \sum_{t=1}^T l_t \left( \{\mathbf{W}^{(p)}\}, \{\mathbf{L}^{*(p)}\}, \{\mathbf{b}^{(p)}\}, \beta \right). \tag{2.8}$$

Finally, we formulate our contrastive function as

$$J^\beta(\{\mathbf{W}^{(p)}\}, \{\mathbf{b}^{(p)}\}) = E(\beta) - E(0), \tag{2.9}$$

which is to be minimized over feedforward and feedback weights  $\{\mathbf{W}^{(p)}\}$ , as well as bias  $\{\mathbf{b}^{(p)}\}$ . For fixed bias, minimization of the first term,  $E(\beta)$ , corresponds exactly to the optimization of the minimax dual of nudged deep similarity matching, equation 2.4. The second term,  $E(0)$ , corresponds to a free phase, where no nudging is applied. We note that in order to arrive at a contrastive minimization problem, we use the same optimal lateral weights, equation 2.7, from the nudged phase in the free phase. Compared to the minimax dual of nudged deep similarity matching, equation 2.4, we also optimize it over the bias for better performance.

Minimization of the contrastive function, equation 2.9, closes the energy gap between nudged and free phases. Because the energy functions are evaluated at the fixed point of the neural dynamics, equation 2.6, this procedure enforces the output of the nudged network to be a fixed point of the free neural dynamics.

To optimize our contrastive function, equation 2.9, in a stochastic (one training datum at a time) manner, we use the following procedure. For each pair of training data  $\{\mathbf{r}_t^0, \mathbf{z}_t^l\}$ , we run the nudged phase ( $\beta \neq 0$ ) dynamics, equation 2.6, until convergence to get the fixed point  $\mathbf{r}_{\beta,t}^{(p)}$ . Next, we run the free phase ( $\beta = 0$ ) neural dynamics, equation 2.6, until convergence. We collect the fixed points  $\mathbf{r}_{0,t}^{(p)}$ .  $\mathbf{L}^{(p)}$  is updated following a gradient ascent of, equation 2.7, while  $\mathbf{W}^{(p)}$  and  $\mathbf{b}^{(p)}$  follow a gradient descent of equation 2.9:

$$\begin{aligned} \Delta \mathbf{L}^{(p)} &\propto \left( \mathbf{r}_{\beta,t}^{(p)} \mathbf{r}_{\beta,t}^{(p)\top} - \mathbf{L}^{(p)} \right), \\ \Delta \mathbf{W}^{(p)} &\propto \left( \mathbf{r}_{\beta,t}^{(p)} \mathbf{r}_{\beta,t}^{(p-1)\top} - \mathbf{r}_{0,t}^{(p)} \mathbf{r}_{0,t}^{(p-1)\top} \right), \\ \Delta \mathbf{b}^{(p)} &\propto \left( \mathbf{r}_{\beta,t}^{(p)} - \mathbf{r}_{0,t}^{(p)} \right). \end{aligned} \tag{2.10}$$

In practice, learning rates can be chosen differently to achieve the best performance. A constant prefactor before  $\mathbf{L}^{(p)}$  can be added to achieve numerical stability. The CSM algorithm is summarized in algorithm 1.

*2.3.1 Relation to Gradient Descent.* The CSM algorithm can be related to gradient descent in the  $\beta \rightarrow 0$  limit using similar arguments as in Scellier and Bengio (2017). To see this explicitly, we first simplify the notation by collecting all  $\mathbf{W}^{(p)}$  and  $\mathbf{b}^{(p)}$  parameters under one vector variable  $\boldsymbol{\theta}$ , denote all the lateral connection matrices defined in equation 2.7 by  $\mathbf{L}^*$ , and represent the fixed points of the network by  $\bar{\mathbf{r}}$ . Now the energy function can be written as  $E(\boldsymbol{\theta}, \beta; \mathbf{L}^*, \bar{\mathbf{r}})$ , where  $\mathbf{L}^*$  and  $\bar{\mathbf{r}}$  depend on  $\boldsymbol{\theta}$  and  $\beta$  implicitly. In the



**Algorithm 1:** Constrative Similarity Matching (CSM).**Input:** Initial  $\{\mathbf{W}^{(p)}\}$ ,  $\{\mathbf{L}^{(p)}\}$ ,  $\{\mathbf{b}^{(p)}\}$ ,  $\{\mathbf{r}^{(p)}\}$ ,  $p = 1, \dots, P$ **for**  $t = 1$  **to**  $T$  **do**Run the nudged phase neural dynamics, equation 2.6, with  $\beta \neq 0$  until convergence, collect the fixed point  $\{\mathbf{r}_{\beta,t}^{(p)}\}$ Run the free phase dynamics, equation 2.6, with  $\beta = 0$  until convergence, collect fixed point  $\{\mathbf{r}_{0,t}^{(p)}\}$ Update  $\{\mathbf{L}^{(p)}\}$ ,  $\{\mathbf{W}^{(p)}\}$  and  $\{\mathbf{b}^{(p)}\}$  according to equation 2.10.**end for**

limit of small  $\beta$ , one can approximate the energy function to leading order by

$$E(\boldsymbol{\theta}, \beta; \mathbf{L}^*, \bar{\mathbf{r}}) \approx E(\boldsymbol{\theta}, 0) + \left( \text{Tr} \left( \frac{\partial E}{\partial \mathbf{L}^*} \frac{\partial \mathbf{L}^*}{\partial \beta} \right) + \frac{\partial E}{\partial \bar{\mathbf{r}}} \cdot \frac{\partial \bar{\mathbf{r}}}{\partial \beta} + \frac{\partial E}{\partial \beta} \right) \Big|_{\beta=0} \beta. \quad (2.11)$$

Note that the maximization in equation 2.7 implies  $\frac{\partial E}{\partial \mathbf{L}^*} = \mathbf{0}$ . If  $\frac{\partial E}{\partial \bar{\mathbf{r}}}$  is also  $\mathbf{0}$ , that is, the minima of equation 2.5 are not on the boundaries but at the interior of the feasible set; then in the limit  $\beta \rightarrow 0$ , the gradient of the contrastive function is the gradient of the mean square error function with respect to  $\boldsymbol{\theta}$ :

$$\lim_{\beta \rightarrow 0} \frac{1}{\beta} \frac{\partial J}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} \frac{\partial E}{\partial \beta} \Big|_{\beta=0} = \frac{\partial}{\partial \boldsymbol{\theta}} \frac{1}{T} \sum_{t=1}^T \|\bar{\mathbf{r}}_t^p - \mathbf{z}_t^l\|_2^2. \quad (2.12)$$

It is important to note that while the  $\beta \rightarrow 0$  limit of CSM is related to gradient descent, this limit is not necessarily the best-performing one (as also observed in Scellier & Bengio, 2017, for EP) and  $\beta$  is a hyperparameter to be tuned. In appendix F, we present simulations that confirm the existence of an optimal  $\beta$  away from  $\beta = 0$ .

*2.3.2 Relation to Other Energy-Based Learning Algorithms.* The CSM algorithm is similar in spirit to other contrastive algorithms, such as CHL and EP. Like these algorithms, CSM performs two runs of the neural dynamics in a “free” and a “nudged” phase. However, there are important differences. One is that in CSM, the contrastive function is minimized by the feedforward weights. The lateral weights take part in the maximization of a different minimax objective, equation 2.7. In CHL and EP, such minimization is done with respect to all the weights.

As a consequence of this difference, CSM uses a different update for lateral weights than CHL and EP. This anti-Hebbian update is different in two ways. First, it has the opposite sign, that is, EP and CHL nudged/clamped phase lateral updates are Hebbian. Second, no update is applied in the free phase. As we will demonstrate in numerical simulations, our lateral update imposes a competition between different units in the same layer. When network activity is constrained to be nonnegative, such lateral interactions are inhibitory and sparsify neural activity.

Analogous of two hyperparameters of our algorithm play special roles in EP and CHL. The  $\beta \rightarrow \infty$  limit of EP corresponds to clamping the output to the desired value in the nudged phase (Scellier & Bengio, 2017). Similarly, the  $\beta \rightarrow \infty$  limit of CSM also corresponds to training with fully clamped output units. We discussed the gradient descent interpretation of the  $\beta \rightarrow 0$  limits of both algorithms above. CHL is equivalent to backpropagation when feedback strength, the analog of our  $\gamma$  parameter, vanishes (Xie & Seung, 2003). In CSM,  $\gamma$  is a hyperparameter to be tuned, which we explore in appendix F.

**2.4 Introducing Structured Connectivity.** We can also generalize the nudged supervised similarity matching (see equation 2.3) to derive a Hebbian/anti-Hebbian network with structured connectivity. Following Obeid et al. (2019), we can modify any of the cross terms in the layer-wise similarity matching objective, equation 2.3, by introducing synapse-specific structure constants. For example,

$$-\frac{1}{T^2} \sum_i^{N^{(p)}} \sum_j^{N^{(p-1)}} \sum_t^T \sum_{t'}^T r_{t,i}^{(p)} r_{t',i}^{(p)} r_{t,j}^{(p-1)} r_{t',j}^{(p-1)} s_{ij}^W, \tag{2.13}$$

where  $N^{(p)}$  is the number of neurons in the  $p$ th layer, and  $s_{ij}^W \geq 0$  are constants that set the structure of feedforward weight matrix between the  $p$ th layer and the  $(p - 1)$ th layer. In particular, setting them to zero removes the connection without changing the interpretation of the energy function (Obeid et al., 2019). Similarly, we can introduce constants  $s_{ij}^L$  to specify the structure of the lateral connections (see Figure 6A). Using such structure constants, one can introduce many different architectures, some of which we experiment with below. We present a detailed explanation of these points in appendix C.

### 3 Numerical Simulations

---

In this section, we report the simulation results of the CSM algorithm on a supervised classification task using the MNIST data set of handwritten digits (LeCun, Cortes, & Burges, 2010) and the CIFAR-10 image data set

(Krizhevsky, 2009). For our simulations, we used the Theano deep learning framework (Theano Development Team et al., 2016) and modified the code released by Scellier and Bengio (2017). The activation functions of the units were  $f(x) = \min\{1, \max\{x, 0\}\}$  and  $c^{(p)} = 1/2(1 - \delta_{pp})$ . Following Scellier and Bengio (2017), we used the persistent particle technique to tackle the long period of free phase relaxation. We stored the fixed points of hidden layers at the end of the free phase and used them to initialize the state of the network in the next epoch.

**3.1 MNIST.** The inputs consist of gray-scale 28-by-28 pixel images, and each image is associated with a label ranging from  $\{0, \dots, 9\}$ . We encoded the labels  $\mathbf{z}^l$  as one-hot 10-dimensional vectors. We trained fully connected NNs with one and three hidden layers with lateral connections within each hidden layer. The performance of the CSM algorithm was compared with several variants of the EP algorithm: (1) EP: beta regularized, where the networks had no lateral connections and the sign of  $\beta$  was randomized to act as a regularizer as in Scellier and Bengio (2017); (2) EP: beta positive, where the networks had no lateral connections and  $\beta$  was a positive constant; and (3) EP: lateral, where networks had lateral connections and were trained with a positive constant  $\beta$ . In all the fully connected network simulations for MNIST, the number of neurons in each hidden layer is 500. We attained 0% training error and 2.16% and 3.52% validation errors with CSM, in the one and three hidden-layer cases, respectively. This is on par with the performance of the EP algorithm, which attains validation errors of 2.53% and 2.73%, respectively, for variant 1 and 2.18% and 2.77% for variant 2 (see Figure 3). In the three-layer case, a training error-dependent adaptive learning rate scheme (CSM-Adaptive) was used, wherein the learning rate for the lateral updates is successively decreased when the training error drops below certain thresholds (see appendix D for details).

**3.2 CIFAR-10.** CIFAR-10 is a more challenging data set that contains 32-by-32 RGB images of objects belonging to 10 classes of animals and vehicles. For fully connected networks, the performance of CSM was compared with EP (positive constant  $\beta$ ). We obtain validation errors of 59.21% and 51.76% in the one and two hidden-layer networks, respectively, in CSM, and validation errors of 57.60% and 53.43% in EP (see Figure 4). The mean and standard errors on the mean, of the last 20 validation errors, are reported here in order to account for fluctuations about the mean. It is interesting to note that for both algorithms, deeper networks perform better for CIFAR-10 but not for MNIST. For both data sets, the best-performing network trained with CSM achieves slightly better validation accuracy than the best-performing network trained with EP. The errors corresponding to the fully connected networks for both algorithms and data sets are summarized in Table 1. Here, CSM has been compared to the variant of EP with  $\beta > 0$ .

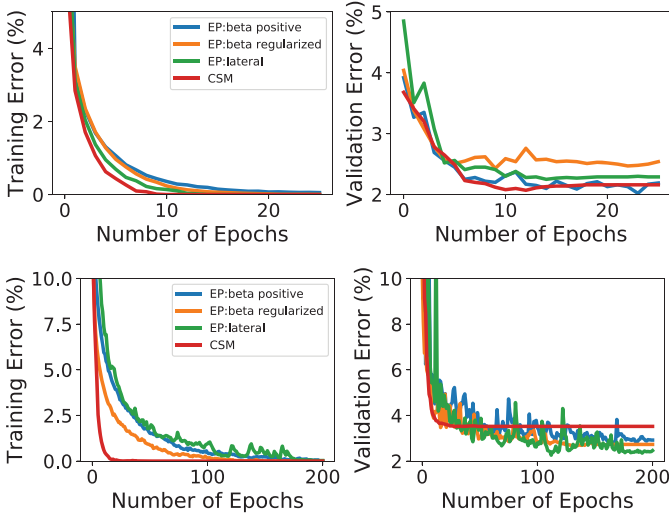


Figure 3: Comparison of training (left) and validation (right) errors between CSM and EP algorithms for a network with one hidden layer (784-500-10, upper panels) and three hidden layers (784-500-500-500-10, lower panels) trained on the MNIST data set.

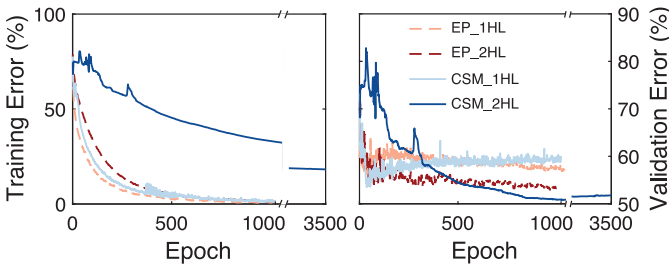


Figure 4: Training (left) and validation (right) error curves for fully connected networks trained on CIFAR-10 data set with CSM (solid) and EP (dashed) algorithms. The best fully connected CSM network attains slightly better validation accuracy than the best fully connected EP network.

For CIFAR-10, the CSM algorithm with two hidden layers has a validation error around 51% after 1000 epochs but was run for a total of 3584 epochs since the training error was still decreasing. The simulation does not reach zero training error but starts plateauing at around 18%, with a decrease of only 0.3% for the last 100 epochs. The validation error does not decrease with the additional training beyond 1000 epochs. It is possible that

Table 1: Comparison of the Training and Validation Errors of Fully Connected Networks for EP (Beta Positive) and CSM.

MNIST			CIFAR-10		
Rule	Train (%)	Validate (%)	Rule	Train (%)	Validate (%)
CSM:1hl	0.00	2.16	CSM:1hl	1.77	59.21 $\pm$ 0.08
EP:1hl	0.03	2.18	EP:1hl	0.76	57.60 $\pm$ 0.06
CSM:3hl	0.00	3.52	CSM:2hl	17.96	51.76 $\pm$ 0.002
EP:3hl	0.00	2.77	EP:2hl	1.25	53.43 $\pm$ 0.04

Notes: For both algorithms, the best-performing networks correspond to two hidden-layer networks for CIFAR-10 and one hidden-layer network for MNIST. Here, xHL means that the network has x hidden layers. For the CIFAR-10, CSM, 2HL simulation, errors at the end of 3584 epochs are reported. For the other CIFAR-10 simulations, errors at the end of 1000 epochs are reported.

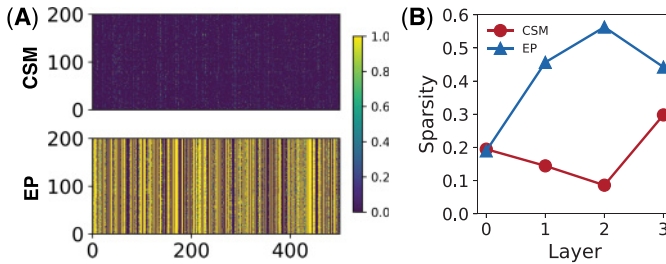


Figure 5: Representations of neurons in NNs trained by the CSM algorithm are much sparser than that of the EP algorithm on the MNIST data set. (A) Heat maps of representations of 500 neurons to a given digit image; each row is the response of 500 neurons to a given digit image. Top: CSM algorithm. Bottom: EP algorithm. (B) Representation sparsity, defined as a fraction of neurons whose activity is larger than a threshold (0.01) along different layers. Layer 0 is the input. The network has a 784-500-500-500-10 architecture.

better validation accuracy could be reached if better training errors were achieved—for example, by better-performing learning rate schedules.

**3.3 Neuronal Representations.** While CSM and EP perform similarly, their learned representations differ in sparseness (see Figure 5). Due to the nonnegativity of hidden unit activity and anti-Hebbian lateral updates, the CSM network ends up with inhibitory lateral connections, which enforce sparse response (see Figure 5). This can also be seen from the similarity matching objective equation 2.3. Imagine there are only two inputs with a negative dot product,  $\mathbf{x} \cdot \mathbf{x}' < 0$ . The next layer will at least partially match this dot product, however, because the lowest value of  $\mathbf{y} \cdot \mathbf{y}'$  is zero due to  $\mathbf{y}, \mathbf{y}' \geq 0, \mathbf{y}$ , and  $\mathbf{y}'$  will be forced to be orthogonal with nonoverlapping sets

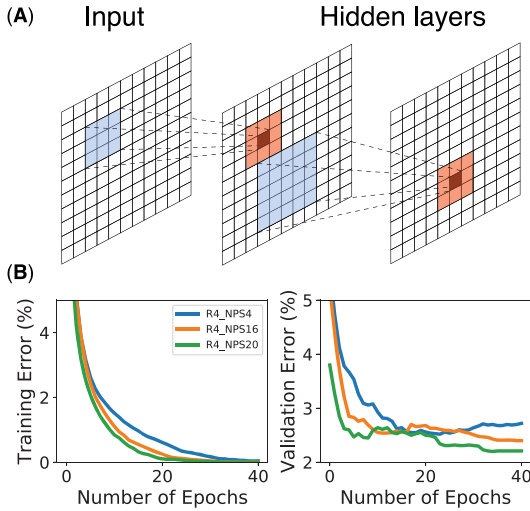


Figure 6: (A) Sketch of structured connectivity in a deep neural network. Neurons live on a 2D grid. Each neuron takes input from a small grid (blue) from the previous layer and a small grid of inhibition from its nearby neurons (orange). (B) Training and validation curves of CSM with structured single hidden-layer networks on the MNIST data set, with a receptive field of radius 4 and neurons per site 4, 16, and 20.

of active neurons. Sparse response is a general feature of cortical neurons (Olshausen & Field, 2004) and energy efficient, making the representations learned by CSM more biologically relevant.

**3.4 Structured Networks.** We also examined the performance of CSM in networks with structured connectivity. Every hidden layer can be constructed by first considering sites arranged on a two-dimensional grid. Each site only receives inputs from selected nearby sites controlled by the radius parameter (see Figure 6A). This setting resembles retinotopy (Kandel, Schwartz, Jessell, Siegelbaum, & Hudspeth, 2000) in the visual cortex. Multiple neurons can be present at a single site controlled by the neurons per site (NPS) parameter. We consider lateral connections only between neurons sharing the same  $(x, y)$  coordinate.

For the MNIST data set, networks with structured connectivity trained with the CSM rule achieved 2.22% validation error for a single hidden-layer network with a radius of 4 and NPS of 20 (see Figure 6B; see appendix D for details). For the CIFAR-10 data set, a one-hidden-layer structured network using CSM algorithm achieves 34% training error and 49.5% validation error after 250 epochs, a significant improvement compared to the fully connected one-layer network. This structured network had a radius of 4 and

NPS of 3. A two-hidden-layer structured network yielded a training error of 46.8% and a validation error of 51.4% after 200 epochs. Errors reported for the structured runs are the averages of five trials. The results for all fully connected and structured networks are reported in appendix D and E.

#### 4 Discussion

---

In this letter, we propose a new solution to the credit assignment problem by generalizing the similarity matching principle to the supervised domain and proposed a biologically plausible supervised learning algorithm, the contrastive similarity matching algorithm. In CSM, a supervision signal is introduced by minimizing the energy difference between a free phase and a nudged phase. CSM differs significantly from other energy-based algorithms in how the contrastive function is constructed. We showed that when a nonnegativity constraint is imposed on neural activity, the anti-Hebbian learning rule for the lateral connections makes the representations sparse and biologically relevant. We also derived the CSM algorithm for neural networks with structured connectivity.

The idea of using representational similarity for training neural networks has taken various forms in previous work. The similarity matching principle has recently been used to derive various biologically plausible unsupervised learning algorithms (Pehlevan & Chklovskii, 2019), such as principal subspace projection (Pehlevan & Chklovskii, 2015), blind source separation (Pehlevan, Mohan, & Chklovskii, 2017), feature learning (Obeid et al., 2019), and manifold learning (Sengupta, Pehlevan, Tepper, Genkin, & Chklovskii, 2018). It has been used for semisupervised classification (Genkin, Sengupta, & Chklovskii, 2019). Similarity matching has also been used as part of a local cost function to train a deep convolutional network (Nøkland & Eidnes, 2019), where instead of layer-wise similarity matching, each hidden layer aims to learn representations similar to the output layer. Representational similarity matrices derived from neurobiology data have recently been used to regularize CNNs trained for image classification. The resulting networks are more robust to noise and adversarial attacks (Li et al., 2019). It would be interesting to study the robustness of neural networks trained by the CSM algorithm.

Like other contrastive learning algorithms, CSM operates with two phases: free and nudged. Previous studies in contrastive learning provided various biologically possible implementations of such two-phased learning. One proposal is to introduce the teacher signal into the network through an oscillatory coupling with a period longer than the timescale of neural activity converging to a steady state. Baldi and Pineda (1991) proposed that such oscillations might be related to rhythms in the brain. In more recent work, Scellier and Bengio (2017) provided an implementation of EP also applicable to CSM with minor modifications. They proposed that synaptic update happens only in the nudged phase with weights continuously updating

according to a differential anti-Hebbian rule as the neuron’s state moves from the fixed point at free phase to the fixed point at the nudged phase. Further, such differential rule can be related to spike-time-dependent plasticity (Xie & Seung, 2000; Bengio, Mesnard, Fischer, Zhang, & Wu, 2015). CSM can use the same mechanism for feedforward and feedback updates. Lateral connections need to be separately updated in the free phase. The differential updating of synapses in different phases of the algorithm can be implemented by neuromodulatory gating of synaptic plasticity (Brzosko, Mierau, & Paulsen, 2019; Bazzari & Parri, 2019).

A practical issue of CSM and other energy-based algorithms such as EP and CHL is that the recurrent dynamics takes a long time to converge. Recently, a discrete-time version of EP has shown much faster training (Ernault, Grollier, Querlioz, Bengio, & Scellier, 2019), and the application to the CSM could be an interesting future direction.

### Appendix A: Derivation of a Supervised Similarity Matching Neural Network

---

The supervised similarity matching cost function, equation 2.1, is formulated in terms of the activities of units, but a statement about the architecture and the dynamics of the network has not been made. We will derive all these from the cost function, without prescribing them. To do so, we need to introduce variables that correspond to the synaptic weights in the network. As it turns out, these variables are dual to correlations between unit activities (Pehlevan et al., 2018).

To see this explicitly, following the method of Pehlevan et al. (2018), we expand the squares in equation 2.1 and introduce new dual variables  $\mathbf{W}_1 \in \mathbb{R}^{m \times n}$ ,  $\mathbf{W}_2 \in \mathbb{R}^{k \times m}$ , and  $\mathbf{L}_1 \in \mathbb{R}^{m \times m}$  using the following identities:

$$\begin{aligned}
 -\frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T \mathbf{y}_t^\top \mathbf{y}_{t'} \mathbf{x}_t^\top \mathbf{x}_{t'} &= \min_{\mathbf{W}_1} -\frac{2}{T} \sum_{t=1}^T \mathbf{y}_t^\top \mathbf{W}_1 \mathbf{x}_t + \text{Tr} \mathbf{W}_1^\top \mathbf{W}_1, \\
 \frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T \mathbf{y}_t^\top \mathbf{y}_{t'} \mathbf{y}_t^\top \mathbf{y}_{t'} &= \max_{\mathbf{L}_1} \frac{2}{T} \sum_{t=1}^T \mathbf{y}_t^\top \mathbf{L}_1 \mathbf{y}_t - \text{Tr} \mathbf{L}_1^\top \mathbf{L}_1, \\
 -\frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T \mathbf{y}_t^\top \mathbf{y}_{t'} \mathbf{z}_t^\top \mathbf{z}_{t'} &= \min_{\mathbf{W}_2} -\frac{2}{T} \sum_{t=1}^T \mathbf{z}_t^\top \mathbf{W}_2 \mathbf{y}_t + \text{Tr} \mathbf{W}_2^\top \mathbf{W}_2. \quad (\text{A.1})
 \end{aligned}$$

Plugging these into equation 2.1, and changing orders of optimization, we arrive at the following dual, minimax formulation of supervised similarity matching:

$$\min_{\mathbf{W}_1, \mathbf{W}_2} \max_{\mathbf{L}_1} \frac{1}{T} \sum_{t=1}^T l_t(\mathbf{W}_1, \mathbf{W}_2, \mathbf{L}_1, \mathbf{x}_t, \mathbf{z}_t), \quad (\text{A.2})$$



where

$$l_t := \text{Tr} \mathbf{W}_1^\top \mathbf{W}_1 - \text{Tr} \mathbf{L}_1^\top \mathbf{L}_1 + \text{Tr} \mathbf{W}_2^\top \mathbf{W}_2 + \min_{\mathbf{y}_t} 2(-\mathbf{y}_t^\top \mathbf{W}_1 \mathbf{x}_t + \mathbf{y}_t^\top \mathbf{L}_1 \mathbf{y}_t - \mathbf{y}_t^\top \mathbf{W}_2^\top \mathbf{z}_t^l). \quad (\text{A.3})$$

A stochastic optimization of the above objective can be mapped to a Hebbian/anti-Hebbian network following steps in Pehlevan et al. (2018). For each training datum,  $\{\mathbf{x}_t, \mathbf{z}_t^l\}$ , a two-step procedure is performed. First, optimal  $\mathbf{y}_t$  that minimizes  $l_t$  is obtained by a gradient flow until convergence:

$$\dot{\mathbf{y}} = \mathbf{W}_1 \mathbf{x}_t - 2\mathbf{L}_1 \mathbf{y}_t + \mathbf{W}_2^\top \mathbf{z}_t^l. \quad (\text{A.4})$$

We interpret this flow as the dynamics of a neural circuit with linear activation functions, where the dual variables  $\mathbf{W}_1$ ,  $\mathbf{W}_2$ , and  $\mathbf{L}_1$  are synaptic weight matrices (see Figure 7A). In the second part of the algorithm, we update the synaptic weights by a gradient descent-ascent on equation A.3 with  $\mathbf{y}_t$  fixed. This gives the following synaptic plasticity rules:

$$\Delta \mathbf{W}_1 = \eta(\mathbf{y}_t \mathbf{x}_t^\top - \mathbf{W}_1), \quad \Delta \mathbf{L}_1 = \eta(\mathbf{y}_t \mathbf{y}_t^\top - \mathbf{L}_1), \quad \Delta \mathbf{W}_2 = \eta(\mathbf{z}_t^l \mathbf{y}_t^\top - \mathbf{W}_2). \quad (\text{A.5})$$

The learning rate  $\eta$  of each matrix can be chosen differently to achieve best performance.

Overall, the network dynamics, equation A.4, and the update rules, equation A.5, map to an NN with one hidden layer, with the output layer clamped to the desired state. The updates of the feedforward weight are Hebbian, and updates of the lateral weight are anti-Hebbian (see Figure 7A).

For prediction, the network takes an input data point,  $\mathbf{x}_t$ , and runs with unclamped output until convergence. We take the value of the  $\mathbf{z}$  units at the fixed point as the network's prediction.

Because the  $\mathbf{z}$  units are not clamped during prediction and are dynamical variables, the correct outputs are not necessarily the fixed points of the network in the prediction phase. To make sure that the network produces correct fixed points, at least for training data, we introduce the following step to the training procedure. We aim to construct a neural dynamics for the output layer in the prediction phase such that its fixed point  $\mathbf{z}$  corresponds to the desired output  $\mathbf{z}^l$ . Since the output layer receives input  $\mathbf{W}_2 \mathbf{y}$  from the previous layer, a decay term that depends on  $\mathbf{z}$  is required to achieve a stable fixed point at  $\mathbf{z} = \mathbf{z}^l$ . The simplest way is introducing lateral inhibition. And now the output layer has the following neural dynamics:

$$\dot{\mathbf{z}} = \mathbf{W}_2 \mathbf{y} - \mathbf{L}_2 \mathbf{z}, \quad (\text{A.6})$$

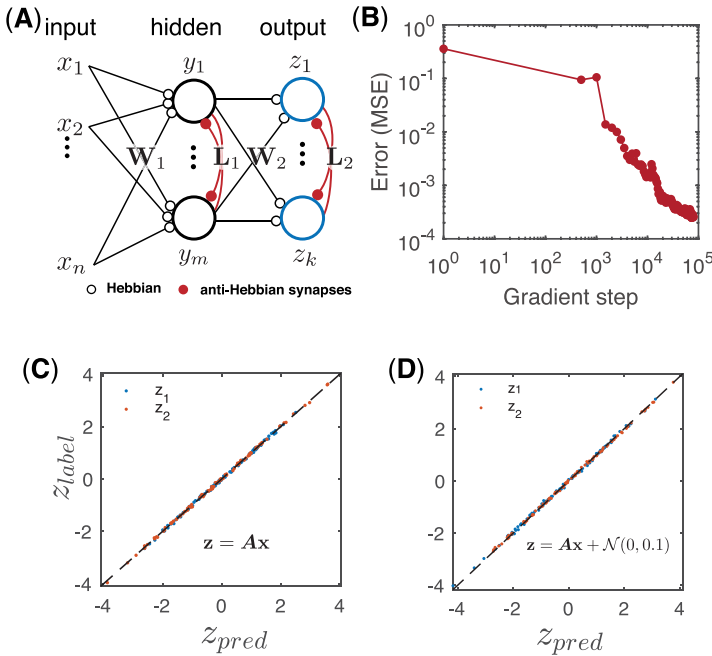


Figure 7: A linear NN with Hebbian/anti-Hebbian learning rules. (A) During the learning process, the output neurons (blue) are clamped at their desired states. After training, prediction for a new input  $x$  is given by the value of  $z$  at the fixed point of neural dynamics. (B) The network is trained on a linear task:  $z_t^l = Ax_t$ . Test error, defined as the mean square error between the network’s prediction,  $z_t^p$ , and the ground-truth value,  $z_t^l$ ,  $1/T \sum_{t=1}^T \|z_t^p - z_t^l\|_F^2$ , decreases with the gradient ascent-descent steps during learning. (C) Scatter plot of the predicted value versus the desired value (element-wise). (D) The algorithm learns the correct mapping between  $x$  and  $z$  even in the presence of small gaussian noise. In these examples,  $x \in \mathbb{R}^5$ ,  $A \in \mathbb{R}^{2 \times 5}$ , elements of  $x$  and  $A$  are drawn from a uniform distribution in the range  $[-1, 1]$ ,  $y \in \mathbb{R}^2$ , and  $z \in \mathbb{R}^2$ . In panels C and D, 200 data points are shown.

where the lateral connections  $L_2$  are learned such that the fixed point  $z^* \approx z^l$ . This is achieved by minimizing the following target function:

$$\min_{L_2} \frac{1}{T} \sum_{t=1}^T \|W_2 y_t - L_2 z_t^l\|_2^2. \tag{A.7}$$

Taking the derivative of the above target function with respect to  $L_2$  while keeping the other parameters and variables evaluated at the fixed

point of neural dynamics, we get the following “delta” learning rule for  $\mathbf{L}_2$ :

$$\Delta \mathbf{L}_2 = \eta (\mathbf{W}_2 \mathbf{y}_t - \mathbf{L}_2 \mathbf{z}_t^l) \mathbf{z}_t^{l\top}. \quad (\text{A.8})$$

After learning, the NN makes a prediction about a new input  $\mathbf{x}$  by running the neural dynamics of  $\mathbf{y}$  and  $\mathbf{z}$ , equations A.4 and A.6, until they converge to a fixed point. We take the value of  $\mathbf{z}$  units at the fixed point as the prediction. As shown in Figures 7B to 7D, the linear network and weight update rule solve linear tasks efficiently.

Although the above procedure can be generalized to multilayer and non-linear networks, one has to address the issue of spurious fixed points of non-linear dynamics for a given input  $\mathbf{x}_t$ . The CSM algorithm presented in the main text overcomes this problem, which borrows ideas from energy-based learning algorithms such as contrastive Hebbian learning and equilibrium propagation.

## Appendix B: Supervised Deep Similarity Matching

In this appendix, we follow Obeid et al. (2019) to derive the minimax dual of deep similarity matching objective function. We start from rewriting the objective function, equation 2.3, by expanding its first term and combining the same terms from adjacent layers, which gives

$$\begin{aligned} \min_{\substack{a_1 \leq r_t^p \leq a_2 \\ t=1, \dots, T \\ p=1, \dots, P}} & \sum_{p=1}^P \frac{\gamma^{p-P}}{2T^2} \sum_{t=1}^T \sum_{t'=1}^T \left( \mathbf{r}_t^{(p)\top} \mathbf{r}_{t'}^{(p)} \mathbf{r}_t^{(p-1)\top} \mathbf{r}_{t'}^{(p-1)} \right. \\ & \left. - \frac{1 + \gamma(1 - \delta_{pP})}{2} c^{(p)} \mathbf{r}_t^{(p)\top} \mathbf{r}_{t'}^{(p)} \mathbf{r}_t^{(p)\top} \mathbf{r}_{t'}^{(p)} \right) \\ & + \sum_{p=1}^P \frac{2\gamma^{p-P}}{T} \sum_{t=1}^T \mathbf{F}(\mathbf{r}_t^{(p)})^\top \mathbf{1} + \frac{\beta}{T} \sum_{t=1}^T \left\| \mathbf{r}_t^{(p)} - \mathbf{z}_t^l \right\|_2^2, \end{aligned} \quad (\text{B.1})$$

where  $c^{(p)}$  is a parameter that changes the relative importance of within-layer and between-layer similarity; we set it to be 1/2 in our numerical simulations. Plug the following identities:

$$\begin{aligned} & -\frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T \mathbf{r}_t^{(p)\top} \mathbf{r}_{t'}^{(p)} \mathbf{r}_t^{(p-1)\top} \mathbf{r}_{t'}^{(p-1)} \\ = \min_{\mathbf{W}^{(p)}} & -\frac{2}{T} \sum_{t=1}^T \mathbf{r}_t^{(p)\top} \mathbf{W}^{(p)} \mathbf{r}_t^{(p-1)} + \text{Tr} \mathbf{W}^{(p)\top} \mathbf{W}^{(p)}, \end{aligned} \quad (\text{B.2})$$

$$\begin{aligned}
 & \frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T \mathbf{r}_t^{(p)\top} \mathbf{r}_{t'}^{(p)} \mathbf{r}_t^{(p)\top} \mathbf{r}_{t'}^{(p)} \\
 &= \max_{\mathbf{L}^{(p)}} \frac{2}{T} \sum_{t=1}^T \mathbf{r}_t^{(p)\top} \mathbf{L}^{(p)} \mathbf{r}_t^{(p)} - \text{Tr} \mathbf{L}^{(p)\top} \mathbf{L}^{(p)}, \tag{B.3}
 \end{aligned}$$

in equation B.1 and exchange the optimization order of  $\mathbf{r}_t^{(p)}$  and the weight matrices. We turn the target function, equation 2.3, into the following min-max problem,

$$\min_{\{\mathbf{W}^{(p)}\}} \max_{\{\mathbf{L}^{(p)}\}} \frac{1}{T} \sum_{t=1}^T l_t \left( \{\mathbf{W}^{(p)}\}, \{\mathbf{L}^{(p)}\}, \mathbf{r}_t^{(0)}, \mathbf{z}_t^l, \beta \right), \tag{B.4}$$

where we have defined an “energy” term (equation 2.5). The neural dynamics of each layer can be derived by following the gradient of  $l_t$ :

$$\begin{aligned}
 \frac{d\mathbf{u}_t^{(p)}}{dt} &\propto -\frac{\partial l_t}{\partial \mathbf{r}_t^{(p)}} = 2\gamma^{p-P} \left[ -\mathbf{u}^{(p)} + \mathbf{b}_t^{(p)} + \mathbf{W}^{(p)} \mathbf{r}_t^{(p-1)} + \gamma(1 - \delta_{pP}) \mathbf{W}^{(p+1)\top} \mathbf{r}_t^{(p+1)} \right. \\
 &\quad \left. - [1 + \gamma(1 - \delta_{pP})] \mathbf{c}^{(p)} \mathbf{L}^{(p)} \mathbf{r}_t^{(p)} - 2\beta \delta_{pP} (\mathbf{r}_t^{(P)} - \mathbf{z}_t^l) \right], \\
 \mathbf{r}_t^{(p)} &= \mathbf{f}(\mathbf{u}^{(p)}). \tag{B.5}
 \end{aligned}$$

Define  $\tau_p^{-1} = 2\gamma^{p-P}$ , and the above equation becomes equation 2.6.

### Appendix C: Supervised Similarity Matching for Neural Networks with Structured Connectivity

In this appendix, we derive the supervised similarity matching algorithm for neural networks with structured connectivity. Structure can be introduced to the quartic terms in equation B.1:

$$\begin{aligned}
 & -\frac{1}{T^2} \sum_i^{N^{(p)}} \sum_j^{N^{(p-1)}} \sum_t^T \sum_{t'}^T r_{t,i}^{(p)} r_{t',i}^{(p)} r_{t,j}^{(p-1)} r_{t',j}^{(p-1)} S_{ij}^{W,(p)}, \\
 & -\frac{1}{T^2} \sum_i^{N^{(p)}} \sum_j^{N^{(p)}} \sum_t^T \sum_{t'}^T r_{t,i}^{(p)} r_{t',i}^{(p)} r_{t,j}^{(p)} r_{t',j}^{(p)} S_{ij}^{L,(p)}, \tag{C.1}
 \end{aligned}$$

where  $s_{ij}^{W,(p)}$  and  $s_{ij}^{L,(p)}$  specify the feedforward connections of layer  $p$  with  $p-1$  layer and lateral connections within layer  $p$ , respectively. For example, setting them to be zeros eliminates all connections. Now we have the following deep structured similarity matching cost function for supervised learning:

$$\begin{aligned} \min_{\substack{a_1 \leq t' \leq a_2 \\ t=1, \dots, T \\ p=1, \dots, P}} \sum_{p=1}^P \frac{\gamma^{p-P}}{2T^2} \sum_{t=1}^T \sum_{t'=1}^T \left( \mathbf{r}_t^{(p)\top} \mathbf{r}_{t'}^{(p)} \mathbf{r}_t^{(p-1)\top} \mathbf{r}_{t'}^{(p-1)} s_{ij}^{W,(p)} \right. \\ \left. - \frac{1 + \gamma(1 - \delta_{pP})}{2} \mathbf{r}_t^{(p)\top} \mathbf{r}_{t'}^{(p)} \mathbf{r}_t^{(p)\top} \mathbf{r}_{t'}^{(p)} s_{ij}^{L,(p)} \right) \\ + \sum_{i=1}^P \frac{2\gamma^{p-P}}{T} \sum_{t=1}^T \mathbf{F}(\mathbf{r}_t^{(p)})^\top \mathbf{1} + \frac{\beta}{T} \sum_{t=1}^T \left\| \mathbf{r}_t^{(p)} - \mathbf{z}_t^l \right\|_2^2. \end{aligned} \quad (\text{C.2})$$

For each layer, we can define dual variables for  $\mathbf{W}_{ij}^{(p)}$  and  $\mathbf{L}_{ij}^{(p)}$  for interactions with positive constants and define the following variables:

$$\bar{W}_{ij}^{(p)} = \begin{cases} W_{ij}^{(p)}, & s_{ij}^{W,(p)} \neq 0 \\ 0, & s_{ij}^{W,(p)} = 0 \end{cases}, \quad \bar{L}_{ij}^{(p)} = \begin{cases} L_{ij}^{(p)}, & s_{ij}^{L,(p)} \neq 0 \\ 0, & s_{ij}^{L,(p)} = 0 \end{cases} \quad (\text{C.3})$$

Now we can rewrite equation C.2 as

$$\min_{\{\bar{\mathbf{W}}^{(p)}\}} \max_{\{\bar{\mathbf{L}}^{(p)}\}} \frac{1}{T} \sum_{t=1}^T \bar{l}_t \left( \{\bar{\mathbf{W}}^{(p)}\}, \{\bar{\mathbf{L}}^{(p)}\}, \mathbf{r}_t^{(0)}, \mathbf{z}_t^l, \beta \right), \quad (\text{C.4})$$

where

$$\begin{aligned} \bar{l}_t := \min_{\substack{a_1 \leq t^{(p)} \leq a_2 \\ p=1, \dots, P}} \sum_{p=1}^P \gamma^{p-P} \left\{ \sum_{\substack{i,j \\ s_{ij}^{W,(p)} \neq 0}} W_{ij}^{(p)2} - \sum_{\substack{i,j \\ s_{ij}^{L,(p)} \neq 0}} \frac{1 + \gamma(1 - \delta_{pP})}{s_{ij}^{L,(p)}} L_{ij}^{(p)2} \right. \\ \left. + [1 + \gamma(1 - \delta_{pP})] \mathbf{r}_t^{(p)\top} \mathbf{L}^{(p)} \mathbf{r}_t^{(p)} - 2\mathbf{r}_t^{(p)\top} \mathbf{W}^{(p)} \mathbf{r}_t^{(p-1)} + 2\mathbf{F}(\mathbf{r}_t^{(p)})^\top \right\} \\ + \beta \left\| \mathbf{r}_t^{(p)} - \mathbf{z}_t^l \right\|_2^2. \end{aligned} \quad (\text{C.5})$$

Table 2: Comparison of the Training and Validation Errors of Different Algorithms for One Hidden-Layer NNs on the MNIST Data Set.

Algorithm	Learning Rate	Training Error (%)	Validation Error (%)	Number of Epochs
EP: $\pm\beta$	$\alpha_W = 0.1, 0.05$	0	2.53	40
EP: $+\beta$	$\alpha_W = 0.5, 0.125$	0.034	2.18	100
EP: lateral	$\alpha_W = 0.5, 0.25, \alpha_L = 0.75$	0	2.29	25
CSM	$\alpha_W = 0.5, 0.375, \alpha_L = 0.01$	0	2.16	25

The neural dynamics follows the gradient of equation C.5, which is

$$\begin{aligned} \tau_p \frac{d\mathbf{u}_t^{(p)}}{dt} &= -\mathbf{u}^{(p)} + \mathbf{b}_t^{(p)} + \bar{\mathbf{W}}^{(p)} \mathbf{r}_t^{(p-1)} + \gamma(1 - \delta_{pP}) \bar{\mathbf{W}}^{(p+1)\top} \mathbf{r}_t^{(p+1)} \\ &\quad - [1 + \gamma(1 - \delta_{pP})] \bar{\mathbf{L}}^{(p)} \mathbf{r}_t^{(p)} - \beta \delta_{pP} (\mathbf{r}_t^{(P)} - \mathbf{z}^l), \\ \mathbf{r}_t^{(p)} &= \mathbf{f}(\mathbf{u}_t^{(p)}), \quad p = 1, \dots, P. \end{aligned} \quad (\text{C.6})$$

Local learning rules follow the gradient descent and ascent of equation C.5:

$$\Delta W_{ij}^{(p)} \propto \left( r_j^{(p)} r_i^{(p-1)} - \frac{W_{ij}^{(p)}}{S_{ij}^{W,(p)}} \right), \quad (\text{C.7})$$

$$\Delta L_{ij}^{(p)} \propto \left( r_j^{(p)} r_i^{(p)} - \frac{L_{ij}^{(p)}}{S_{ij}^{L,(p)}} \right). \quad (\text{C.8})$$

## Appendix D: Hyperparameters and Performance in Numerical Simulations with the MNIST Data Set

**D.1 One Hidden Layer.** Table 2 reports the training and validation errors of three variants of the EP algorithm and the CSM algorithm for a single hidden-layer network on MNIST. The models were trained until the training error dropped to 0% or as close to 0% as possible (as in the case of EP algorithm with  $\beta > 0$ ); errors reported here correspond to errors obtained for specific runs and do not reflect ensemble averages. The training and validation errors below are reported at an epoch when the training error has dropped to 0%, or at the last epoch for the run (e.g., for EP,  $\beta > 0$ ). This epoch number is recorded in the last column.

**D.2 Three Hidden Layers.** In Table 3, the CSM algorithm employs a scheme with decaying learning rates. Specifically, the learning rates for

Table 3: Comparison of the Training and Validation Errors of Different Algorithms for Three-Hidden Layer NNs on the MNIST Data Set.

Algorithm	Learning Rate	Training Error (%)	Validation Error (%)	Number of Epochs
EP: $\pm\beta$	$\alpha_W=0.128, 0.032, 0.008, 0.002$	0	2.73	250
EP: $+\beta$	$\alpha_W=0.128, 0.032, 0.008, 0.002$	0	2.77	250
EP lateral	$\alpha_W=0.128, 0.032, 0.008, 0.002$ ; $\alpha_L=0.192, 0.048, 0.012$	0	2.4	250
CSM	$\alpha_W=0.5, 0.375, 0.281, 0.211$ ; $\alpha_L=0.75, 0.562, 0.422$	0	4.82	250
CSM Adaptive	$\alpha_W=0.5, 0.375, 0.281, 0.211$ ; $\alpha_L=0.75, 0.562, 0.422$	0	3.52	250

Table 4: Comparison of the Training and Validation Errors of Different Algorithms for One Hidden-Layer NNs with Structured Connectivity on the MNIST Data Set.

Algorithm	Learning Rate	Training Error (%)	Validation Error (%)	Number of Epochs
R4, NPS4, Full	$\alpha_W=0.5, 0.375$ ; $\alpha_L=0.01$	0.02	2.71	50
R4, NPS16, Full	$\alpha_W=0.5, 0.25$ ; $\alpha_L=0.75$	0	2.41	49
R4, NPS20, Full	$\alpha_W=0.664, 0.577$ ; $\alpha_L=0.9$	0	2.22	50
R8, NPS80, Crop	$\alpha_W=0.664, 0.577$ ; $\alpha_L=0.9$	0.01	2.27	20
R4, NPS4, Crop	$\alpha_W=0.099, 0.065$ ; $\alpha_L=0.335$	0.08	2.98	100
R8, NPS4, Crop	$\alpha_W=0.099, 0.065$ ; $\alpha_L=0.335$	0	2.73	100
R8, NPS20, Crop	$\alpha_W=0.664, 0.577$ ; $\alpha_L=0.9$	0	2.23	79

lateral updates are divided by a factor of 5, 10, 50, and 100 when the training error dropped below 5%, 1%, 0.5%, and 0.1%, respectively.

**D.3 Structured Connectivity.** In this section, we explain the simulation for structured connectivity and report the results. Every hidden layer in these networks can be considered as multiple two-dimensional grids stacked onto each other, with each grid containing neurons or units at periodically arranged sites. Each site receives inputs only from selected nearby sites. In this scheme, we consider lateral connections only between neurons sharing the same  $(x, y)$  coordinate, and the length and width of the grid are the same. In Table 4, “Full” refers to simulations where the input is the  $28 \times 28$  MNIST input image, and “Crop” refers to simulations in which the input image is a cropped  $20 \times 20$  MNIST image. The first three, annotated by Full, correspond to the simulations reported in the main text. Errors are reported at the last epoch for the run. In networks with

Table 5: Comparison of the Validation Errors of Different Algorithms for Different Networks on the CIFAR-10 Data Set.

Algorithm, Connectivity, Number of Hidden Layers	Learning Rate	Training Error (%)	Validation Error (%)	Number of Errors
CSM, FC, 1HL	$\alpha_W = 0.059, 0.017$ $\alpha_L = 0.067$	1.77	$59.21 \pm 0.08$	1000
CSM, FC, 2HL	$\alpha_W = 0.018, 7.51 \times 10^{-4}, 3.07 \times 10^{-5}$ $\alpha_L = 0.063, 2.59 \times 10^{-3}$	17.96	$51.76 \pm 0.002$	3584
CSM, Str, 1HL	$\alpha_W = 0.050, 0.0375$ $\alpha_L = 0.01$	$34 \pm 3.7$	$49.5 \pm 0.7$	250
CSM, Str, 2HL	$\alpha_W = 0.265, 0.073, 0.020$ $\alpha_L = 0.075, 0.020$	$46.8 \pm 0.6$	$51.4 \pm 0.7$	200
EP, FC, 1HL	$\alpha_W = 0.014, 0.011$	0.76	$57.60 \pm 0.06$	1000
EP, FC, 2HL	$\alpha_W = 0.014, 0.011, 1.25$	1.25	$53.43 \pm 0.04$	1000

structural connectivity, additional hyperparameters are required to constrain the structure, which are enumerated below:

- **Neurons per site (nps):** The number of neurons placed at each site in a given hidden layer, that is, the number of two-dimensional grids stacked onto each other. The nps for the input is 1.
- **Stride:** Spacing between adjacent sites, relative to the input channel. The stride of the input is always 1, that is, sites are placed at (0, 0), (0, 1), (1, 0), so on, on the two-dimensional grid. If the stride of the  $l$ th layer is  $s$ , the nearest sites to the site at the origin will be (0,  $s$ ) and ( $s$ , 0). The stride increases deeper into the network. Specifying the stride also determines the dimension of the grid. A layer with stride  $s$  and nps  $n$  will have  $d \times d \times n$  units, where  $d = 28/s$  for the Full runs and  $d = 20/s$  for the Crop runs. The nps values and stride together assign coordinates to all the units in the network.
- **Radius:** The radius of the circular two-dimensional region that all units in the previous layer must lie within in order to have nonzero weights to the current unit. Any units in the previous layer, lying outside the circle, will not be connected to the unit.

## Appendix E: Hyperparameters and Performance in Numerical Simulations with the CIFAR-10 Data Set

Table 5 records the training and validation errors obtained for the CSM and EP algorithms for fully connected networks, as well as for CSM with structured networks, on the CIFAR-10 data set. The validation error column for fully connected runs reports the mean of the last 20 validation errors



Table 6: Validation Errors at the End of the Training Period for a Fully Connected One-Hidden-Layer Network Trained on MNIST, with Different  $\beta$  Values.

$\beta$ value	Mean Validation Error (%)	Minimum Validation Error (%)	Maximum Validation Error (%)
0.01	89.70	89.70	89.70
0.1	90.09	90.09	90.09
0.25	46.20	2.28	90.09
0.5	2.19	2.17	2.21
0.75	2.42	2.22	2.51
1.0	2.36	2.26	2.48
1.2	23.30	2.36	85.91
1.5	2.62	2.40	2.75
2.0	79.55	79.55	79.55

Notes: For values that lay within the parameter range that converged to low ( $< 3\%$ ) validation errors, four trials were run, and the mean, minimum, and maximum errors over the trials have been reported. In all runs,  $\gamma = 1$ .

Table 7: Validation Errors at the End of the Training Period for a Fully Connected One-Hidden-Layer Network Trained on MNIST, with Different  $\gamma$  Values.

$\gamma$ value	Mean Validation Error (%)	Minimum Validation Error (%)	Maximum Validation Error (%)
0.2	2.75	2.64	2.85
0.5	2.51	2.43	2.60
0.7	2.38	2.24	2.47
0.8	2.41	2.32	2.47
0.9	2.26	2.21	2.31
1.0	2.45	2.38	2.53
1.1	2.46	2.37	2.63
1.2	2.35	2.26	2.48
1.3	2.43	2.28	2.54
1.5	83.16	83.16	83.16
1.8	90.09	90.09	90.09

Notes: For parameters that converged to low ( $< 3\%$ ) validation errors, four trials were run, and the mean, minimum, and maximum errors over the trials have been reported. In all runs,  $\beta = 1$ .

reported at the end of the training period, as well as the standard error on the mean. For the structured runs, the training and validation errors reported are the average of the last epoch's reported errors from five trials and the standard error on the means. This is done in order to account for fluctuations in the error during training.

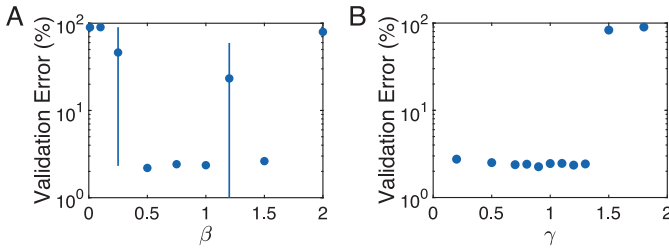


Figure 8: Validation error of a single hidden-layer network trained by the CSM algorithm on the MNIST data set as a function of parameter  $\beta$  (A) and  $\gamma$  (B). Four trials were conducted for values for which the validation error was less than 3%. Dots indicate the mean validation error over trials, and error bars indicate the standard deviation over trials.

## Appendix F: Performance of CSM as a Function of Nudge Strength and Feedback Strength

---

In the nudged deep similarity matching objective, equation 2.3,  $\beta$  controls the strength of nudging, while  $\gamma$  specifies the strength of feedback input compared with the feedforward input. Scellier and Bengio (2017) used  $\beta = 1$  in their simulations. In all the simulations reported here in the main text, we have set  $\beta = 1$ ,  $\gamma = 1$ . In this appendix, we trained a single hidden-layer network using CSM on MNIST, while systematically varying the value of  $\beta$  and  $\gamma$  and keeping other parameters fixed. The validation errors for these experiments are documented in Tables 6 and 7 and plotted in Figure 8. We find that the network has optimal values for  $\gamma$  less than 1.5 and  $\beta$  in the range bounded by 0.5 and 1. At these values, the network is able to converge to low validation errors ( $<3\%$ ).

## Acknowledgments

---

This work was supported by NIH (1UF1NS111697-01), the Intel Corporation through Intel Neuromorphic Research Community, and a Google Faculty Research Award. We thank Dina Obeid and Blake Bordelon for helpful discussions.

## References

---

- Anderson, J. R., & Peterson, C. (1987). A mean field theory learning algorithm for neural networks. *Complex Systems*, 1, 995–1019.
- Baldi, P., & Pineda, F. (1991). Contrastive learning and neural oscillations. *Neural Computation*, 3(4), 526–545.

- Bazzari, A. H., & Parri, H. R. (2019). Neuromodulators and long-term synaptic plasticity in learning and memory: A steered-glutamatergic perspective. *Brain Sciences*, 9(11), 300.
- Belilovsky, E., Eickenberg, M., & Oyallon, E. (2018). *Greedy layerwise learning can scale to imagenet*. arXiv:1812.11446.
- Bengio, Y., Mesnard, T., Fischer, A., Zhang, S., & Wu, Y. (2015). *STDP as presynaptic activity times rate of change of postsynaptic activity*. arXiv:1509.05936.
- Brzosko, Z., Mierau, S. B., & Paulsen, O. (2019). Neuromodulation of spike-timing-dependent plasticity: Past, present, and future. *Neuron*, 103(4), 563–581.
- Crick, F. (1989). The recent excitement about neural networks. *Nature*, 337(6203), 129–132.
- Ernault, M., Grollier, J., Querlioz, D., Bengio, Y., & Scellier, B. (2019). Updates of equilibrium prop match gradients of backprop through time in an RNN with static input. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'AlchéBuc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*, 32 (pp. 7079–7089). Red Hook, NY: Curran.
- Genkin, A., Sengupta, A. M., & Chklovskii, D. (2019). A neural network for semi-supervised learning on manifolds. In *Proceedings of the International Conference on Artificial Neural Networks* (pp. 375–386). Berlin: Springer.
- Grill-Spector, K., & Weiner, K. S. (2014). The functional architecture of the ventral temporal cortex and its role in categorization. *Nature Reviews Neuroscience*, 15(8), 536–548.
- Guerguiev, J., Lillicrap, T. P., & Richards, B. A. (2017). Towards deep learning with segregated dendrites. *eLife*, 6, e22901.
- Hinton, G. E., & McClelland, J. L. (1988). Learning representations by recirculation. In D. Anderson (Ed.), *Neural information processing systems* (pp. 358–366). College Park, MD: American Institute of Physics.
- Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S., & Hudspeth, A. (2000). *Principles of neural science*. New York: McGraw-Hill.
- Kriegeskorte, N., Mur, M., Ruff, D. A., Kiani, R., Bodurka, J., Esteky, H., . . . Bandettini, P. A. (2008). Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron*, 60(6), 1126–1141.
- Krizhevsky, A. (2009). Learning multiple layers of features from tiny images (Master's thesis, Department of Computer Science). Toronto: University of Toronto.
- LeCun, Y., Cortes, C., & Burges, C. (2010). *MNIST handwritten digit database*. <http://yann.lecun.com/exdb/mnist>.
- Lee, D.-H., Zhang, S., Fischer, A., & Bengio, Y. (2015). Difference target propagation. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 498–515). Berlin: Springer.
- Li, Z., Brendel, W., Walker, E., Cobos, E., Muhammad, T., Reimer, . . . J., Tolias, A. (2019). Learning from brains how to regularize machines. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'AlchéBuc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*, 32 (pp. 9525–9535). Red Hook, NY: Curran.
- Lillicrap, T. P., Cownden, D., Tweed, D. B., & Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7, 13276.

- Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., & Hinton, G. (2020). Backpropagation and the brain. *Nature Reviews Neuroscience*, *21*, 335–336.
- Movellan, J. R. (1991). Contrastive Hebbian learning in the continuous Hopfield model. In D. S. Touretzky, J. L. Ellman, T. J. Sejnowski, & G. E. Hinton (Eds.), *Connectionist models* (pp. 10–17). Amsterdam: Elsevier.
- Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems*, *29* (pp. 1037–1045). Red Hook, NY: Curran.
- Nøkland, A., & Eidnes, L. H. (2019). *Training neural networks with local error signals*. arXiv:1901.06656.
- Obeid, D., Ramambason, H., & Pehlevan, C. (2019). Structured and deep similarity matching via structured and deep Hebbian networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*, *32*, (pp. 15377–15386). Red Hook, NY: Curran.
- Olshausen, B. A., & Field, D. J. (2004). Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, *14*(4), 481–487.
- Ororbica, A. G., & Mali, A. (2019). Biologically motivated algorithms for propagating local target representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, *33* (pp. 4651–4658). Palo Alto, CA: AAAI.
- Pehlevan, C., & Chklovskii, D. (2015). A normative theory of adaptive dimensionality reduction in neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems*, *28* (pp. 2269–2277). Red Hook, NY: Curran.
- Pehlevan, C., & Chklovskii, D. B. (2019). Neuroscience-inspired online unsupervised learning algorithms: Artificial neural networks. *IEEE Signal Processing Magazine*, *36*(6), 88–96.
- Pehlevan, C., Mohan, S., & Chklovskii, D. B. (2017). Blind nonnegative source separation using biological neural networks. *Neural Computation*, *29*(11), 2925–2954.
- Pehlevan, C., Sengupta, A. M., & Chklovskii, D. B. (2018). Why do similarity matching objectives lead to Hebbian/anti-Hebbian networks? *Neural Computation*, *30*(1), 84–124.
- Richards, B. A., & Lillicrap, T. P. (2019). Dendritic solutions to the credit assignment problem. *Current Opinion in Neurobiology*, *54*, 28–36.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.
- Sacramento, J., Costa, R. P., Bengio, Y., & Senn, W. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems*, *31* (pp. 8721–8732). Red Hook, NY: Curran.
- Scellier, B., & Bengio, Y. (2017). Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, *11*, 24.
- Sengupta, A., Pehlevan, C., Tepper, M., Genkin, A., & Chklovskii, D. (2018). Manifold-tiling localized receptive fields are optimal in similarity-preserving

- neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems 31* (pp. 7080–7090). Red Hook, NY: Curran.
- Theano Development Team, Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., . . . Zhang, Y. (2016). *Theano: A Python framework for fast computation of mathematical expressions*. arXiv:1605.02688.
- Whittington, J. C., & Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity. *Neural Computation, 29*(5), 1229–1262.
- Whittington, J. C., & Bogacz, R. (2019). Theories of error backpropagation in the brain. *Trends in Cognitive Sciences, 23*, 235–250.
- Xie, X., & Seung, H. S. (2000). Spike-based learning rules and stabilization of persistent neural activity. In S. Solla, T. Leen, & K. Müller (Eds.), *Advances in neural information processing systems, 12* (pp. 199–208). Cambridge, MA: MIT Press.
- Xie, X., & Seung, H. S. (2003). Equivalence of backpropagation and contrastive Hebbian learning in a layered network. *Neural Computation, 15*(2), 441–454.
- Yamins, D. L., & DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience, 19*(3), 356.

---

Received July 23, 2020; accepted November 23, 2020.