Correlative Information Maximization: A Biologically Plausible Approach to Supervised Deep Neural Networks without Weight Symmetry

Bariscan Bozkurt[1,2,3] **Cengiz Pehlevan**[4,5] **Alper T. Erdogan**[2,3]
[1] Gatsby Computational Neuroscience Unit, UCL, United Kingdom
[2]KUIS AI Center, Koc University, Turkey [3]EEE Department, Koc University, Turkey
[4]John A. Paulson School of Engineering & Applied Sciences and Center for Brain Science, Harvard University, Cambridge, 02138 MA, USA
[5]Kempner Institute for the Study of Natural and Artificial Intelligence
{bbozkurt15, alperdogan}@ku.edu.tr cpehlevan@seas.harvard.edu

Abstract

The backpropagation algorithm has experienced remarkable success in training large-scale artificial neural networks; however, its biological plausibility has been strongly criticized, and it remains an open question whether the brain employs supervised learning mechanisms akin to it. Here, we propose correlative information maximization between layer activations as an alternative normative approach to describe the signal propagation in biological neural networks in both forward and backward directions. This new framework addresses many concerns about the biological-plausibility of conventional artificial neural networks and the backpropagation algorithm. The coordinate descent-based optimization of the corresponding objective, combined with the mean square error loss function for fitting labeled supervision data, gives rise to a neural network structure that emulates a more biologically realistic network of multi-compartment pyramidal neurons with dendritic processing and lateral inhibitory neurons. Furthermore, our approach provides a natural resolution to the weight symmetry problem between forward and backward signal propagation paths, a significant critique against the plausibility of the conventional backpropagation algorithm. This is achieved by leveraging two alternative, yet equivalent forms of the correlative mutual information objective. These alternatives intrinsically lead to forward and backward prediction networks without weight symmetry issues, providing a compelling solution to this long-standing challenge.

1 Introduction

How biological neural networks learn in a supervised manner has long been an open problem. The backpropagation algorithm [1], with its remarkable success in training large-scale artificial neural networks and intuitive structure, has inspired proposals for how biologically plausible neural networks can perform the necessary efficient credit-assignment for supervised learning in deep neural architectures [2]. Nonetheless, certain aspects of the backpropagation algorithm, combined with the oversimplified nature of artificial neurons, have been viewed as impediments to proposals rooted in this inspiration [3].

One of the primary critiques regarding the biological plausibility of the backpropagation algorithm is the existence of a parallel backward path for backpropagating error from the output towards the input, which uses the same synaptic weights as the forward path [1, 2, 4]. Although such weight transport, or weight symmetry, is deemed highly unlikely based on experimental evidence [3, 4],

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

some biologically plausible frameworks still exhibit this feature, which is justified by the symmetric structure of the Hebbian updates employed in these frameworks [2, 5, 6].

The concerns about the simplicity of artificial neurons have been addressed by models which incorporate multi-compartment neuron models into networked architectures and ascribe important functions to dendritic processing in credit assignment [7, 8, 9, 10]. This new perspective has enabled the development of neural networks with improved biological plausibility.

In this article, we propose the use of correlative information maximization (CorInfoMax) among consecutive layers of a neural network as a new supervised objective for biologically plausible models, which offers

- a principled solution to the weight symmetry problem: our proposed information theoretic criterion aims to maximize the linear dependence between the signals in two neighboring layers, naturally leading to the use of linear or affine transformations in between them. A key property of this approach is that employing two alternative expressions for the correlative mutual information (CMI) results in potentially *asymmetric forward and backward prediction networks*, offering a natural solution to the weight transport problem. Consequently, predictive coding in both directions emerges as the inherent solution to the correlative information maximization principle, fostering signal transmission in both forward and top-down directions through asymmetrical connections. While the CorInfoMax principle enhances information flow in both directions, the introduction of set membership constraints on the layer activations, such as non-negativity, through activation nonlinearities and lateral inhibitions, encourages compression of information and sparse representations [11].
- a normative approach for deriving networks with multi-compartment neurons: the gradient-based optimization of the CorInfoMax objective naturally leads to network models that employ multi-compartment pyramidal neuron models accompanied by interneurons as illustrated in Figure 1.

As derived and explained in detail in Section 2, the resulting networks incorporate lateral connections and auto-synapses (autapses) to increase the entropy of a layer, promoting utilization of all dimensions within the representation space of that layer. Meanwhile, asymmetric feedforward and feedback connections act as forward and backward predictors of layer activation signals, respectively, to reduce the conditional entropies between layers, targeting the elimination of redundancy.

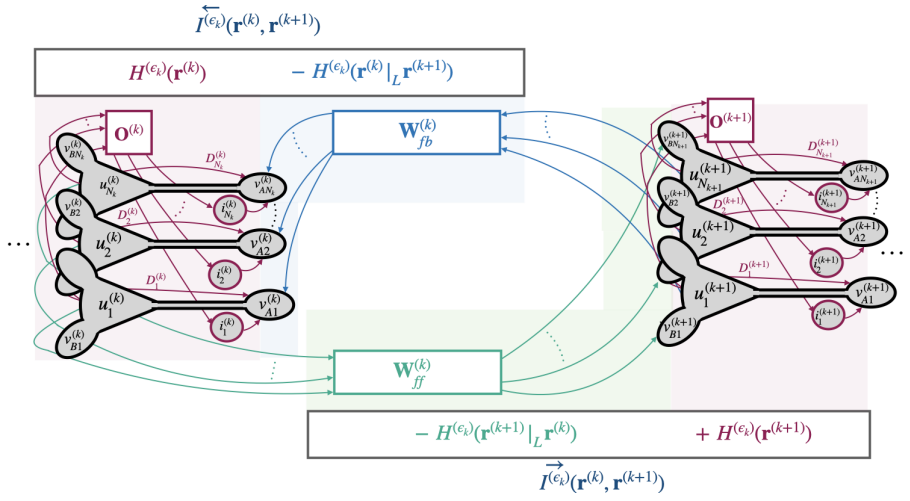


Figure 1: A segment of a correlative information maximization based neural network. Each layer consists of three-compartment pyramidal neurons with outputs $\mathbf{r}^{(k)}$ and membrane voltages ($\mathbf{u}^{(k)}$-soma, $\mathbf{v}_B^{(k)}$-basal dendrites, $\mathbf{v}_A^{(k)}$-distal apical dendrites) and interneurons with outputs $\mathbf{i}^{(k)}$. The CMI expression $\overrightarrow{I}^{(\epsilon_k)}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})$ ($\overleftarrow{I}^{(\epsilon_k)}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})$) defines forward (backward) prediction synapses $\mathbf{W}_{ff}^{(k)}$ ($\mathbf{W}_{fb}^{(k)}$), for minimizing $H^{(\epsilon_k)}(\mathbf{r}^{(k+1)}|_L \mathbf{r}^{(k)})$ ($H^{(\epsilon_k)}(\mathbf{r}^{(k)}|_L \mathbf{r}^{(k+1)})$) and the lateral connections $\mathbf{O}^{(k+1)}$ ($\mathbf{O}^{(k)}$) and autapses $\mathbf{D}^{(k+1)}$ ($\mathbf{D}^{(k)}$) connected to distal apical dendrites at the $k+1$-th (k-th) layer, for maximizing $H^{(\epsilon_k)}(\mathbf{r}^{(k+1)})$ ($H^{(\epsilon_k)}(\mathbf{r}^{(k)})$).

1.1 Related work

1.1.1 Multi-compartmental neuron model based biologically plausible approaches

Experimentally grounded studies, such as [7, 12], have been influential for considering a role for dendritic-processing in multi-compartmental neurons for learning and credit assignment [13]. Subsequent research has explored biologically plausible models with supervised learning functionality, such as the two-compartment neuron model by Urbanczik and Senn [8] and the three-compartment pyramidal neuron model by Sacramento et al. [9]. Both models integrate non-Hebbian learning and spike-time dependent plasticity, while the latter includes SST interneurons [14]. Similar frameworks have been proposed by [15] and [10], with the latter introducing a normative framework based on multi-compartmental neuron structure, top-down feedback, lateral and feedforward connections, and Hebbian and non-Hebbian learning rules, emerging from the optimization of a prediction error objective with a whitening constraint on co-layer neurons.

In a similar vein to [10], we propose an alternative normative framework based on information maximization principle. In this framework, the three-compartment structure and associated forward, top-down and lateral synaptic connections stem from the maximization of CMI between adjacent layers, without the imposition of any whitening constraint.

1.1.2 Weight symmetry problem

A central concern regarding the biological plausibility of the backpropagation algorithm pertains to the weight symmetry issue: synaptic weights in the feedback path for error backpropagation are transposes of those used in the forward inference path [2, 3, 16]. The requirement of tied weights in backpropagation is questionable for physically distinct feedforward and feedback paths in biological systems, leading many researchers to focus on addressing the weight symmetry issue.

Various strategies have been devised to address the weight symmetry issue. For example, the feedback alignment approach, which fixes randomly initialized feedback weights and adapts feedforward weights, was offered as a plausible solution [17]. Later Akrout et.al. [18] proposed its extension by updating feedback weights towards to the transpose of the feedforward weights. Along the similar lines, Amit introduced antisymmetry through separate random initializations [19]. Liao et al. [20] showed that the sign of the feedback weights (rather than their magnitude) affects the learning performance, and proposed the sign-symmetry algorithm.

Intriguingly, this symmetric weight structure is also observed in biologically plausible frameworks such as predictive coding (PC) [21, 22, 23], equilibrium propagation (EP) [24, 25, 26], and similarity matching [27]. This phenomenon can be rationalized by the transpose symmetry of the Hebbian update with respect to inputs and outputs. The EP framework in [25] unties forward and backward connections inspired by [28, 29], and only yields small performance degradation. A more recent approach by Golkar et al. [10] addresses this challenge by integrating two alternative forward prediction error loss function terms associated with the same network layer and leveraging presumed whitening constraints to eliminate shared feedback coefficients.

In existing predictive coding-based schemes such as [21, 22, 23], the loss function contains only forward prediction error terms. The feedback connection with symmetric weights, which backpropagates forward prediction error, emerges due to the gradient-based optimization of the PC loss. In contrast, our framework's crucial contribution is the adoption of two alternative expressions for the correlative mutual information between consecutive network layers as the central normative approach. Utilizing these two alternatives naturally leads to both forward and backward prediction paths with asymmetric weights, promoting information flow in both feedforward and top-down directions. Unlike the work of [10], our method circumvents the need for layer whitening constraints and additional forward prediction terms to achieve asymmetric weights.

1.1.3 Correlative information maximization

Information maximization has been proposed as a governing or guiding principle in several machine learning and neuroscience frameworks for different tasks: (i) The propagation of information within a self-organized network as pioneered by Linsker [30]. (ii) Extracting hidden features or factors associated with observations by maximizing information between the input and its internal representation such as independent component analysis (ICA-InfoMax) approach by [31]. In the neuroscience

domain, the motivation has been to provide normative explanations to the behaviour of cortical activities evidenced by experimental work, such as orientation and visual stimuli length selectivity of primary visual cortex neurons [32, 33]. The same idea has been recently extended in the machine learning field by the Deep Infomax approach where the goal is to transfer maximum information from the input of a deep network to its final layer, while satisfying prior distribution constraints on the output representations [34]. (iii) Matching representations corresponding to two alternative augmentations or modalities of the same input in the context of self-supervised learning [35].

Correlative mutual information maximization has been recently proposed as an alternative for Shannon Mutual Information (SMI), due to its desirable properties [36]: (i) maximization of CMI is equivalent to maximizing linear dependence, which may be more relevant than establishing arbitrary nonlinear dependence in certain applications [37], (ii) it is based only on the second order statistics, making it relatively easier to optimize. We additionally note that criteria based on correlation are intrinsically linked to local learning rules, leading to biologically plausible implementations, [38, 39]. Erdogan [36] proposed the use of CorInfoMax for solving blind source separation (BSS) problem to retrieve potentially correlated components from their mixtures. Ozsoy et al. [37] proposed maximizing the CMI between the representations of two different augmentations of the same input as a self-supervised learning approach. More recently, Bozkurt et al. [11] introduced an unsupervised framework to generate biologically plausible neural networks for the BSS problem with infinitely many domain selections using the CMI objective.

In this article, we suggest employing the CorInfoMax principle for biologically plausible supervised learning. The key difference compared to the unsupervised framework presented in [11] is the utilization of two alternative forms of mutual information. This leads to a bidirectional information flow that enables error backpropagation without encountering the weight symmetry issue.

2 Deep correlative information maximization

2.1 Network data model

We assume a dataset with L input data points $\mathbf{x}[t] \in \mathbb{R}^m, t = 1, \ldots, L$, and let $\mathbf{y}_T[t] \in \mathbb{R}^n$ be the corresponding labels. We consider a network with $P-1$ hidden layers whose activities are denoted by $\mathbf{r}^{(k)} \in \mathbb{R}^{N_k}, k = 1, \ldots, P-1$. For notational simplicity, we also denote input and output of the network by $\mathbf{r}^{(0)}$ and $\mathbf{r}^{(P)}$, i.e., $\mathbf{r}^{(0)}[t] = \mathbf{x}[t]$ and $\mathbf{r}^{(P)}[t] = \hat{\mathbf{y}}[t]$. We consider polytopic constraints for the hidden and output layer activities, i.e., $\mathbf{r}^{(k)} \in \mathcal{P}^{(k)}$, where $\mathcal{P}^{(k)}$ is the presumed polytopic domain for the k-th layer [11, 40]. We note that the polytopic assumptions are plausible as the activations of neurons in practice are bounded. In particular, we will make the specific assumption that $\mathcal{P}^{(k)} = \mathcal{B}_{\infty,+} = \{\mathbf{r} : \mathbf{0} \preccurlyeq \mathbf{r} \preccurlyeq \mathbf{1}\}$, i.e., (normalized) activations lie in a nonnegative unit-hypercube. Such nonnegativity constraints have been connected to disentangling behavior [41, 42, 43], however, we consider extensions in the form of alternative polytopic sets corresponding to different feature priors [11] (see Appendix C). More broadly, the corresponding label $\mathbf{y}_T$ can be, one-hot encoded label vectors for a classification problem, or discrete or continuous valued vectors for a regression problem.

2.2 Correlative information maximization based signal propagation

Our proposed CorInfoMax framework represents a principled approach where both the structure of the network and its internal dynamics as well as the learning rules governing adaptation of its parameters are not predetermined. Instead, these elements emerge naturally from an explicit optimization process. As the optimization objective, we propose the maximization of *correlative mutual information* (see Appendix A between two consecutive network layers. As derived in future sections, the proposed objective facilitates information flow—input-to-output and vice versa, while the presumed domains for the hidden and output layers inherently induce information compression and feature shaping.

In Sections 2.2.1 and 2.2.2, we outline the correlative mutual information-based objective and its implementation based on samples, respectively. Section 2.3 demonstrates that the optimization of this objective through gradient ascent naturally results in recurrent neural networks with multi-compartment neurons. Finally, Section 2.4 explains how the optimization of the same criterion leads to biologically plausible learning dynamics for the resulting network structure.

2.2.1 Stochastic CorInfoMax based supervised criterion

We propose the total correlative mutual information among consecutive layers, augmented with the mean-square-error (MSE) training loss, as the stochastic objective to be maximized:

$$J(\mathbf{r}^{(1)},\ldots,\mathbf{r}^{(P)}) = \sum_{k=0}^{P-1} I^{(\epsilon_k)}(\mathbf{r}^{(k)},\mathbf{r}^{(k+1)}) - \frac{\beta}{2}E(\|\mathbf{y}_T - \mathbf{r}^{(P)}\|_2^2), \tag{1}$$

where, as defined in [36, 37] and in Appendix A,

$$\overrightarrow{I}^{(\epsilon_k)}(\mathbf{r}^{(k)},\mathbf{r}^{(k+1)}) = \frac{1}{2}\log\det\left(\mathbf{R}_{\mathbf{r}^{(k+1)}} + \epsilon_k \boldsymbol{I}\right) - \frac{1}{2}\log\det\left(\boldsymbol{R}_{\overrightarrow{\mathbf{e}}_*^{(k+1)}} + \epsilon_k \boldsymbol{I}\right), \tag{2}$$

is the correlative mutual information between layers $\mathbf{r}^{(k)}$ an $\mathbf{r}^{(k+1)}$, $\mathbf{R}_{\mathbf{r}^{(k+1)}} = E(\mathbf{r}^{(k+1)}{\mathbf{r}^{(k+1)}}^T)$ is the autocorrelation matrix corresponding to the layer $\mathbf{r}^{(k+1)}$ activations, and $\mathbf{R}_{\overrightarrow{\mathbf{e}}_*^{(k+1)}}$ corresponds to the error autocorrelation matrix for the best linear regularized minimum MSE predictor of $\mathbf{r}^{(k+1)}$ from $\mathbf{r}^{(k)}$. Therefore, the mutual information objective in (2) makes a referral to the *regularized* **forward** *prediction problem* represented by the optimization

$$\underset{\boldsymbol{W}_{ff}^{(k)}}{\text{minimize}}\; E(\|\overrightarrow{\mathbf{e}}^{(k+1)}\|_2^2) + \epsilon_k\|\boldsymbol{W}_{ff}^{(k)}\|_F^2 \quad \text{s.t.} \quad \overrightarrow{\mathbf{e}}^{(k+1)} = \mathbf{r}^{(k+1)} - \boldsymbol{W}_{ff}^{(k)}\mathbf{r}^{(k)}, \tag{3}$$

and $\mathbf{e}_*^{(k+1)}$ is the forward prediction error corresponding to the optimal forward predictor $\boldsymbol{W}_{ff,*}^{(k)}$.

If we interpret the maximization of CMI in (2): the first term on the right side of (2) encourages the spread of $\mathbf{r}^{(k+1)}$ in its presumed domain $\mathcal{P}^{(k+1)}$, while the second term incites the minimization of redundancy in $\mathbf{r}^{(k+1)}$ beyond its component predictable from $\mathbf{r}^{(k)}$.

An equal and alternative expression for the CMI can be written as (Appendix A)

$$\overleftarrow{I}^{(\epsilon_k)}(\mathbf{r}^{(k)},\mathbf{r}^{(k+1)}) = \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{r}^{(k)}} + \epsilon_k \boldsymbol{I}) - \frac{1}{2}\log\det\left(\boldsymbol{R}_{\overleftarrow{\mathbf{e}}_*^{(k)}} + \epsilon_k \boldsymbol{I}\right), \tag{4}$$

where $\boldsymbol{R}_{\overleftarrow{\mathbf{e}}_*^{(k)}}$ corresponds to the error autocorrelation matrix for the best linear regularized minimum MSE predictor of $\mathbf{r}^{(k)}$ from $\mathbf{r}^{(k+1)}$. The corresponding *regularized* **backward** *prediction problem* is defined by the optimization

$$\underset{\boldsymbol{W}_{fb}^{(k)}}{\text{minimize}}\; E(\|\overleftarrow{\mathbf{e}}^{(k)}\|_2^2) + \epsilon_k\|\boldsymbol{W}_{fb}^{(k)}\|_F^2 \quad \text{s.t.} \quad \overleftarrow{\mathbf{e}}^{(k)} = \mathbf{r}^{(k)} - \boldsymbol{W}_{fb}^{(k)}\mathbf{r}^{(k+1)}. \tag{5}$$

We observe that the two alternative yet equivalent representations of the correlative mutual information between layers $\mathbf{r}^{(k)}$ and $\mathbf{r}^{(k+1)}$ in (2) and (4) are intrinsically linked to the forward and backward prediction problems between these layers, which are represented by the optimizations in (3) and (5), respectively. As we will demonstrate later, the existence of these two alternative forms for the CMI plays a crucial role in deriving a neural network architecture that overcomes the weight symmetry issue.

2.2.2 Sample-based supervised CorInfoMax criterion

Our aim is to construct a biologically plausible neural network that optimizes the total CMI, equation (1), in an adaptive manner. Here, we obtain a sample-based version of (1) as a step towards that goal.

We first define the exponentially-weighted sample auto and cross-correlation matrices as follows:

$$\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] = \frac{1-\lambda_{\mathbf{r}}}{1-\lambda_{\mathbf{r}}^t}\sum_{i=1}^{t}\lambda_{\mathbf{r}}^{t-i}\mathbf{r}^{(k)}[i]{\mathbf{r}^{(k)}[i]}^T,\; \hat{\mathbf{R}}_{\mathbf{r}^{(k)}\mathbf{r}^{(k+1)}}[t] = \frac{1-\lambda_{\mathbf{r}}}{1-\lambda_{\mathbf{r}}^t}\sum_{i=1}^{t}\lambda_{\mathbf{r}}^{t-i}\mathbf{r}^{(k)}[i]{\mathbf{r}^{(k+1)}[i]}^T, \tag{6}$$

for $k = 0,\ldots,P$, respectively, where $0 \ll \lambda_{\mathbf{r}} < 1$ is the forgetting factor. Next, we define two equivalent forms of the sample-based CMI, $\hat{I}^{(\epsilon)}(\mathbf{r}^{(k)},\mathbf{r}^{(k+1)})[t]$:

$$\hat{\overrightarrow{I}}^{(\epsilon_k)}(\mathbf{r}^{(k)},\mathbf{r}^{(k+1)})[t] = \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k+1)}}[t] + \epsilon_k \boldsymbol{I}) - \frac{1}{2}\log\det(\hat{\boldsymbol{R}}_{\overrightarrow{\mathbf{e}}_*^{(k+1)}}[t] + \epsilon_k \boldsymbol{I}), \tag{7}$$

$$\hat{\overleftarrow{I}}^{(\epsilon_k)}(\mathbf{r}^{(k)},\mathbf{r}^{(k+1)})[t] = \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_k \boldsymbol{I}) - \frac{1}{2}\log\det(\hat{\boldsymbol{R}}_{\overleftarrow{\mathbf{e}}_*^{(k)}}[t] + \epsilon_k \boldsymbol{I}), \tag{8}$$

where $\hat{\boldsymbol{R}}_{\overrightarrow{\mathbf{e}}_*^{(k+1)}}[t]$ is the exponentially-weighted sample autocorrelation matrix for the forward prediction error at level-$(k+1)$, $\overrightarrow{\mathbf{e}}^{(k+1)_*}[t]$, corresponding to the best linear exponentially-weighted regularized least squares predictor of $\mathbf{r}^{(k+1)}[t]$ from the lower level activations $\mathbf{r}^{(k)}[t]$. Similarly, $\hat{\boldsymbol{R}}_{\overleftarrow{\mathbf{e}}^{(k)}}[t]$ is the exponentially-weighted autocorrelation matrix for the backward prediction error at level-(k), $\overleftarrow{\mathbf{e}}^{(k)}[t]$, corresponding to the best linear exponentially-weighted regularized least squares predictor of $\mathbf{r}^{(k)}[t]$ from the higher level activations $\mathbf{r}^{(k+1)}[t]$.

The sample-based CorInfoMax optimization can be written as:

$$\underset{\mathbf{r}^{(k)}[t], k=0,\ldots,P}{\text{maximize}} \quad \sum_{k=0}^{P-1} \hat{I}^{(\epsilon_k)}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})[t] - \frac{\beta}{2}\|\mathbf{y}_T[t] - \mathbf{r}^{(P)}[t]\|_2^2 = \hat{J}(\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(P)})[t] \tag{9a}$$

$$\text{subject to} \qquad \mathbf{r}^{(k)}[t] \in \mathcal{P}^{(k)}, k = 1, \ldots, P, \tag{9b}$$

$$\mathbf{r}^{(0)}[t] = \mathbf{x}[t], \tag{9c}$$

As outlined in Appendix B, we can employ Taylor series linearization to approximate the $\log\det$ terms associated with forward and backward prediction errors in (2) and (4) in the form

$$\begin{aligned}
&\log\det\left(\hat{\boldsymbol{R}}_{\overrightarrow{\mathbf{e}}^{(k+1)}}[t] + \epsilon_k \boldsymbol{I}\right) \\
&\approx \frac{1}{\epsilon_k}\sum_{i=1}^{t} \lambda_{\mathbf{r}}^{t-i}\|\mathbf{r}^{(k+1)}[i] - \mathbf{W}_{ff,*}^{(k)}[t]\mathbf{r}^{(k)}[i]\|_2^2 + \epsilon_k\|\boldsymbol{W}_{ff,*}^{(k)}[t]\|_F^2 + N_{k+1}\log(\epsilon_k) \qquad (10)\\
&\log\det\left(\hat{\boldsymbol{R}}_{\overleftarrow{\mathbf{e}}^{(k)}}[t] + \epsilon_k \boldsymbol{I}\right) \\
&\approx \frac{1}{\epsilon_k}\sum_{i=1}^{t} \lambda_{\mathbf{r}}^{t-i}\|\mathbf{r}^{(k)}[i] - \mathbf{W}_{fb,*}^{(k)}[t]\mathbf{r}^{(k+1)}[i]\|_2^2 + \epsilon_k\|\boldsymbol{W}_{fb,*}^{(k)}[t]\|_F^2 + N_k\log(\epsilon_k), \qquad (11)
\end{aligned}$$

where $\mathbf{W}_{ff,*}^{(k)}[t]$ is the optimal linear regularized weighted least squares forward predictor coefficients in predicting $\mathbf{r}^{(k+1)}[i]$ from $\mathbf{r}^{(k)}[i]$ for $i = 1, \ldots, t$, and $\mathbf{W}_{fb,*}^{(k)}[t]$ is the optimal linear regularized weighted least squares backward predictor coefficients in predicting $\mathbf{r}^{(k)}[i]$ from $\mathbf{r}^{(k+1)}[i]$ for $i = 1, \ldots, t$. Consequently, the optimal choices of forward and backward predictor coefficients are coupled with the optimal choices of layer activations.

In the online optimization process, we initially relax the requirement on the optimality of predictors and start with random predictor coefficient selections. During the learning process, we apply a coordinate ascent-based procedure on activation signals and predictor coefficients. Specifically, at time step-t, we consider two phases:

1. First, we optimize with respect to the activations $\{\mathbf{r}^{(k)}[t], k = 1, \ldots, P\}$, where we assume predictor coefficients to be fixed. This phase yields network structure and output dynamics,
2. Next, we update the forward and backward predictor coefficients $\boldsymbol{W}_{ff}^{(k)}$ and $\boldsymbol{W}_{fb}^{(k)}$, for $k = 1, \ldots, P$, to reduce the corresponding forward and backward prediction errors, respectively. This phase provides update expressions to be utilized in learning dynamics.

As the algorithm iterations progress, the predictor coefficients converge to the vicinity of their optimal values.

For the first phase of the online optimization, we employ a projected gradient ascent-based approach for activations: for $k = 1, \ldots, P-1$, the layer activation vector $\mathbf{r}^{(k)}[t]$ is included in the objective function terms $\hat{I}^{(\epsilon)}(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k)})[t]$ and $\hat{I}^{(\epsilon)}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})[t]$. Therefore, to calculate the gradient with respect to $\mathbf{r}^{(k)}[t]$, we can use expressions in (7) and (8). More specifically, we choose $\hat{J}_k(\mathbf{r}^{(k)})[t] = \hat{I}^{(\overrightarrow{\epsilon}_{k-1})}(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k)})[t] + \hat{I}^{(\overleftarrow{\epsilon}_k)}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})[t]$ for $k = 1, \ldots, P-1$, to represent the components of the objective function in (9a) involving $\mathbf{r}^{(k)}[t]$. As described in Appendix G.1, this choice is instrumental in avoiding weight transport problem. Similarly, we can write the component of

the objective function in (9a) that is dependent on the final layer activations as $\hat{J}_P(\mathbf{r}^{(P)})[t] = \hat{I}^{(\epsilon_{P-1})}(\mathbf{r}^{(P-1)}, \mathbf{r}^{(P)})[t] - \frac{\beta}{2}\|\mathbf{r}^{(P)}[t] - \mathbf{y}_T[t]\|_2^2$.

Based on the derivations presented in Appendix D, which directly incorporate the approximations from (10) and (11), we can express the gradient of the objective function in (9a) with respect to $\mathbf{r}^{(k)}$, for $k = 1, \ldots, P-1$ as:

$$\nabla_{\mathbf{r}^{(k)}} \hat{J}(\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(P)})[t] = 2\gamma \boldsymbol{B}_{\mathbf{r}^{(k)}}[t]\mathbf{r}^{(k)}[t] - \frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}^{(k)}[t] - \frac{1}{\epsilon_k}\overleftarrow{\mathbf{e}}^{(k)}[t], \tag{12}$$

where $\gamma = \frac{1-\lambda_{\mathbf{r}}}{\lambda_{\mathbf{r}}}$,

$$\overrightarrow{\mathbf{e}}^{(k)}[t] = \mathbf{r}^{(k)}[t] - \boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t], \quad \overleftarrow{\mathbf{e}}^{(k)}[t] = \mathbf{r}^{(k)}[t] - \boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)}[t], \tag{13}$$

and $\boldsymbol{B}_{\mathbf{r}^{(k)}}[t] = (\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_{k-1}\boldsymbol{I})^{-1} \approx (\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_k \boldsymbol{I})^{-1}$. Similarly, for $k = P$, we have

$$\nabla_{\mathbf{r}^{(P)}} \hat{J}(\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(P)})[t] = \gamma \boldsymbol{B}_{\mathbf{r}^{(P)}}[t]\mathbf{r}^{(P)}[t] - \frac{1}{\epsilon_{P-1}}\overrightarrow{\mathbf{e}}^{(P)}[t] - \beta(\mathbf{r}^{(P)}[t] - \boldsymbol{y}_T[t]). \tag{14}$$

2.3 Neural network formulation based on information maximization

In this section, we develop a biologically plausible neural network grounded on the correlative information maximization-based network propagation model outlined in Section 2.2. To achieve this, we employ projected gradient ascent optimization for determining layer activations $\mathbf{r}^{(1)}[t], \mathbf{r}^{(2)}[t], \ldots, \mathbf{r}^{(P)}[t]$, which shape the network structure and dynamics, as well as updating the corresponding synapses that govern the learning dynamics.

2.3.1 Network structure and neural dynamics

In this section, we show that the projected gradient ascent solution to the optimization in (9) defines a multilayer recurrent neural network. To this end, we introduce the intermediate variable $\mathbf{u}^{(k)}$ as the updated layer-k activations prior to the projection onto the domain set $\mathcal{P}^{(k)}$. Utilizing the gradient expressions in (12)-(13), we can express the network dynamics for layers $k = 1, \ldots, P-1$ as follows (see Appendix E for details):

$$\tau_{\mathbf{u}}\frac{d\mathbf{u}^{(k)}[t;s]}{ds} = -g_{lk}\mathbf{u}^{(k)}[t;s] + \frac{1}{\epsilon_k}\boldsymbol{M}^{(k)}[t]\boldsymbol{r}^{(k)}[t;s] - \frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}_u^{(k)}[t;s] - \frac{1}{\epsilon_k}\overleftarrow{\mathbf{e}}_u^{(k)}[t;s], \tag{15}$$

$$\overrightarrow{\mathbf{e}}_u^{(k)}[t;s] = \mathbf{u}^{(k)}[t;s] - \boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t;s], \quad \overleftarrow{\mathbf{e}}_u^{(k)}[t;s] = \mathbf{u}^{(k)}[t;s] - \boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)}[t;s], \tag{16}$$

$$\mathbf{r}^{(k)}[t;s] = \sigma_+(\mathbf{u}^{(k)}[t;s]), \tag{17}$$

where t is the discrete data index, s is the continuous time index corresponding to network dynamics, $\tau_{\mathbf{u}}$ is the update time constant, $\boldsymbol{M}^{(k)}[t] = \epsilon_k(2\gamma \boldsymbol{B}_{\mathbf{r}^{(k)}}[t] + g_{lk}\boldsymbol{I})$, and σ_+ represents the elementwise clipped-ReLU function corresponding to the projection onto the nonnegative unit-hypercube $\mathcal{B}_{\infty,+}$, defined as $\sigma_+(u) = \min(1, \max(u, 0))$.

To reinterpret the dynamics in (15) to (17) as a multi-compartmental neural network, for $k = 1, \ldots, P-1$, we define the signals:

$$\mathbf{v}_A^{(k)}[t;s] = \boldsymbol{M}^{(k)}[t]\boldsymbol{r}^{(k)}[t;s] + \boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)}[t;s], \quad \mathbf{v}_B^{(k)}[t;s] = \boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t;s], \tag{18}$$

which allow us to rewrite the network activation dynamics (15) to (17) as:

$$\tau_{\mathbf{u}}\frac{d\mathbf{u}^{(k)}[t;s]}{ds} = -g_{lk}\mathbf{u}^{(k)}[t;s] + g_{A,k}(\mathbf{v}_A^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s]) + g_{B,k}(\mathbf{v}_B^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s]), \tag{19}$$

$$\mathbf{r}^{(k)}[t;s] = \sigma_+(\mathbf{u}^{(k)}[t;s]), \tag{20}$$

where $g_{A,k} = \frac{1}{\epsilon_{k-1}}$ and $g_{B,k} = \frac{1}{\epsilon_k}$. Similarly, for the output layer, we employ the same expressions as (19) and (20) with $k = P$, except that in this case we have:

$$\mathbf{v}_A^{(P)}[t;s] = \boldsymbol{M}^{(P)}[t]\boldsymbol{r}^{(k)}[t;s] - (\mathbf{r}^{(P)}[t;s] - \boldsymbol{y}_T[t]), \quad \mathbf{v}_B^{(P)}[t;s] = \boldsymbol{W}_{ff}^{(P-1)}[t]\mathbf{r}^{(P-1)}[t;s], \tag{21}$$

where $g_{B,P} = \frac{1}{\epsilon_{P-1}}$, $g_{A,P} = \beta$ and $\boldsymbol{M}^{(P)}[t] = \beta^{-1}(\gamma \boldsymbol{B}_{\mathbf{r}^{(P)}}[t] + g_{lk}\boldsymbol{I})$.

Remarkably, the equations (18) to (21) reveal a biologically plausible neural network that incorporates three-compartment pyramidal neuron models, as presented in [9, 10]. This intricate architecture, of which two-layer segment is demonstrated in Figure 1, naturally emerges from the proposed correlative information maximization framework. In this network structure:

- $\mathbf{u}^{(k)}$ embodies the membrane potentials for neuronal somatic compartments of the neurons at layer-k, where $\tau_{\mathbf{u}}$ is the membrane leak time constant of soma.
- $\mathbf{v}_B^{(k)}$ corresponds to membrane potentials for basal dendrite compartments, receiving feedforward input originating from the previous layer.
- $\mathbf{v}_A^{(k)}$ denotes the membrane potentials for distal apical dendrite compartments, which gather top-down input from the subsequent layer and lateral inputs represented by $\boldsymbol{M}^{(k)}[t]\boldsymbol{r}^{(k)}$ in (18) and (21). Decomposing $\boldsymbol{M}^{(k)}$ into $\boldsymbol{D}^{(k)} - \boldsymbol{O}^{(k)}$, we find that $\boldsymbol{D}^{(k)}$ mirrors autapses[44], and the off-diagonal component $\boldsymbol{O}^{(k)}$ corresponds to lateral inhibition synapses. We use $\mathbf{i}^{(k)} = -\boldsymbol{O}^{(k)}\mathbf{r}^{(k)}$ to represent the activations of SST interneurons [14] that generate lateral inhibitions to the apical dendrites.
- Forward (backward) prediction errors manifest in the membrane voltage differences between soma and basal (distal) compartments of the pyramidal neurons.
- Forward (backward) prediction coefficients $\boldsymbol{W}_{ff}^{(k)}$ ($\boldsymbol{W}_{fb}^{(k)}$) are associated with feedforward (top-down) synapses connecting layers (k) and $(k+1)$.
- The inverse of the regularization coefficient ϵ_k is related to the conductance between soma and dendritic compartments. This is compliant with the interpretation of the ϵ^{-1} in Appendix A.2 as the sensitivity parameter that determines the contribution of the prediction errors to the CMI. Conversely, at the output layer, the augmentation constant β corresponds to the conductance between soma and distal compartments. This relationship can be motivated by modifying the objective in (9a) as

$$\sum_{k=0}^{P-1} \hat{I}^{(\epsilon_k)}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})[t] + \frac{1}{2}\overleftarrow{\hat{I}^{(\beta^{-1})}}(\mathbf{r}^{(P)}, \mathbf{y}_T)[t], \tag{22}$$

  where, through the first-order approximation, the $\mathbf{r}^{(P)}[t]$ dependent portion of $\overleftarrow{\hat{I}^{(\beta^{-1})}}(\mathbf{r}^{(P)}, \mathbf{y}_T)[t]$ can be expressed as $-\beta\|\mathbf{r}^{(P)}[t] - \boldsymbol{W}_{fb}^{(P)}\mathbf{y}_T[t]\|_2^2$. For accuracy, we enforce $\boldsymbol{W}_{fb}^{(P)} = \boldsymbol{I}$.

2.4 Learning dynamics

Network parameters consists of feedforward $\boldsymbol{W}_{ff}^{(k)}$, feedback $\boldsymbol{W}_{fb}^{(k)}$ and lateral $\boldsymbol{B}^{(k)}$ coefficients.The learning dynamics of these coefficients are elaborated below:

- *Feedforward Coefficients* are connected to the forward prediction problem defined by the optimization in (3). We can define the corresponding online optimization objective function as $C_{ff}(\boldsymbol{W}_{ff}^{(k)}) = \epsilon_k\|\boldsymbol{W}_{ff}^{(k)}\|_F^2 + \|\overrightarrow{\mathbf{e}}^{(k+1)}[t]\|_2^2$ for which the the partial derivative is given by

$$\frac{\partial C_{ff}(\boldsymbol{W}_{ff}^{(k)}[t])}{\partial \boldsymbol{W}_{ff}^{(k)}} = 2\epsilon_k \boldsymbol{W}_{ff}^{(k)}[t] - 2\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T. \tag{23}$$

  In Appendix H, we provide a discussion on rewriting (23) in terms of the membrane voltage difference between the distal apical and soma compartments of the neuron, based on the equilibrium condition for the neuronal dynamics:

$$-\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T = g_{B,k}^{-1}(g_{A,k}\mathbf{v}_A^{(k)}[t] - (g_{lk} + g_{A_k})\mathbf{u}_*^{(k)}[t] + \mathbf{h}_*[t])\mathbf{r}^{(k)}[t]^T, \tag{24}$$

  where $\mathbf{h}_*[t]$ is nonzero only for neurons that are silent or firing at the maximum rate.
- Similarly, *Feedback Coefficients* are connected to the backward prediction problem defined by the optimization in (5), and the corresponding online optimization objective function as $C_{fb}(\boldsymbol{W}_{fb}^{(k)}) = \epsilon_k\|\boldsymbol{W}_{ff}^{(k)}\|_F^2 + \|\overleftarrow{\mathbf{e}}^{(k)}[t]\|_2^2$ for which the partial derivative is given by

$$\frac{\partial C_{fb}(\boldsymbol{W}_{fb}^{(k)}[t])}{\partial \boldsymbol{W}_{fb}^{(k)}} = 2\epsilon_k \boldsymbol{W}_{fb}^{(k)}[t] - 2\overleftarrow{\mathbf{e}}^{(k)}[t]\mathbf{r}^{(k+1)}[t]^T. \tag{25}$$

To compute the updates of both feedforward and feedback coefficients, we use the EP approach [24], where the update terms are obtained based on the contrastive expressions of partial derivatives in (23) and (25) for the nudge phase, i.e., $\beta = \beta' > 0$, and the free phase, i.e., $\beta = 0$, :

$$\delta \boldsymbol{W}_{ff}^{(k)}[t] \propto \frac{1}{\beta'} \left((\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T)\Big|_{\beta=\beta'} - (\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T)\Big|_{\beta=0} \right), \tag{26}$$

$$\delta \boldsymbol{W}_{fb}^{(k)}[t] \propto \frac{1}{\beta'} \left((\overleftarrow{\mathbf{e}}^{(k)}[t]\mathbf{r}^{(k+1)}[t]^T)\Big|_{\beta=\beta'} - (\overleftarrow{\mathbf{e}}^{(k)}[t]\mathbf{r}^{(k+1)}[t]^T)\Big|_{\beta=0} \right). \tag{27}$$

- *Lateral Coefficients*, $\mathbf{B}^{(k)}$ are the inverses of the $\epsilon\mathbf{I}$ perturbed correlation matrices. We can use the update rule in [11] for their learning dynamics after the nudge phase:

$$\boldsymbol{B}^{(k)}[t+1] = \lambda_{\mathbf{r}}^{-1}(\boldsymbol{B}^{(k)}[t] - \gamma \mathbf{z}^{(k)}[t]\mathbf{z}^{(k)}[t]^T), \text{ where } \mathbf{z}^{(k)} = \boldsymbol{B}^{(k)}[t]\mathbf{r}^{(k)}[t]\big|_{\beta=\beta'}. \tag{28}$$

As we derived in Appendix F, we can rewrite the update rule of the lateral weights in terms of the updates of autapses and lateral inhibition synapses as follows:

$$\mathbf{D}_{ii}^{(k)}[t+1] = \lambda_{\mathbf{r}}^{-1}\mathbf{D}_{ii}^{(k)}[t] - \lambda_{\mathbf{r}}^{-1}\epsilon_k 2\gamma^2(\mathbf{z}_i^{(k)}[t])^2 + \epsilon_k g_{lk}(1-\lambda_{\mathbf{r}}^{-1}), \quad \forall i \in \{1,\dots,N_k\} \tag{29}$$

$$\mathbf{O}_{ij}^{(k)}[t+1] = \lambda_{\mathbf{r}}^{-1}\mathbf{O}^{(k)}[t]_{ij} + \lambda_{\mathbf{r}}^{-1}\epsilon_k 2\gamma^2 \mathbf{z}_i^{(k)}[t]\mathbf{z}_j^{(k)}[t], \ \ \forall i,j \in \{1,\dots,N_k\}, \text{ where } i \neq j \tag{30}$$

3 Discussion of results

- In (A.14), we devise an update for layer activation $\mathbf{r}^{(k)}$ by employing two distinct forms of the CMI associated with $\mathbf{r}^{(k)}$: $\hat{I}^{(\epsilon_{k-1})}(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k)})[t]$, the CMI with the preceding layer, encompassing the forward prediction error for estimating $\mathbf{r}^{(k)}$, and $\hat{I}^{(\epsilon_k)}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})[t]$, the CMI with the subsequent layer, incorporating the backward prediction error for estimating $\mathbf{r}^{(k)}$. Employing these alternative expressions is crucial in circumventing the weight transport problem and offering a more biologically plausible framework. For further discussion, please refer to Appendix G.
- In the context of the proposed correlative information maximization framework, forward and backward predictive coding naturally emerges as a crucial mechanism. By incorporating both alternative expressions of CMI, the framework focuses on minimizing both forward and backward prediction errors between adjacent layers via feedforward and feedback connections. These connections foster bidirectional information flow, thereby enhancing the overall learning process.
- Figure 1 depicts the interplay between the CorInfoMax objective and the corresponding network architecture. The emergence of lateral connections and autapses can be attributed to the maximization of the unconditional layer entropy component of the CMI, which allows for efficient utilization of the available representation dimensions and avoids dimensional degeneracy. Simultaneously, the minimization of conditional entropies between adjacent layers gives rise to feedforward and feedback connections, effectively reducing redundancy within representations.
- We employ time-contrastive learning, as in GenRec [45], EP [24] and CSM [27], by implementing separate phases with Hebbian and anti-Hebbian updates, governed by an assumed teaching signal. It has been conjectured that the teaching signal in biological networks can be modeled by the oscillations in the brain [2, 46, 47]. Although the oscillatory rhythms and their synchronization in the brain are elusive, they are believed to play an important role in adaptive processes such as learning and predicting upcoming events [48, 49].

4 Numerical experiments

In this section, we evaluate the performance of our CorInfoMax framework with two layer fully connected networks on image classification tasks using three popular datasets: MNIST [50], Fashion-MNIST [51], and CIFAR10 [52]. We used layer sizes of $784, 500, 10$ for both MNIST and Fashion-MNIST datasets while we used layer sizes of $3072, 1000, 10$ for CIFAR10 dataset, and the final layer size 10 corresponds to one-hot encoded ouput vectors. Further details including full set of hyperparameters can be found in Appendix J. We compare the effectiveness of our approach against

other contrastive methods, such as EP [24] and CSM [27], as well as explicit methods, including PC [22] and PC-Nudge [53], when training multilayer perceptron (MLP) architectures.

We examine two distinct constraints on the activations of CorInfoMax Networks: (i) $\mathcal{B}_{\infty,+}$, representing the nonnegative part of the unit hypercube, and (ii) $\mathcal{B}_{1,+} = \{\mathbf{r} : \mathbf{r} \succcurlyeq 0, \|\mathbf{r}\|_1 \leq 1\}$, denoting the nonnegative part of the unit ℓ_1-norm ball [40]. Table 1 presents the test accuracy results for each algorithm, averaged over 10 realizations along with the corresponding standard deviations. These findings demonstrate that CorInfoMax networks can achieve comparable or superior performance in relation to the state-of-the-art methods for the selected tasks. Additional information regarding these experiments, as well as further experiments, can be found in the Appendix. Our code is available online[1].

Table 1: Test accuracy results (mean $\pm$ standard deviation from $n = 10$ runs) for CorInfoMax networks are compared with other biologically-plausible algorithms. The performance of CSM on the CIFAR10 dataset is taken from [27], while the remaining results stem from our own simulations.

	MNIST	FashionMNIST	CIFAR10
CorInfoMax-$\mathcal{B}_{\infty,+}$ (Appendix J.4)	97.62 ± 0.1	88.14 ± 0.3	51.86 ± 0.3
CorInfoMax-$\mathcal{B}_{1,+}$ (Appendix J.6)	97.71 ± 0.1	88.09 ± 0.1	51.19 ± 0.4
EP	97.61 ± 0.1	88.06 ± 0.7	49.28 ± 0.5
CSM	98.08 ± 0.1	88.73 ± 0.2	40.79^*
PC	98.17 ± 0.2	89.31 ± 0.4	-
PC-Nudge	97.71 ± 0.1	88.49 ± 0.3	48.58 ± 0.7
Feedback Alignment (with MSE Loss)	97.99 ± 0.03	88.72 ± 0.5	50.75 ± 0.4
Feedback Alignment (with CrossEntropy Loss)	97.95 ± 0.08	88.38 ± 0.9	52.37 ± 0.4
BP (with MSE Loss)	97.58 ± 0.01	88.39 ± 0.1	52.75 ± 0.1
BP (with CrossEntropy Loss)	98.27 ± 0.03	89.41 ± 0.2	53.96 ± 0.3

5 Discussion and Conclusion

In this article, we have presented the correlative information maximization (CorInfoMax) framework as a biologically plausible approach to constructing supervised neural network models. Our proposed method addresses the long-standing weight symmetry issue by providing a principled solution, which results in asymmetric forward and backward prediction networks. The experimental analyses demonstrates that CorInfoMax networks provide better or on-par performance in image classification tasks compared to other biologically plausible networks while alleviating the weight symmetry problem. Furthermore, the CorInfoMax framework offers a normative approach for developing network models that incorporate multi-compartment pyramidal neuron models, aligning more closely with the experimental findings about the biological neural networks. The proposed framework is useful in obtaining potential insights such as the role of lateral connections in embedding space expansion and avoiding degeneracy, feedback and feedforward connections for prediction to reduce redundancy, and activation functions/interneurons to shape feature space and compress. Despite the emphasis on supervised deep neural networks in our work, it's crucial to highlight that our approach—replacing the backpropagation algorithm, which suffers from the weight transportation problem, with a normative method devoid of such issues—is potentially extendable to unsupervised and self-supervised learning contexts.

One potential limitation of our framework, shared by other supervised approaches, is the necessity for model parameter search to improve accuracy. We discuss this issue in detail in Appendix K. Another limitation stems from the intrinsic nature of our approach, which involves the determination of neural activities through recursive dynamics (see Appendix J). While this aspect is fundamental to our methodology, it does result in slower computation times compared to conventional neural networks in digital hardware implementation. However, it is worth noting that our proposed network, characterized by local learning rules, holds the potential for efficient and low-power implementations on future neuromorphic hardware chips. Furthermore, our method employs the time contrastive learning technique known as Equilibrium Propagation, which necessitates two distinct phases for learning.

[1] https://github.com/BariscanBozkurt/Supervised-CorInfoMax

6 Acknowledgments and Disclosure of Funding

This research was supported by KUIS AI Center Research Award. B. Bozkurt acknowledges the support by Gatsby PhD programme, which is supported by the Gatsby Charitable Foundation (GAT3850). C. Pehlevan is supported by NSF Award DMS-2134157, NSF CAREER Award IIS-2239780, and a Sloan Research Fellowship. This work has been made possible in part by a gift from the Chan Zuckerberg Initiative Foundation to establish the Kempner Institute for the Study of Natural and Artificial Intelligence.

References

[1] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[2] James CR Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 23(3):235–250, 2019.

[3] Francis Crick. The recent excitement about neural networks. *Nature*, 337:129–132, 1989.

[4] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.

[5] Xiaohui Xie and H Sebastian Seung. Equivalence of backpropagation and contrastive hebbian learning in a layered network. *Neural computation*, 15(2):441–454, 2003.

[6] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

[7] Matthew Larkum. A cellular mechanism for cortical associations: an organizing principle for the cerebral cortex. *Trends in neurosciences*, 36(3):141–151, 2013.

[8] Robert Urbanczik and Walter Senn. Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3):521–528, 2014.

[9] João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31, 2018.

[10] Siavash Golkar, Tiberiu Tesileanu, Yanis Bahroun, Anirvan Sengupta, and Dmitri Chklovskii. Constrained predictive coding as a biologically plausible model of the cortical hierarchy. *Advances in Neural Information Processing Systems*, 35:14155–14169, 2022.

[11] Bariscan Bozkurt, Ateş İsfendiyaroğlu, Cengiz Pehlevan, and Alper Tunga Erdogan. Correlative information maximization based biologically plausible neural networks for correlated source separation. In *The Eleventh International Conference on Learning Representations*, 2023.

[12] Leopoldo Petreanu, Tianyi Mao, Scott M Sternson, and Karel Svoboda. The subcellular organization of neocortical excitatory connections. *Nature*, 457(7233):1142–1145, 2009.

[13] Blake A Richards and Timothy P Lillicrap. Dendritic solutions to the credit assignment problem. *Current opinion in neurobiology*, 54:28–36, 2019.

[14] Joanna Urban-Ciecko and Alison L Barth. Somatostatin-expressing neurons in cortical networks. *Nature Reviews Neuroscience*, 17(7):401–409, 2016.

[15] Jordan Guerguiev, Timothy P Lillicrap, and Blake A Richards. Towards deep learning with segregated dendrites. *Elife*, 6:e22901, 2017.

[16] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cogn. Sci.*, 11:23–63, 1987.

[17] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276, 2016.

[18] Mohamed Akrout, Collin Wilson, Peter Humphreys, Timothy Lillicrap, and Douglas B Tweed. Deep learning without weight transport. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[19] Yali Amit. Deep learning with asymmetric connections and hebbian updates. *Frontiers in computational neuroscience*, 13:18, 2019.

[20] Qianli Liao, Joel Leibo, and Tomaso Poggio. How important is weight symmetry in backpropagation? *Proceedings of the AAAI Conference on Artificial Intelligence*, 30, 10 2015.

[21] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

[22] James CR Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262, 2017.

[23] Yuhang Song, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Can the brain do backpropagation?—exact implementation of backpropagation in predictive coding networks. *Advances in neural information processing systems*, 33:22566–22579, 2020.

[24] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, 11, 2017.

[25] Axel Laborieux, Maxence Ernoult, Benjamin Scellier, Yoshua Bengio, Julie Grollier, and Damien Querlioz. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in Neuroscience*, 15:633674, 02 2021.

[26] Axel Laborieux and Friedemann Zenke. Holomorphic equilibrium propagation computes exact gradients through finite size oscillations. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[27] Shanshan Qin, Nayantara Mudur, and Cengiz Pehlevan. Contrastive similarity matching for supervised learning. *Neural computation*, 33(5):1300–1328, 2021.

[28] Benjamin Scellier, Anirudh Goyal, Jonathan Binas, Thomas Mesnard, and Yoshua Bengio. Generalization of equilibrium propagation to vector field dynamics, 2018.

[29] J.F. Kolen and J.B. Pollack. Backpropagation without weight transport. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 3, pages 1375–1380 vol.3, 1994.

[30] Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

[31] Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

[32] D.H. Hubel and T.N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148:574–591, 1959.

[33] Anthony J Bell and Terrence J Sejnowski. The "independent components" of natural scenes are edge filters. *Vision research*, 37(23):3327–3338, 1997.

[34] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *International Conference on Learning Representations (ICLR)*, 2019.

[35] Suzanna Becker and Geoffrey E Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, 1992.

[36] Alper T Erdogan. An information maximization based blind source separation approach for dependent and independent sources. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4378–4382, 2022.

[37] Serdar Ozsoy, Shadi Hamdan, Sercan O Arik, Deniz Yuret, and Alper Tunga Erdogan. Self-supervised learning with an information maximization criterion. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[38] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15:267–273, 1982.

[39] David Lipshutz, Yanis Bahroun, Siavash Golkar, Anirvan M Sengupta, and Dmitri B Chklovskii. A biologically plausible neural network for multichannel canonical correlation analysis. *Neural Computation*, 33(9):2309–2352, 2021.

[40] Gokcan Tatli and Alper T Erdogan. Polytopic matrix factorization: Determinant maximization based criterion and identifiability. *IEEE Transactions on Signal Processing*, 69:5431–5447, 2021.

[41] Mark D Plumbley. Algorithms for nonnegative independent component analysis. *IEEE Transactions on Neural Networks*, 14(3):534–543, 2003.

[42] Cengiz Pehlevan, Sreyas Mohan, and Dmitri B Chklovskii. Blind nonnegative source separation using biological neural networks. *Neural computation*, 29(11):2925–2954, 2017.

[43] James C. R. Whittington, Will Dorrell, Surya Ganguli, and Timothy Behrens. Disentanglement with biological constraints: A theory of functional cell types. In *The Eleventh International Conference on Learning Representations*, 2023.

[44] Joachim Lübke, Henry Markram, Michael Frotscher, and Bert Sakmann. Frequency and dendritic distribution of autapses established by layer 5 pyramidal neurons in the developing rat neocortex: comparison with synaptic innervation of adjacent neurons of the same class. *Journal of Neuroscience*, 16(10):3209–3218, 1996.

[45] Randall C O'Reilly. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural computation*, 8(5):895–938, 1996.

[46] Pierre Baldi and Fernando Pineda. Contrastive learning and neural oscillations. *Neural Computation*, 3(4):526–545, 1991.

[47] Nicholas Ketz, Srinimisha G Morkonda, and Randall C O'Reilly. Theta coordinated error-driven learning in the hippocampus. *PLoS computational biology*, 9(6):e1003067, 2013.

[48] Juergen Fell and Nikolai Axmacher. The role of phase synchronization in memory processes. *Nature reviews. Neuroscience*, 12:105–18, 02 2011.

[49] Andreas Engel, Pascal Fries, and Wolf Singer. Dynamic predictions: oscillations and synchrony in top-down processing. *Nature reviews. Neuroscience*, 2:704–716, 11 2001.

[50] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

[51] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[52] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[53] Beren Millidge, Yuhang Song, Tommaso Salvatori, Thomas Lukasiewicz, and Rafal Bogacz. Backpropagation at the infinitesimal inference limit of energy-based models: Unifying predictive coding, equilibrium propagation, and contrastive hebbian learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[54] Zhanghao Zhouyin and Ding Liu. Understanding neural networks with logarithm determinant entropy estimator. *arXiv preprint arXiv:1401.3420*, 2021.

[55] Thomas Kailath, Ali H Sayed, and Babak Hassibi. *Linear estimation*. Prentice-Hall information and system sciences series. Prentice Hall, 2000.

[56] Xiao Fu, Kejun Huang, Bo Yang, Wing-Kin Ma, and Nicholas D Sidiropoulos. Robust volume minimization-based matrix factorization for remote sensing and document clustering. *IEEE Transactions on Signal Processing*, 64(23):6254–6268, 2016.

[57] Arne Brondsted. *An Introduction to Convex Polytopes*, volume 90. Springer Science & Business Media, 2012.

[58] David L Donoho. For most large underdetermined systems of equations, the minimal ℓ_1-norm near-solution approximates the sparsest near-solution. *Communications on Pure and Applied Mathematics*, 59(7):907–934, July 2006.

[59] Michael Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Science & Business Media, 2010.

[60] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine learning*, pages 272–279, July 2008.

[61] Eren Babatas and Alper T Erdogan. An algorithmic framework for sparse bounded component analysis. *IEEE Transactions on Signal Processing*, 66(19):5194–5205, August 2018.

[62] Christoph Studer, Tom Goldstein, Wotao Yin, and Richard G Baraniuk. Democratic representations. *arXiv preprint arXiv:1401.3420*, 2014.

[63] Alper T Erdogan. A class of bounded component analysis algorithms for the separation of both independent and dependent sources. *IEEE Transactions on Signal Processing*, 61(22):5730–5743, August 2013.

[64] Huseyin A Inan and Alper T Erdogan. A convolutive bounded component analysis framework for potentially nonstationary independent and/or dependent sources. *IEEE Transactions on Signal Processing*, 63(1):18–30, November 2014.

[65] Bariscan Bozkurt, Cengiz Pehlevan, and Alper Tunga Erdogan. Biologically-plausible determinant maximization neural networks for blind separation of correlated sources. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[66] Christopher J Rozell, Don H Johnson, Richard G Baraniuk, and Bruno A Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, 20(10):2526–2563, 2008.

[67] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[68] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and trends® in Optimization*, 1(3):127–239, 2014.

[69] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Appendix

A Preliminaries on correlative entropy and mutual information

In this section, we present the essential background on correlative entropy and information measures that will be employed in our proposed approach.

A.1 Correlative entropy and correlative mutual information

Consider a random vector $\mathbf{x} \in \mathbb{R}^m$ with a correlation matrix $\mathbf{R}_\mathbf{x} = E(\mathbf{x}\mathbf{x}^T)$. The correlative entropy for this random vector $\mathbf{x}$ is defined as follows [36, 54]:

$$H^{(\epsilon)}(\mathbf{x}) = \frac{1}{2}\log\det(\mathbf{R}_\mathbf{x} + \epsilon\boldsymbol{I}) + \frac{m}{2}\log(2\pi e), \tag{A.1}$$

where $\epsilon > 0$ is a small positive constant. Please note that (A.1) is an adapted version of Shannon's differential entropy for Gaussian vectors. However, we utilize it as an entropy definition based on second-order statistics, independent of the distribution of the vector $\mathbf{x}$. Furthermore, the joint correlative entropy of two random vectors $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^n$ can be expressed as:

$$\begin{aligned} H^{(\epsilon)}(\mathbf{x},\mathbf{y}) &= \frac{1}{2}\log\det(\mathbf{R}_{\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}} + \epsilon\boldsymbol{I}) + \frac{m+n}{2}\log\det(2\pi e) \\ &= \frac{1}{2}\log\det\left(\begin{bmatrix}\mathbf{R}_\mathbf{x} + \epsilon\boldsymbol{I} & \mathbf{R}_\mathbf{xy} \\ \mathbf{R}_\mathbf{yx} & \mathbf{R}_\mathbf{y} + \epsilon\boldsymbol{I}\end{bmatrix}\right) + \frac{m+n}{2}\log\det(2\pi e) \\ &= \frac{1}{2}\log\left(\det(\mathbf{R}_\mathbf{x} + \epsilon\boldsymbol{I})\det(\mathbf{R}_\mathbf{y} + \epsilon\boldsymbol{I} - \mathbf{R}_\mathbf{xy}^T(\mathbf{R}_{\boldsymbol{x}} + \epsilon\boldsymbol{I})^{-1}\mathbf{R}_\mathbf{xy})\right) \\ &+ \frac{m+n}{2}\log\det(2\pi e) \\ &= \frac{1}{2}\log\det(\mathbf{R}_\mathbf{x} + \epsilon\boldsymbol{I}) + \frac{m}{2}\log(2\pi e) + \frac{1}{2}\log\det(\mathbf{R}_\mathbf{e} + \epsilon\boldsymbol{I}) + \frac{n}{2}\log(2\pi e), \end{aligned} \tag{A.2}$$

where $\mathbf{R}_\mathbf{xy} = E(\mathbf{x}\mathbf{y}^T)$, and $\mathbf{R}_\mathbf{e} = \mathbf{R}_\mathbf{y} - \mathbf{R}_\mathbf{xy}^T(\mathbf{R}_{\boldsymbol{x}} + \epsilon\boldsymbol{I})^{-1}\mathbf{R}_\mathbf{xy}$ is the autocorrelation matrix of the error vector corresponding to the best linear minimum mean square estimator (MMSE) of $\mathbf{y}$ from $\mathbf{x}$, and the third equality is obtained by writing the determinant in terms of a principle submatrix and its Schur's complement [55]. This derivation naturally leads to the expression of $H^{(\epsilon)}(\mathbf{x},\mathbf{y}) = H^{(\epsilon)}(\mathbf{x}) + H^{(\epsilon)}(\mathbf{y}|_L\mathbf{x})$. Note that we use the notation $|_L$ to distinguish that $H^{(\epsilon)}(\mathbf{x}|\mathbf{y})$ requires to use $\mathbf{R}_{\mathbf{x}|\mathbf{y}}$ that is not equal to $\mathbf{R}_\mathbf{x} - \mathbf{R}_\mathbf{yx}^T(\mathbf{R}_{\boldsymbol{y}} + \epsilon\boldsymbol{I})^{-1}\mathbf{R}_\mathbf{yx}$ in general. Therefore, we can express the conditional correlative entropy definitions as [36]

$$\begin{aligned} H^{(\epsilon)}(\mathbf{y}|_L\mathbf{x}) &= \frac{1}{2}\log\det(\mathbf{R}_\mathbf{y} - \mathbf{R}_\mathbf{xy}^T(\mathbf{R}_{\boldsymbol{x}} + \epsilon\boldsymbol{I})^{-1}\mathbf{R}_\mathbf{xy} + \epsilon\boldsymbol{I}) + \frac{n}{2}\log(2\pi e) \\ H^{(\epsilon)}(\mathbf{x}|_L\mathbf{y}) &= \frac{1}{2}\log\det(\mathbf{R}_\mathbf{x} - \mathbf{R}_\mathbf{yx}^T(\mathbf{R}_{\boldsymbol{y}} + \epsilon\boldsymbol{I})^{-1}\mathbf{R}_\mathbf{yx} + \epsilon\boldsymbol{I}) + \frac{m}{2}\log(2\pi e). \end{aligned}$$

Using the alternative Schur's complement in (A.2), it can be also shown that $H^{(\epsilon)}(\mathbf{x},\mathbf{y}) = H^{(\epsilon)}(\mathbf{y}) + H^{(\epsilon)}(\mathbf{x}|_L\mathbf{y})$. Based on these definitions, the correlative mutual information (CMI) is defined as follows

$$\begin{aligned} I^{(\epsilon)}(\mathbf{x},\mathbf{y}) &= H^{(\epsilon)}(\mathbf{y}) - H^{(\epsilon)}(\mathbf{y}|_L\mathbf{x}), \\ &= H^{(\epsilon)}(\mathbf{x}) - H^{(\epsilon)}(\mathbf{x}|_L\mathbf{y}), \\ &= H^{(\epsilon)}(\mathbf{x}) + H^{(\epsilon)}(\mathbf{y}) - H^{(\epsilon)}(\mathbf{x},\mathbf{y}), \end{aligned} \tag{A.3}$$

More explicitly, we can write the correlative mutual information using the following two alternative yet equivalent expressions,

$$\begin{aligned} I^{(\epsilon)}(\mathbf{x},\mathbf{y}) &= \frac{1}{2}\log\det(\mathbf{R}_\mathbf{x} + \epsilon\boldsymbol{I}) - \frac{1}{2}\log\det(\mathbf{R}_\mathbf{x} - \mathbf{R}_\mathbf{yx}^T(\mathbf{R}_{\boldsymbol{y}} + \epsilon\boldsymbol{I})^{-1}\mathbf{R}_\mathbf{yx} + \epsilon\boldsymbol{I}), \\ &= \frac{1}{2}\log\det(\mathbf{R}_\mathbf{y} + \epsilon\boldsymbol{I}) - \frac{1}{2}\log\det(\mathbf{R}_\mathbf{y} - \mathbf{R}_\mathbf{xy}^T(\mathbf{R}_{\boldsymbol{x}} + \epsilon\boldsymbol{I})^{-1}\mathbf{R}_\mathbf{xy} + \epsilon\boldsymbol{I}). \end{aligned}$$

At its core, the correlative mutual information, $I^{(\epsilon)}(\mathbf{x},\mathbf{y})$, quantifies the degree of correlation or linear association between the random vectors $\mathbf{x}$ and $\mathbf{y}$ [36].

A.2 On the interpretation of ϵ parameter

The definition of correlative entropy as presented in equation (A.1) includes a perturbation term, ϵ, in the eigenvalues of the correlation matrix argument of the log-determinant. At first glance, this term appears to function as a correction factor to compensate for rank-deficient correlation matrices of degenerate random vectors. From this perspective, this adjustment serves two primary purposes:

i. To establish a finite lower bound for the entropy, and
ii. To circumvent numerical optimization issues, given that the derivative of the $\log\det$ function is the inverse of its argument.

In fact, robust matrix factorization methods that rely on determinant-maximization often use the perturbation term $\epsilon\mathbf{I}$ for these reasons, as suggested by Fu et al. [56].

Beyond these, upon examining the expression for correlative mutual information more closely,

$$I^{(\epsilon)}(\mathbf{x},\mathbf{y}) = \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}} + \epsilon\boldsymbol{I}) - \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{e}_{\mathbf{x}|\mathbf{y}}} + \epsilon\boldsymbol{I}), \tag{A.4}$$

we can draw the following insights:

- The matrix $\mathbf{R}_{\mathbf{e}_{\mathbf{x}|\mathbf{y}}} = \mathbf{R}_{\mathbf{x}} - \mathbf{R}_{\mathbf{yx}}^T(\mathbf{R}_{\boldsymbol{y}} + \epsilon\boldsymbol{I})^{-1}\mathbf{R}_{\mathbf{yx}}$ corresponds to the error correlation matrix for the best linear regularized minimum mean square estimator of $\boldsymbol{x}$ from $\boldsymbol{y}$. This estimator is obtained as the solution of the optimization problem

$$\underset{\boldsymbol{W}_{\mathbf{x}|\mathbf{y}}}{\text{minimize}}\; E(\|\mathbf{x} - \boldsymbol{W}_{\mathbf{x}|\mathbf{y}}\mathbf{y}\|_2^2) + \epsilon\|\boldsymbol{W}_{\mathbf{x}|\mathbf{y}}\|_F^2. \tag{A.5}$$

  In this context, ϵ parameter acts as a regularizing coefficient for the linear estimation problem integral to measuring linear dependence between the two arguments of the CMI.
- Maximizing the CMI given by equation (A.4) can be accomplished by increasing the correlative entropy of $\mathbf{x}$ while decreasing the correlative entropy of the estimation error $\mathbf{e}_{\mathbf{x}|\mathbf{y}}$. Given the relationship $\boldsymbol{R}_{\mathbf{x}} \succeq \boldsymbol{R}_{\mathbf{e}_{\mathbf{x}|\mathbf{y}}}$, we anticipate that the choice of ϵ will primarily influence the correlative entropy of $\mathbf{e}_{\mathbf{x}|\mathbf{y}}$. Indeed, since ϵ is added to all the eigenvalues of $\boldsymbol{R}_{\mathbf{e}_{\mathbf{x}|\mathbf{y}}}$, reducing its eigenvalues below ϵ would have only incremental increase in the mutual information. As such, a smallerϵ value will place greater emphasis on reducing the estimation error $\mathbf{e}_{\mathbf{x}|\mathbf{y}}$. Consequently, one can consider ϵ^{-1} can be viewed as an indicator of the sensitivity of the CMI to the levels of estimation error $\mathbf{e}_{\mathbf{x}|\mathbf{y}}$, determining how far we need to push down the estimation error values to increase the CMI.
- The role of ϵ parameter in adjusting sensitivity to estimation errors becomes evident when we linearize the $\boldsymbol{R}_{\mathbf{e}_{\mathbf{x}|\mathbf{y}}}$ dependent (second) term in (A.4) using the truncated Taylor series approximation equation (A.8) in Appendix B (by choosing $\mathbf{A} = \epsilon\boldsymbol{I}$ and $\boldsymbol{\Delta} = \boldsymbol{R}_{\mathbf{e}_{\mathbf{x}|\mathbf{y}}}$ in (A.8)):

$$I^{(\epsilon)}(\mathbf{x},\mathbf{y}) \approx \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}}) - \frac{\epsilon^{-1}}{2}\text{Tr}(\mathbf{R}_{\mathbf{e}_{\mathbf{x}|\mathbf{y}}}) + \frac{m}{2}\log(\epsilon). \tag{A.6}$$

  In the above expression, we have assumed that the choice of ϵ is less than the eigenvalues of $\mathbf{R}_{\mathbf{x}}$, so that we can approximate $\mathbf{R}_{\mathbf{x}} + \epsilon\boldsymbol{I} \approx \mathbf{R}_{\mathbf{x}}$, and greater than the eigenvalues of $\mathbf{R}_{\mathbf{e}_{\mathbf{x}|\mathbf{y}}}$. As apparent from equation (A.6), ϵ^{-1} serves as a sensitivity parameter that determines the contribution of estimation errors $\mathbf{e}_{\mathbf{x}|\mathbf{y}}$ to the CMI.

B Linear approximation of correlative entropy

In this section, we provide linear approximation of the $\log\det$ terms on the rightmost terms of (7) and (8). For this purpose we utilize the the first-order Taylor series approximation of the $\log\det$ function:

$$\log\det(\mathbf{A}+\boldsymbol{\Delta}) \approx \log\det(\mathbf{A}) + \text{Tr}(\nabla_{\mathbf{A}}\log\det(\mathbf{A})^T\boldsymbol{\Delta}) \tag{A.7}$$

$$\approx \log\det(\mathbf{A}) + \text{Tr}(\mathbf{A}^{-1}\boldsymbol{\Delta}), \tag{A.8}$$

assuming $\mathbf{A}$ is Hermitian (or real symmetric). Therefore, using (A.8) and choosing $\mathbf{A} = \epsilon_k\boldsymbol{I}$ and $\boldsymbol{\Delta} = \hat{\boldsymbol{R}}_{\overrightarrow{\mathbf{e}}^{(k+1)}}[t]$, we can approximate the rightmost term of (7) corresponding to the correlative

entropy of forward error $\overrightarrow{\mathbf{e}}^{(k+1)}$

$$\log\det\left(\hat{\boldsymbol{R}}_{\overrightarrow{\mathbf{e}}^{(k+1)}}[t]+\epsilon_k\boldsymbol{I}\right)\approx\frac{1}{\epsilon_k}\operatorname{Tr}\left(\hat{\boldsymbol{R}}_{\overrightarrow{\mathbf{e}}^{(k+1)}}[t]\right)+N_{k+1}\log(\epsilon_k)$$

$$=\frac{1}{\epsilon_k}\sum_{i=1}^{t}\lambda_{\mathbf{r}}^{t-i}\|\mathbf{r}^{(k+1)}[i]-\mathbf{W}_{ff,*}^{(k)}[t]\mathbf{r}^{(k)}[i]\|_2^2+\epsilon_k\|\boldsymbol{W}_{ff,*}^{(k)}[t]\|_F^2+N_{k+1}\log(\epsilon_k). \tag{A.9}$$

This approximation would be more accurate for prediction error correlation matrices with smaller eigenvalues.

Similarly, using (A.8) and choosing $\mathbf{A}=\epsilon_k\boldsymbol{I}$ and $\boldsymbol{\Delta}=\hat{\boldsymbol{R}}_{\overrightarrow{\mathbf{e}}^{(k)}}[t]$, we can approximate the rightmost term of (8) corresponding to the correlative entropy of forward error $\overleftarrow{\mathbf{e}}^{(k)}$

$$\log\det\left(\hat{\boldsymbol{R}}_{\overleftarrow{\mathbf{e}}^{(k)}}[t]+\epsilon_k\boldsymbol{I}\right)\approx\frac{1}{\epsilon_k}\operatorname{Tr}\left(\hat{\boldsymbol{R}}_{\overleftarrow{\mathbf{e}}^{(k)}}[t]\right)+N_k\log(\epsilon_k)$$

$$=\frac{1}{\epsilon_k}\sum_{i=1}^{t}\lambda_{\mathbf{r}}^{t-i}\|\mathbf{r}^{(k)}[i]-\mathbf{W}_{fb,*}^{(k)}[t]\mathbf{r}^{(k+1)}[i]\|_2^2+\epsilon_k\|\boldsymbol{W}_{fb,*}^{(k)}[t]\|_F^2+N_k\log(\epsilon_k). \tag{A.10}$$

Note that in (A.9), $\mathbf{W}_{ff,*}^{(k)}[t]$ denotes the optimal linear regularized weighted least squares forward predictor coefficients in predicting $\mathbf{r}^{(k+1)}[i]$ from $\mathbf{r}^{(k)}[i]$ for $i=1,\ldots,t$. Likewise, $\mathbf{W}_{fb,*}^{(k)}[t]$ in (A.10) represents the optimal linear regularized weighted least squares backward predictor coefficients in predicting $\mathbf{r}^{(k)}[i]$ from $\mathbf{r}^{(k+1)}[i]$ for $i=1,\ldots,t$.

C Background on polytopic representations

Convex polytopes, compact sets formed by the intersections of halfspaces [57], serve as constraint domains for latent representations. The choice of a particular polytope reflects the attribute assignments to these vectors. For instance, the ℓ_1-norm-ball polytope enforces sparsity and finds extensive use in machine learning, signal processing and computational neuroscience [58, 59, 60, 61]. Conversely, the ℓ_∞-norm-ball polytope is prevalent in antisparse (democratic) representations, especially in bounded component analysis applications [62, 63, 64].

The Polytopic Matrix Factorization (PMF) extends these examples to an infinite set of polytopes, incorporating specific symmetry restrictions for identifiability [40]. In the PMF paradigm, observation vectors $\{\mathbf{y}_1,\ldots,\mathbf{y}_N\}\subset\mathbb{R}^M$ are modeled as unknown linear transformations of latent vectors $\{\mathbf{s}_1,\ldots,\mathbf{s}_N\}$ from a selected polytope $\mathcal{P}$:

$$\underbrace{\begin{bmatrix}\mathbf{y}_1 & \mathbf{y}_2 & \ldots & \mathbf{y}_N\end{bmatrix}}_{\mathbf{Y}}=\mathbf{H}\underbrace{\begin{bmatrix}\mathbf{s}_1 & \mathbf{s}_2 & \ldots & \mathbf{s}_N\end{bmatrix}}_{\mathbf{S}}, \tag{A.11}$$

where $\mathbf{H}$ denotes the unknown linear transformation. The PMF's primary objective is to deduce the factors $\mathbf{H}$ and $\mathbf{S}$ from the observation matrix $\mathbf{Y}$, by exploiting the information that the columns of $\mathbf{S}$ as "representative" samples from $\mathcal{P}$. The shape of the chosen polytope determine the latent vector features by combining common attributes such as sparsity, nonnegativity and anti-sparsity in subvector level. For example, the reference [11] proposes the canonical polytope description

$$\mathcal{P}=\left\{\boldsymbol{s}\in\mathbb{R}^n\ \middle|\ s_i\in[-1,1]\,\forall i\in\mathcal{I}_s,\ s_i\in[0,1]\,\forall i\in\mathcal{I}_+,\ \|\boldsymbol{s}_{\mathcal{J}_l}\|_1\leq 1,\ \mathcal{J}_l\subseteq\mathbb{Z}_n^+,\ l\in\mathbb{Z}_L^+\right\},$$

where L is the number of mutually sparse subvector constraints, $\mathcal{I}_+$ is the index set of nonnegative elements, $\mathcal{I}_s$ is the index set of signed elements, the sets $\mathcal{J}_l, l\in\mathbb{Z}_L^+$ are the index sets for the sparse subvectors. The availability of infinitely many polytope options provides a powerful and diverse toolkit for representing and characterizing latent vectors.

The online correlative information maximization solution for obtaining factors in (A.11), combined with the polytopic constraints on the columns of $\mathbf{S}$ results in biologically plausible neural networks with local learning rules [65]. These polytopic constraints manifest as piecewise linear neural activation functions, such as ReLU and clipping functions.

In a vein similar to the unsupervised problem in [65], our proposed framework employs polytopic representations to characterize embedding vectors for each network layer. The inclusion of polytopic constraints influences:

- The characterization of embeddings based on assigned attributes.
- The network's nonlinear component via piecewise activation functions.

In summary, polytopic representations offer a versatile and mathematically rigorous framework for modeling latent vectors in various applications, from machine learning to signal processing. Their ability to encapsulate diverse attributes, such as sparsity and non-negativity, provides a rich characterization of latent features.

D Gradient derivation for the CorInfoMax objective

In this section, we offer a derivation of the gradients for the CorInfoMax objective function $\hat{J}(\mathbf{r}^{(1)},\ldots,\mathbf{r}^{(P)})$ in (9a) with respect to the layer activation vector $\mathbf{r}^{(k)}$. As outlined in Section 2.2.2, the components of $\hat{J}(\mathbf{r}^{(1)},\ldots,\mathbf{r}^{(P)})$ containing $\mathbf{r}^{(k)}$ can be expressed as:

$$\hat{J}_k(\mathbf{r}^{(k)})[t] = \hat{I}^{(\overrightarrow{\epsilon_{k-1}})}(\mathbf{r}^{(k-1)},\mathbf{r}^{(k)})[t] + \hat{I}^{(\overleftarrow{\epsilon_k})}(\mathbf{r}^{(k)},\mathbf{r}^{(k+1)})[t], \tag{A.12}$$

for $k=1,\ldots,P$, and

$$\hat{J}_P(\mathbf{r}^{(P)})[t] = \hat{I}^{(\overrightarrow{\epsilon_{P-1}})}(\mathbf{r}^{(P-1)},\mathbf{r}^{(P)})[t] - \frac{\beta}{2}\|\mathbf{r}^{(P)}[t]-\boldsymbol{y}_T[t]\|_2^2. \tag{A.13}$$

To simplify our derivations, as explained in Appendix B, we replace the correlative entropy terms for the prediction errors (the rightmost terms of (7) and (8)) with their linear approximations in (A.9)-(A.10). Thus, the resulting gradient with respect to $\mathbf{r}^{(k)}[t]$ can be expressed as:

$$\begin{aligned}
\nabla_{\mathbf{r}^{(k)}}\hat{J}(\mathbf{r}^{(1)},\ldots,\mathbf{r}^{(P)})[t] &= \nabla_{\mathbf{r}^{(k)}}\hat{J}_k(\mathbf{r}^{(k)})[t] \\
&= \nabla_{\mathbf{r}^{(k)}}\hat{I}^{(\overrightarrow{\epsilon_{k-1}})}(\mathbf{r}^{(k-1)},\mathbf{r}^{(k)})[t] + \nabla_{\mathbf{r}^{(k)}}\hat{I}^{(\overleftarrow{\epsilon_k})}(\mathbf{r}^{(k)},\mathbf{r}^{(k+1)})[t] \\
&= \frac{1}{2}\nabla_{\mathbf{r}^{(k)}}(\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t]+\epsilon_{k-1}\boldsymbol{I}) + \log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t]+\epsilon_k\boldsymbol{I})) \\
&\quad -\frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}^{(k)}[t] - \frac{1}{\epsilon_k}\overleftarrow{\mathbf{e}}^{(k)}[t],
\end{aligned} \tag{A.14}$$

where

$$\overrightarrow{\mathbf{e}}^{(k)}[t] = \mathbf{r}^{(k)}[t] - \boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t], \quad \overleftarrow{\mathbf{e}}^{(k)}[t] = \mathbf{r}^{(k)}[t] - \boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)}[t] \tag{A.15}$$

are the forward and backward prediction errors at level-k, based on the current estimates of the corresponding predictor matrices. Following the procedure detailed in [11], for the gradient term in (A.14), we obtain:

$$\frac{1}{2}\nabla_{\mathbf{r}^{(k)}}(\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t]+\epsilon_{k-1}\boldsymbol{I}) + \log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t]+\epsilon_k\boldsymbol{I})) = 2\gamma\boldsymbol{B}_{\mathbf{r}^{(k)}}[t]\mathbf{r}^{(k)}[t], \tag{A.16}$$

where $\boldsymbol{B}\mathbf{r}^{(k)}[t] = (\hat{\mathbf{R}}\mathbf{r}^{(k)}[t]+\epsilon_{k-1}\boldsymbol{I})^{-1} \approx (\hat{\mathbf{R}}\mathbf{r}^{(k)}[t]+\epsilon_k\boldsymbol{I})^{-1}$ and $\gamma = \frac{1-\lambda_{\mathbf{r}}}{\lambda_{\mathbf{r}}}$. The gradient of the objective for the final layer can be written as:

$$\nabla_{\mathbf{r}^{(P)}}\hat{J}_P(\mathbf{r}^{(P)})[t] = \gamma\boldsymbol{B}_{\mathbf{r}^{(P)}}[t]\mathbf{r}^{(P)}[t] - \frac{1}{\epsilon_{P-1}}\overrightarrow{\mathbf{e}}^{(P)}[t] - \beta(\mathbf{r}^{(P)}[t]-\boldsymbol{y}_T[t]). \tag{A.17}$$

In conclusion, combining expressions (A.18)-(A.17), we can formulate:

$$\nabla_{\mathbf{r}^{(k)}}\hat{J}(\mathbf{r}^{(1)},\ldots,\mathbf{r}^{(P)})[t] = 2\gamma\boldsymbol{B}_{\mathbf{r}^{(k)}}[t]\mathbf{r}^{(k)}[t] - \frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}^{(k)}[t] - \frac{1}{\epsilon_k}\overleftarrow{\mathbf{e}}^{(k)}[t], \tag{A.18}$$

for $k=1,\ldots,P-1$, and

$$\nabla_{\mathbf{r}^{(P)}}\hat{J}(\mathbf{r}^{(1)},\ldots,\mathbf{r}^{(P)})[t] = \gamma\boldsymbol{B}_{\mathbf{r}^{(P)}}[t]\mathbf{r}^{(P)}[t] - \frac{1}{\epsilon_{P-1}}\overrightarrow{\mathbf{e}}^{(P)}[t] - \beta(\mathbf{r}^{(P)}[t]-\boldsymbol{y}_T[t]). \tag{A.19}$$

E Derivation of CorInfoMax network dynamics

In this section, we provide details about the derivation of the CorInfoMax network dynamics equations (15)-(17) in Section 2.3.1.

As discussed in Section 2.2.2, network dynamics naturally emerge from the application of the projected gradient ascent algorithm to obtain solution of the optimization problem (9). Therefore, we update layer activations $\mathbf{r}^{(k)}[t]$ using $\nabla_{\mathbf{r}^{(k)}}\hat{J}(\mathbf{r}^{(1)},\dots,\mathbf{r}^{(P)})$ and then project the result to the presumed domain $\mathcal{P}^{(k)}=\mathcal{B}_{\infty,+}$.

Following the approach in [66], we define pre-projection signal $\mathbf{u}^{(k)}$, which is updated with gradient, and then project it onto $\mathcal{P}^{(k)}=\mathcal{B}_{\infty}$. We use continuous dynamic update in the form

$$\tau_u\frac{d\mathbf{u}^{(k)}[t;s]}{ds}=\nabla_{\mathbf{r}}^{(k)}\hat{J}(\mathbf{r}^{(1)},\dots,\mathbf{r}^{(P)})[t;s] \tag{A.20}$$

$$\mathbf{r}^{(k)}[t;s]=\sigma_+(\mathbf{u}^{(k)}[t;s]). \tag{A.21}$$

In above equations, t is the discrete data index, referring to the index of input ($\mathbf{x}[t]$) and label ($\mathbf{y}_T[t]$) samples, as defined in Section 2.1. Whereas, s is the continuous time index corresponding to network dynamics. Furthermore, σ_+ represents the elementwise clipped-ReLU function corresponding to the projection onto the nonnegative unit-hypercube $\mathcal{B}_{\infty,+}$, defined as $\sigma_+(u)=\min(1,\max(u,0))$.

We can plug in the gradient expression in (12) in (A.20) to obtain

$$\tau_u\frac{d\mathbf{u}^{(k)}[t;s]}{ds}=2\gamma\boldsymbol{B}_{\mathbf{r}^{(k)}}[t]\mathbf{r}^{(k)}[t;s]-\frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}^{(k)}[t;s]-\frac{1}{\epsilon_k}\overleftarrow{\mathbf{e}}^{(k)}[t;s] \tag{A.22}$$

$$\overrightarrow{\mathbf{e}}_u^{(k)}[t;s]=\mathbf{u}^{(k)}[t;s]-\boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t;s],\quad \overleftarrow{\mathbf{e}}_u^{(k)}[t;s]=\mathbf{u}^{(k)}[t;s]-\boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)} \tag{A.23}$$

$$\mathbf{r}^{(k)}[t;s]=\sigma_+(\mathbf{u}^{(k)}[t;s]). \tag{A.24}$$

In order to covert neural dynamics in (A.22)-(A.24) to leaky-integrate-and-fire form, we add leaky term $-g_{lk}\mathbf{u}^{(k)}[t;s]$ and modify the output feedback form to compensate this addition:

$$\tau_u\frac{d\mathbf{u}^{(k)}[t;s]}{ds}=-g_{lk}\mathbf{u}^{(k)}[t;s]+(2\gamma\boldsymbol{B}_{\mathbf{r}^{(k)}}[t]+g_{lk}\boldsymbol{I})\mathbf{r}^{(k)}[t]-\frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}^{(k)}[t]-\frac{1}{\epsilon_k}\overleftarrow{\mathbf{e}}^{(k)}[t] \tag{A.25}$$

$$\overrightarrow{\mathbf{e}}_u^{(k)}[t;s]=\mathbf{u}^{(k)}[t;s]-\boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t;s],\quad \overleftarrow{\mathbf{e}}_u^{(k)}[t;s]=\mathbf{u}^{(k)}[t;s]-\boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)} \tag{A.26}$$

$$\mathbf{r}^{(k)}[t;s]=\sigma_+(\mathbf{u}^{(k)}[t;s]). \tag{A.27}$$

Finally, substituting $\boldsymbol{M}^{(k)}[t]=\epsilon_k(2\gamma\boldsymbol{B}_{\mathbf{r}^{(k)}}[t]+g_{lk}\boldsymbol{I})$, we can rewrite the neural dynamics as

$$\tau_u\frac{d\mathbf{u}^{(k)}[t;s]}{ds}=-g_{lk}\mathbf{u}^{(k)}[t;s]+\frac{1}{\epsilon_k}\boldsymbol{M}^{(k)}[t]\mathbf{r}^{(k)}[t]-\frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}^{(k)}[t]-\frac{1}{\epsilon_k}\overleftarrow{\mathbf{e}}^{(k)}[t] \tag{A.28}$$

$$\overrightarrow{\mathbf{e}}_u^{(k)}[t;s]=\mathbf{u}^{(k)}[t;s]-\boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t;s],\quad \overleftarrow{\mathbf{e}}_u^{(k)}[t;s]=\mathbf{u}^{(k)}[t;s]-\boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)} \tag{A.29}$$

$$\mathbf{r}^{(k)}[t;s]=\sigma_+(\mathbf{u}^{(k)}[t;s]). \tag{A.30}$$

F Derivation of lateral weight updates in terms of autopses and lateral inhibition synapses

Here, we provide the derivation of the lateral weight updates provided in (29) and (30). Recall that in Section 2.3.1, we defined $\boldsymbol{M}^{(k)}[t]=\epsilon_k(2\gamma\boldsymbol{B}_{\mathbf{r}^{(k)}}[t]+g_{lk}\boldsymbol{I})$. Therefore, using (28), we can write:

$$\mathbf{M}^{(k)}[t+1]=\epsilon_k(2\gamma\lambda_{\mathbf{r}}^{-1}(\mathbf{B}^{(k)}[t]-\gamma\mathbf{z}^{(k)}[t]\mathbf{z}^{(k)}[t]^T)+g_{lk}\mathbf{I}) \tag{A.31}$$

$$=\lambda_{\mathbf{r}}^{-1}\epsilon_k 2\gamma\mathbf{B}^{(k)}[t]-\lambda_{\mathbf{r}}^{-1}\epsilon_k 2\gamma^2\mathbf{z}^{(k)}[t]\mathbf{z}^{(k)}[t]^T+\epsilon_k g_{lk}\mathbf{I} \tag{A.32}$$

Using the relationship $\mathbf{B}^{(k)}[t] = \frac{1}{\epsilon_k 2\gamma}\mathbf{M}^{(k)}[t] - \frac{g_{lk}}{2\gamma}\mathbf{I}$, we can rearrange the right-hand side of (A.32) such that

$$\mathbf{M}^{(k)}[t+1] = \lambda_{\mathbf{r}}^{-1}\mathbf{M}^{(k)}[t] - \lambda_{\mathbf{r}}^{-1}\epsilon_k 2\gamma^2 \mathbf{z}^{(k)}[t]\mathbf{z}^{(k)}[t]^T + \epsilon_k g_{lk}(1-\lambda_{\mathbf{r}}^{-1})\mathbf{I} \tag{A.33}$$

Here, $\mathbf{z}^{(k)}[t] = \left(\frac{1}{\epsilon_k 2\gamma}\mathbf{M}^{(k)}[t]\mathbf{r}^{(k)} - \frac{g_{lk}}{2\gamma}\mathbf{r}^{(k)}\right)\Big|_{\beta=\beta'}$. It's worth noting that in Section 2.3.1, we decomposed $\mathbf{M}^{(k)}[t]$ into $\mathbf{D}^{(k)}[t]$ and $\mathbf{O}^{(k)}[t]$ as $\mathbf{M}^{(k)}[t] = \mathbf{D}^{(k)}[t] - \mathbf{O}^{(k)}[t]$, representing autapses and lateral inhibition synapses, respectively. Therefore, the update rule in (A.33) leads to the following updates:

$$\mathbf{D}_{ii}^{(k)}[t+1] = \lambda_{\mathbf{r}}^{-1}\mathbf{D}_{ii}^{(k)}[t] - \lambda_{\mathbf{r}}^{-1}\epsilon_k 2\gamma^2(\mathbf{z}_i^{(k)}[t])^2 + \epsilon_k g_{lk}(1-\lambda_{\mathbf{r}}^{-1}), \quad \forall i \in \{1,\dots,N_k\} \tag{A.34}$$

$$\mathbf{O}_{ij}^{(k)}[t+1] = \lambda_{\mathbf{r}}^{-1}\mathbf{O}^{(k)}[t]_{ij} + \lambda_{\mathbf{r}}^{-1}\epsilon_k 2\gamma^2 \mathbf{z}_i^{(k)}[t]\mathbf{z}_j^{(k)}[t], \ \forall i,j \in \{1,\dots,N_k\}, \text{ where } i \neq j \tag{A.35}$$

G On the asymmetry of feedforward and feedback weights

In the main article, we emphasized that one of the key contributions of the proposed supervised CorInfoMax framework is to offer a natural resolution to the weight symmetry problem found in some biologically plausible neural network frameworks. In this section, we aim to expand upon this aspect and supplement the discussion provided in Section 3.

G.1 The importance of the choice of alternative CMI expressions

In developing the update formula for the layer activation $\mathbf{r}^{(k)}$, we used the gradient (A.14). This gradient comprises two alternative expressions of CMI: the first, $\hat{I}^{\overrightarrow{(\epsilon_{k-1})}}(\mathbf{r}^{(k-1)},\mathbf{r}^{(k)})[t]$ represents the CMI with the preceding layer, incorporating the forward prediction error for estimating $\mathbf{r}^{(k)}$. The second, $\hat{I}^{\overleftarrow{(\epsilon_k)}}(\mathbf{r}^{(k)},\mathbf{r}^{(k+1)})[t]$, signifies the CMI with the subsequent layer, encompassing the backward prediction error for forecasting $\mathbf{r}^{(k)}$. These carefully selected expressions were pivotal in realizing a canonical network model that is free from the weight symmetry issue.

To underscore the significance of our selection, let's examine an alternate form of (A.14):

$$\begin{aligned}\nabla_{\mathbf{r}^{(k)}}\hat{J}_k(\mathbf{r}^{(k)})[t] &= \nabla_{\mathbf{r}^{(k)}}\hat{I}^{\overrightarrow{(\epsilon_{k-1})}}(\mathbf{r}^{(k-1)},\mathbf{r}^{(k)})[t] + \nabla_{\mathbf{r}^{(k)}}\hat{I}^{\overrightarrow{(\epsilon_k)}}(\mathbf{r}^{(k)},\mathbf{r}^{(k+1)})[t] \\ = \frac{1}{2}\nabla_{\mathbf{r}^{(k)}}(\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_{k-1}\boldsymbol{I}) &- \frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}^{(k)}[t] - \frac{1}{\epsilon_k}\boldsymbol{W}_{ff}^{(k)^T}\overrightarrow{\mathbf{e}}^{(k+1)}[t].\end{aligned} \tag{A.36}$$

In this case, both the CMI expressions are founded on forward prediction errors. The gradient expression in (A.36) incorporates both the forward prediction error $\overrightarrow{\mathbf{e}}^{(k)}[t]$ and the backpropagated forward prediction error $\overrightarrow{\mathbf{e}}^{(k+1)}[t]$, weighted by $\boldsymbol{W}_{ff}^{(k)^T}$. Consequently, choosing (A.36) would result in a derived network model displaying symmetric feedforward and feedback weights.

This condition is typically encountered in predictive coding-based supervised schemes, such as [22, 23], where the loss function is formulated based on feedforward prediction errors, and the symmetric feedback path arises from calculating the gradient for this loss function. Recently, Golkar et al. [10] proposed a loss function that involves two alternative feedforward prediction error elements for the same branch. By coupling this with an upper bound optimization and a whitening assumption, they manage to sidestep the weight symmetry issue. In contrast, our CorInfoMax framework provides a natural solution to the weight transport problem encountered in predictive coding networks by carefully selecting alternative CMI expressions, thus eliminating the need for the whitening assumption.

G.2 The analytical comparison of forward and backward prediction coefficients

Regarding the correlative mutual information component of the proposed stochastic objective presented in (1), the optimal solutions for both feedforward and feedback weights are found as solutions

to problems (3) and (5) respectively. To be more specific, these expressions can be framed as follows:

$$\boldsymbol{W}_{ff,*}^{(k)} = \boldsymbol{R}_{\mathbf{r}^{(k)}\mathbf{r}^{(k+1)}}^{T}(\boldsymbol{R}_{\mathbf{r}^{(k)}} + \epsilon_k \mathbf{I})^{-1}, \tag{A.37}$$

$$\boldsymbol{W}_{fb,*}^{(k)} = \boldsymbol{R}_{\mathbf{r}^{(k)}\mathbf{r}^{(k+1)}}(\boldsymbol{R}_{\mathbf{r}^{(k+1)}} + \epsilon_k \mathbf{I})^{-1}. \tag{A.38}$$

Consequently, the condition $\boldsymbol{W}_{ff,}^{(k)} = \boldsymbol{W}_{fb,}^{(k)^T}$ does not generally hold true. Symmetry might be anticipated in very specific scenarios - such as when layer activations are signed with a zero mean, and the autocorrelation matrices meet the whiteness condition, i.e., $\boldsymbol{R}_{\mathbf{r}^{(k)}} = \sigma_r^2\mathbf{I}$ and $\boldsymbol{R}_{\mathbf{r}^{(k+1)}} = \sigma_r^2\mathbf{I}$. This analysis only considers the mutual information maximization component of the objective, yet it offers insight into the expected asymmetry of the forward and backward weights.

While this analysis focuses solely on the mutual information maximization component of the objective, it still provides valuable insight into the anticipated asymmetry of the forward and backward weights.

G.3 Empirical angle between forward and backward weights

To test the level of symmetry between the forward prediction synaptic weight matrix $\boldsymbol{W}_{ff}^{(k)}$ and the transpose of the backward prediction synaptic weight matrix $\boldsymbol{W}_{fb}^{(k)}$, we investigate the cosine angle between them. This measure is calculated using [17]:

$$\Theta^{(k)} = \arccos\left(\frac{\mathrm{Tr}\left(\boldsymbol{W}_{ff}^{(k)}\boldsymbol{W}_{fb}^{(k)}\right)}{\|\boldsymbol{W}_{ff}^{(k)}\|_F\|\boldsymbol{W}_{fb}^{(k)}\|_F}\right). \tag{A.39}$$

This metric serves as an indicator of alignment - with a cosine angle of $\Theta^{(k)} = 0$ degrees signifying perfect symmetry, and $\Theta^{(k)} = 90$ degrees indicating orthogonality and, thus, a significant degree of asymmetry. To provide a clearer illustration of the asymmetry between feedforward and feedback weights for the proposed CorInfoMax networks, we carried out a numerical evaluation of this metric in the context of the experiments documented in Table 1. It's worth noting that in these experiments, the feedforward and feedback weights were independently initialized with random values.

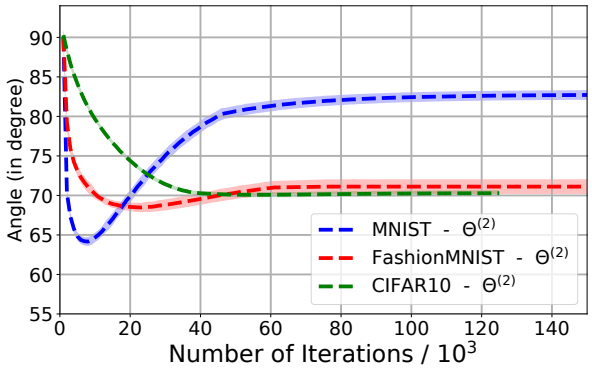


Figure 2: The angle between the feedforward and the transpose of the feedback weights between hidden and output layers (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) as a function of weight update iterations for CorInfoMax-$\mathcal{B}_{\infty,+}$.

Figure 2 presents the experimental alignment results as a function of iterations for the MNIST, Fashion MNIST, and CIFAR10 datasets. According to this figure, the curves initially start at 90 degrees due to the independent random selection of initial weights. As the iterations progress, the angle does indeed lessen, however, the steady-state levels largely persist, suggesting a noteworthy degree of asymmetry between the feedforward and feedback weights.

H Alternative expression for the feedforward weight updates

In Section 2.4, we derived the partial derivative of the forward prediction cost with respect to the feedforward weights as follows:

$$\frac{\partial C_{ff}(\boldsymbol{W}_{ff}^{(k)}[t])}{\partial \boldsymbol{W}_{ff}^{(k)}} = 2\epsilon_k \boldsymbol{W}_{ff}^{(k)}[t] - 2\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T. \tag{A.40}$$

In this equation, $\overrightarrow{\mathbf{e}}^{(k+1)}[t]$ symbolizes the feedforward error value at the point when system dynamics have stabilized. In order to characterize this steady state value, we look at the optimal value of the optimization problem employed to derive the system dynamics:

$$\begin{aligned} \underset{\mathbf{r}^{(k)}[t]}{\text{maximize}} \quad & \left(\tfrac{1}{2}(\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_{k-1}\boldsymbol{I}) + \log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_k \boldsymbol{I})) \right. \\ & \left. - \tfrac{1}{2\epsilon_{k-1}} \left\| \overrightarrow{\mathbf{e}}^{(k)}[t] \right\|_2^2 - \tfrac{1}{2\epsilon_k} \left\| \overleftarrow{\mathbf{e}}^{(k)}[t] \right\|_2^2 \right) \\ \text{subject to} \quad & \mathbf{0} \preccurlyeq \mathbf{r}^{(k)}[t] \preccurlyeq \mathbf{1}. \end{aligned} \tag{A.41}$$

We formulate the corresponding Lagrangian function for this optimization as

$$L(\mathbf{r}^{(k)}[t], \mathbf{q}_1[t], \mathbf{q}_2[t]) = O(\mathbf{r}^{(k)}[t]) + \mathbf{q}_1[t]^T(\mathbf{r}^{(k)}[t]) + \mathbf{q}_2[t]^T(\mathbf{1} - \mathbf{r}^{(k)}[t]) \tag{A.42}$$

where $O(\mathbf{r}^{(k)}[t])$ corresponds to the objective in(A.41), and $\mathbf{q}_1[t], \mathbf{q}_2[t] \succcurlyeq \mathbf{0}$. Applying the KKT optimality conditions [67], we find

$$\nabla_{\mathbf{r}^{(k)}} L(\mathbf{r}_*^{(k)}[t], \mathbf{q}_{1*}[t], \mathbf{q}_{2*}[t]) = \mathbf{0}, \tag{A.43}$$

at the optimal point $(\mathbf{r}_*^{(k)}[t], \mathbf{q}_{1*}, \mathbf{q}_{2*}[t])$. This results in

$$2g_{B,k}\overrightarrow{\mathbf{e}}_*^{(k+1)}[t] + 2\left(g_{A,k}\mathbf{v}_A^{(k)}[t] - (g_{lk} + g_{A_k})\mathbf{r}_*^{(k)}[t]\right) + \mathbf{q}_{1*}[t] + \mathbf{q}_{2*}[t] = \mathbf{0}, \tag{A.44}$$

where $\mathbf{q}_{1*}[t]$ and $\mathbf{q}_{2*}[t]$ are entirely null except at the indices where $\mathbf{r}_*^{(k)}[t]$ equals 0 or 1, owing to the nonnegative-clipping operation of $\sigma_+(\mathbf{u}_*^{(k)}[t])$.

Thus, by defining $\mathbf{q}_*[t] = \mathbf{q}_{1*}[t] + \mathbf{q}_{2*}[t]$, the outer product term in (A.40) used in the feedforward weight update can be equivalently expressed as

$$-2\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T = \frac{2}{g_{B,k}}\left(g_{A,k}\mathbf{v}_A^{(k)}[t] - (g_{lk} + g_{A_k})\mathbf{r}_*^{(k)}[t] + \frac{1}{2}\mathbf{q}_*[t]\right)\mathbf{r}^{(k)}[t]^T. \tag{A.45}$$

Based on $\mathbf{r}_*^{(k)}[t] = \sigma_+(\mathbf{u}_*^{(k)}[t])$, we can express the relationship between $\mathbf{r}_*^{(k)}[t]$ and $\mathbf{u}_*^{(k)}[t]$ as $\mathbf{r}_*^{(k)}[t] = \mathbf{u}_*^{(k)}[t] + \mathbf{d}[t]$, where $\mathbf{d}[t]$ has non-zero values only at the indices $\mathbf{u}^{(k)}$ is negative or exceeds 1. Consequently, we can rewrite the outer product term in (A.45) as

$$-2\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T = \frac{2}{g_{B,k}}\left(g_{A,k}\mathbf{v}_A^{(k)}[t] - (g_{lk} + g_{A_k})\mathbf{u}_*^{(k)}[t] + \mathbf{h}_*[t]\right)\mathbf{r}^{(k)}[t]^T, \tag{A.46}$$

where $\mathbf{h}_*[t] = \frac{1}{2}\mathbf{q}_*[t] - (g_{lk} + g_{A_k})\mathbf{d}[t]$.

Ultimately, following a similar approach as in [10], we infer that the update expression for the feedforward term can be derived from the difference between the soma's membrane voltages and the distal apical compartments. This conclusion aligns with experimental observations that emphasize the dependence of basal synaptic plasticity on apical calcium plateau potentials [7].

I Employing sparsity assumption on neuronal activities

In this section, we elaborate on the structure of the CorInfoMax network and the corresponding neuronal dynamics that emerge with the selection of the activation domain $\mathcal{P}^{(k)} = \mathcal{B}_{1,+} = \{\mathbf{r}$:

$\|\mathbf{r}\|_1 \leq 1, \mathbf{0} \preccurlyeq \mathbf{r}\}$. It's important to note that this domain corresponds to the intersection of the ℓ_1-norm ball and the nonnegative orthant.

To derive the network dynamics corresponding to $\mathbf{r}^{(k)}[t]$, we consider the following constrained optimization similar to (A.41),

$$\begin{aligned} \underset{\mathbf{r}^{(k)}[t]}{\text{maximize}} \quad & \left(\tfrac{1}{2}(\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_{k-1}\boldsymbol{I}) + \log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_k \boldsymbol{I})) \right. \\ & \left. - \tfrac{1}{2\epsilon_{k-1}} \left\| \overrightarrow{\mathbf{e}}^{(k)}[t] \right\|_2^2 - \tfrac{1}{2\epsilon_k} \left\| \overleftarrow{\mathbf{e}}^{(k)}[t] \right\|_2^2 \right) \\ \text{subject to} \quad & \|\mathbf{r}^{(k)}[t]\|_1 \leq 1, \\ & \mathbf{0} \preccurlyeq \mathbf{r}^{(k)}[t]. \end{aligned} \tag{A.47}$$

We can write down the Lagrangian min-max problem corresponding to this optimization as

$$\underset{q^{(k)}[t] \geq 0}{\text{minimize}} \; \underset{\mathbf{r}^{(k)}[t] \succcurlyeq \mathbf{0}}{\text{maximize}} \; L(\mathbf{r}^{(k)}[t], q^{(k)}[t]) = O(\mathbf{r}^{(k)}[t]) + q^{(k)}[t](1 - \|\mathbf{r}^{(k)}[t]\|_1). \tag{A.48}$$

Here $O(\mathbf{r}^{(k)}[t])$ signifies the objective in (A.47). By applying the proximal gradient update [68] for $\mathbf{r}^{(k)}[t]$ using the gradient expression in (A.14), we can represent the output dynamics for layer-k as

$$\begin{aligned} \tau_{\mathbf{u}} \frac{d\mathbf{u}^{(k)}[t;s]}{ds} &= -g_{lk}\mathbf{u}^{(k)}[t;s] + \frac{1}{\epsilon_k}\boldsymbol{M}^{(k)}[t]\boldsymbol{r}^{(k)}[t;s] - \frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}_u^{(k)}[t;s] - \frac{1}{\epsilon_k}\overleftarrow{\mathbf{e}}_u^{(k)}[t;s], \\ \overrightarrow{\mathbf{e}}_u^{(k)}[t;s] &= \mathbf{u}^{(k)}[t;s] - \boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t;s], \\ \overleftarrow{\mathbf{e}}_u^{(k)}[t;s] &= \mathbf{u}^{(k)}[t;s] - \boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)}[t;s], \\ \mathbf{r}^{(k)}[t;s] &= \text{ReLU}(\mathbf{u}^{(k)}[t;s] - q^{(k)}[t;s]\mathbf{1}). \end{aligned}$$

In this formulation, we introduce the intermediate variable $\mathbf{u}^{(k)}$, where ReLU represents the element-wise rectified linear unit. We then define the apical and basal potentials as

$$\begin{aligned} \mathbf{v}_A^{(k)}[t;s] &= \boldsymbol{M}^{(k)}[t]\boldsymbol{r}^{(k)}[t;s] + \boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)}[t;s] - \frac{1}{g_{A,k}}q^{(k)}[t;s]\mathbf{1}, \\ \mathbf{v}_B^{(k)}[t;s] &= \boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t;s]. \end{aligned}$$

With these potentials in place, we can restate the output dynamics in a format similar to (19) and (20),

$$\tau_{\mathbf{u}} \frac{d\mathbf{u}^{(k)}[t;s]}{ds} = -g_{lk}\mathbf{u}^{(k)}[t;s] + g_{A,k}(\mathbf{v}_A^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s]) + g_{B,k}(\mathbf{v}_B^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s]), \tag{A.49}$$

$$\mathbf{r}^{(k)}[t;s] = \text{ReLU}(\mathbf{u}^{(k)}[t;s]). \tag{A.50}$$

For the Lagrangian variable $q^{(k)}[t;s]$, we can derive the update based on the dual minimization as follows,

$$\frac{da^{(k)}[t;s]}{ds} = -a^{(k)}[t;s] + \sum_{j=1}^{N_k} \mathbf{r}_j^{(k)}[t;s] - 1 + q^{(k)}[t;s], \quad q^{(k)}[t;s] = \text{ReLU}(a^{(k)}[t;s]). \tag{A.51}$$

In the above formulation, the Lagrangian variable $q^{(k)}$ equates to an inhibitory inter-neuron that receives input from all neurons of layer-k and generates an inhibitory signal to the apical compartments of all these neurons.

J Supplementary on numerical experiments

In the forthcoming subsections, we delve into the specifics of our experimental setup, which covers everything from hyperparameters to an in-depth analysis. We utilize four image classification tasks to evalutate the effectiveness of our proposed framework: MNIST [50], Fashion MNIST [51], CIFAR10, and CIFAR100 [52]. These image datasets are all acquired from the Pytorch library [69].

The MNIST dataset includes 60000 grayscale training images of hand-drawn digits and 10000 test images, each with a dimension of 28×28. The Fashion MNIST dataset mirrors the MNIST in terms of format, size, and quantity of training and test images. However, instead of digits, it features images of clothing items, divided into 10 categories.

On a larger scale, we have the CIFAR10 and CIFAR100 datasets, which contain 32×32 RGB images. CIFAR10 consists of 50000 training images and 10000 test images, each associated with one of 10 object labels. CIFAR100, while maintaining the same quantity of training and test images as CIFAR10, is distinguished by its 100 object categories.

We initially map the image pixels to the range $[0, 1]$ by rescaling them by 255. For the CIFAR10 and CIFAR100 datasets, normalization is performed using the mean and standard deviation values cited in the published codes of [25, 26]. To accommodate our training of MLP architectures, the images are flattened prior to being fed into the neural networks. Lastly, we have opted not to employ any augmentation in our numerical experiments.

J.1 On the computation of neural dynamics

The neural dynamics equations defined by (19)-(20) implicitely determines the input-output relations of the CorInfoMax-$\mathcal{B}_{\infty,+}$ network for each segments. We obtain the solution of these equations by applying the discrete-time method that is provided in Algorithm 1. Within this algorithm, $\mu_{\mathbf{u}}[s]$ refers to the learning rate of the neural dynamics during the gradual time scale indicated by index s. Additionally, $s_{\max}$ stands for the highest count of iterations in the loop-driven calculations of neural dynamics.

Algorithm 1 CorInfoMax neural dynamic iterations: $\mathcal{P}^{(k)} = \mathcal{B}_{\infty,+}$

Initialize $\mathbf{r}^{(k)}[t;1]$, $\mathbf{u}^{(k)}[t;1]$, $\mu_{\mathbf{u}}[1]$, $s_{\max}$, and $s = 1$
while $s < s_{\max}$ **do**
  for $k = 1, \ldots, P$ **do**
    $\tau_{\mathbf{u}} \frac{d\mathbf{u}^{(k)}[t;s]}{ds} = -g_{lk}\mathbf{u}^{(k)}[t;s] + g_{A,k}(\mathbf{v}_A^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s]) + g_{B,k}(\mathbf{v}_B^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s])$
    $\mathbf{u}^{(k)}[t;s+1] = \mathbf{u}^{(k)}[t;s] + \mu_{\mathbf{u}}[s]\tau_{\mathbf{u}} \frac{d\mathbf{u}^{(k)}[t;s]}{ds}$
    $\mathbf{r}^{(k)}[t;s+1] = \sigma_+(\mathbf{u}^{(k)}[t;s+1])$
  end for
  $s = s + 1$, and adjust $\mu_{\mathbf{u}}[s]$ if necessary.
end while

Similarly, the equations defined by (A.49), (A.50), and (A.51 implicitely determines the input-output relations of the CorInfoMax-$\mathcal{B}_{1,+}$ network for each segment with an additional inhibition signal $q^{(k)}$. In an analogous manner, we use the following discrete-time method outlined in Algorithm 2 to obtain the solution of these equations. In contrast to Algorithm 1, Algorithm 2 introduces an extra hyperparameter denoted as $\mu_a[s]$. This hyperparameter governs the learning rate of supplementary inhibitory neurons $q^{(k)}$. Detailed discussions concerning the specific values of each hyperparameter related to the neural dynamics for individual tasks are presented in the subsequent subsections.

Algorithm 2 CorInfoMax neural dynamic iterations: $\mathcal{P}^{(k)} = \mathcal{B}_{1,+}$

Initialize $\mathbf{r}^{(k)}[t;1]$, $\mathbf{u}^{(k)}[t;1]$, $a^{(k)}[t;1]$, $q^{(k)}[t;1]$, $\mu_{\mathbf{u}}[1]$, $\mu_a[1]$, $s_{\max}$, and $s = 1$
while $s < s_{\max}$ **do**
  for $k = 1, \ldots, P$ **do**
    $\tau_{\mathbf{u}} \frac{d\mathbf{u}^{(k)}[t;s]}{ds} = -g_{lk}\mathbf{u}^{(k)}[t;s] + g_{A,k}(\mathbf{v}_A^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s]) + g_{B,k}(\mathbf{v}_B^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s])$
    $\frac{da^{(k)}[t;s]}{ds} = -a^{(k)}[t;s] + \sum_{j=1}^{N_k} \mathbf{r}_j^{(k)}[t;s] - 1 + q^{(k)}[t;s]$
    $\mathbf{u}^{(k)}[t;s+1] = \mathbf{u}^{(k)}[t;s] + \mu_{\mathbf{u}}[s]\tau_{\mathbf{u}} \frac{d\mathbf{u}^{(k)}[t;s]}{ds}$
    $\mathbf{r}^{(k)}[t;s+1] = \text{ReLU}(\mathbf{u}^{(k)}[t;s+1])$
    $a^{(k)}[t;s+1] = a^{(k)}[t;s]) + \mu_a[s] \frac{da^{(k)}[t;s]}{ds}$
    $q^{(k)}[t;s+1] = \text{ReLU}(a^{(k)}[t;s+1])$
  end for
  $s = s + 1$, and adjust $\mu_{\mathbf{u}}[s]$ and $\mu_a[s]$ if necessary.
end while

J.2 Learning Dynamics

We outline the comprehensive learning dynamics of our proposed framework in Algorithm 3. As we mentioned earlier, we adopt the Equilibrium Propagation technique [24], leading our approach through two distinct phases of neural dynamics prior to weight updates: i) the free phase, and ii) the nudged phase. Initially, we execute the free phase by setting $\beta = 0$. Subsequently, we proceed with the nudged phase, activating the neural dynamics with $\beta = \beta' > 0$. The adjustments to the feedforward and feedback weights are then determined by contrasting neural activities between the nudged and free phases. Concerning the update equation for the forward weights in (26), the error vector $\overrightarrow{\mathbf{e}}^{(k+1)}[t]$ evaluated at $(\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T)\big|_{\beta=\beta'}$ corresponds to $\overrightarrow{\mathbf{e}}^{(k)}[t]_{\beta=\beta'} = \mathbf{r}^{(k)}[t]_{\beta=\beta'} - \boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t]_{\beta=\beta'}$. Similarly, for the update expression of the backward weights in (27), the error vector $\overleftarrow{\mathbf{e}}^{(k)}[t]$ at $(\overleftarrow{\mathbf{e}}^{(k)}[t]\mathbf{r}^{(k+1)}[t]^T)\big|_{\beta=\beta'}$ corresponds to $\overleftarrow{\mathbf{e}}^{(k)}[t]\big|_{\beta=\beta'} = \mathbf{r}^{(k)}[t]\big|_{\beta=\beta'} - \boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)}[t]\big|_{\beta=\beta'}$. In Algorithm 3, the learning rates for the forward and backward weights are denoted as μ_{ff} and μ_{fb}, respectively. Lateral weight updates adhere to the learning rule outlined in [11], and we rewrite them in terms of autapses and lateral inhibition synapses (Appendix F). These updates are performed based on the neural activities subsequent to the nudged phase.

Algorithm 3 CorInfoMax network learning dynamics

Initialize network parameters: $\boldsymbol{W}_{ff}^{(k)}[1]$, $\boldsymbol{W}_{fb}^{(k)}[1]$, $\boldsymbol{B}^{(k)}[1]$, ϵ_k, (for each k), $\lambda_{\mathbf{r}}$, g_{lk}
Initialize hyperparameters: β', T_{free}, T_{nudged}, μ_{ff}, μ_{fb}, $\mu_{\mathbf{u}}$, μ_a (if $\mathcal{P} = \mathcal{B}_{1,+}$)
for t = 1, 2, ... **do**
  Run neural dynamics with $\beta = 0$ **for** $s_{\max} = T_{\text{free}}$ to collect $\mathbf{r}^{(k)}[t]\big|_{\beta=0}$ ($k = 1, \ldots P$)
  Run neural dynamics with $\beta = \beta'$ **for** $s_{\max} = T_{\text{nudged}}$ to collect $\mathbf{r}^{(k)}[t]\big|_{\beta=\beta'}$ ($k = 1, \ldots P$)
  Update synaptic weights with (26) and (27) for each k:

$$\boldsymbol{W}_{ff}^{(k)}[t+1] = \boldsymbol{W}_{ff}^{(k)}[t] + \mu_{\text{ff}} \frac{1}{\beta'} \left((\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T)\Big|_{\beta=\beta'} - (\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T)\Big|_{\beta=0} \right)$$

$$\boldsymbol{W}_{fb}^{(k)}[t+1] = \boldsymbol{W}_{fb}^{(k)}[t] + \mu_{\text{fb}} \frac{1}{\beta'} \left((\overleftarrow{\mathbf{e}}^{(k)}[t]\mathbf{r}^{(k+1)}[t]^T)\Big|_{\beta=\beta'} - (\overleftarrow{\mathbf{e}}^{(k)}[t]\mathbf{r}^{(k+1)}[t]^T)\Big|_{\beta=0} \right)$$

  Update lateral weights for each k (see Appendix F):

$$\mathbf{D}_{ii}^{(k)}[t+1] = \lambda_{\mathbf{r}}^{-1}\mathbf{D}_{ii}^{(k)}[t] - \lambda_{\mathbf{r}}^{-1}\epsilon_k 2\gamma^2(\mathbf{z}_i^{(k)}[t])^2 + \epsilon_k g_{lk}(1 - \lambda_{\mathbf{r}}^{-1}), \quad \forall i \in \{1, \ldots, N_k\}$$

$$\mathbf{O}_{ij}^{(k)}[t+1] = \lambda_{\mathbf{r}}^{-1}\mathbf{O}_{ij}^{(k)}[t] + \lambda_{\mathbf{r}}^{-1}\epsilon_k 2\gamma^2 \mathbf{z}_i^{(k)}[t]\mathbf{z}_j^{(k)}[t], \ \ \forall i, j \in \{1, \ldots, N_k\}, \text{ where } i \neq j$$

end for

J.3 Description of hyperparameters

In the upcoming sections, we present the hyperparameters and their effects in our experiments. Therefore, Table 2 describes notation for the hyperparameters used in CorInfoMax neural dynamics and learning updates.

Table 2: Description of the hyperparameter notations.

Hyperparameter	Description
Architecture	An array containing the dimension of each layer.
T_{free}	Number of neural dynamics iterations for the free phase.
T_{nudged}	Number of neural dynamics iterations for the nudged phase.
μ_{ff}	An array containing the learning rates for feedforward weights.
μ_{fb}	An array containing the learning rates for feedback weights.
$\lambda_{\mathbf{r}}$	Forgetting factor for sample the auto correlation matrices in (6).
ϵ_k	Perturbation coefficient for autocorrelation matrices in (2) and (4).
β'	Nudging parameter for the nudged neural dynamics.
g_{lk}	Leak coefficient for the neural dynamics in (20) and (A.50).
$\mu_{\mathbf{u}}$	Learning rate for the neural dynamics in Algorithm 1 and 2.
μ_a	Learning rate for the inhibition neuron in Algorithm 2.
lr decay	Learning rate decay that multiplies the μ_{ff} and μ_{fb} after each epoch.

J.4 Two layer CorInfoMax-$\mathcal{B}_{\infty,+}$ network

In this section, we provide supplementary experimental results for the CorInfoMax-$\mathcal{B}_{\infty,+}$ network with a single hidden layer.

J.4.1 Network architecture

Figure 3 provides a depiction of a CorInfoMax network with a single hidden layer. In this instance, both the hidden and output layers have the same constraint set $\mathcal{P} = \mathcal{B}_{\infty,+}$.

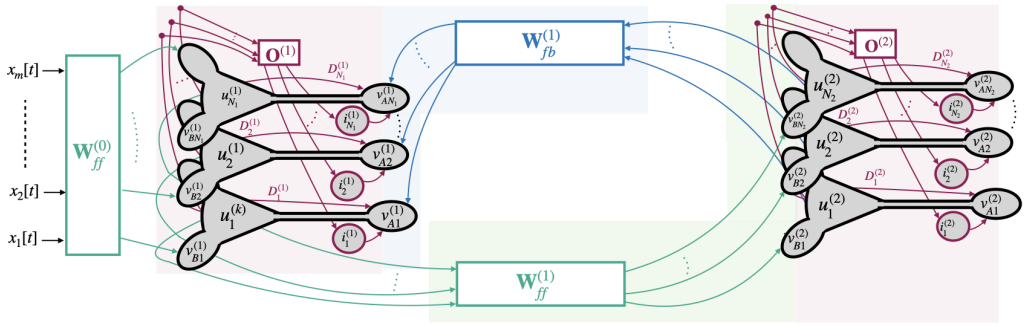


Figure 3: Two layer correlative information maximization based neural network with non-negative antisparsity constraint: $\mathbf{0} \preccurlyeq \mathbf{r}^{(l)} \preccurlyeq \mathbf{1}, l = 1, 2$.

J.4.2 Hyperparameters

Table 3 provides the hyperparameters to train 2-layer CorInfoMax networks on MNIST, Fashion MNIST, and CIFAR10, for which the corresponding test accuracy results are provided in Section 4 and in Appendix J.4.3.

Table 3: Hyperparameters used to train CorInfoMax-$\mathcal{B}_{\infty,+}$ networks with single hidden layer. (In the row stating the lr decay, ep. and O/W means *epoch* and *otherwise*, respectively.)

Hyperparameter	MNIST	Fashion-MNIST	CIFAR10
Batch size	20	20	20
Architecture	$[784, 500, 10]$	$[784, 500, 10]$	$[3072, 1000, 10]$
T_{free}	30	30	30
T_{nudged}	10	10	10
μ_{ff}	$[1.0, 0.7]$	$[0.3, 0.22]$	$[0.08, 0.04]$
μ_{fb}	$[-, 0.15]$	$[-, 0.07]$	$[-, 0.04]$
$\lambda_{\mathbf{r}}$	$1 - 10^{-5}$	$1 - 10^{-5}$	$1 - 5 \times 10^{-5}$
ϵ_k	$0.15 \quad \forall k$	$0.15 \quad \forall k$	$0.15 \quad \forall k$
β'	1.0	1.0	1.0
g_{lk}	0.5	0.3	0.1
$\mu_{\mathbf{u}}$	$\max\{\frac{0.05}{s\times 10^{-2}+1}, 10^{-3}\}$	$\max\{\frac{0.07}{s\times 10^{-2}+1}, 10^{-3}\}$	0.05
lr decay	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$	$\begin{cases} 0.95 & \text{ep.} < 20 \\ 0.9 & 20 \leq \text{ep.} < 25 \\ 0.8 & \text{O/W.} \end{cases}$	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$

J.4.3 Test accuracy results

Figure 4 compares the test accuracy performance of CorInfoMax-$\mathcal{B}_{\infty,+}$ network (as a function of training epochs) with CSM, EP, PC, and PC-Nudge algorithms for the MNIST dataset.

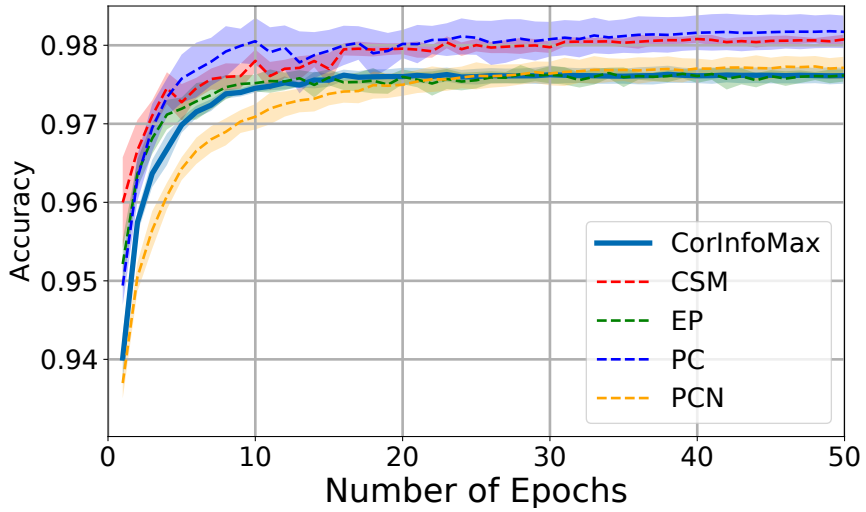


Figure 4: Test accuracy convergences of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks and other (CSM, EP, PC, and PC-Nudge) algorithms as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the MNIST dataset.

Figure 5 compares the test accuracy performance of CorInfoMax-$\mathcal{B}_{\infty,+}$ network (as a function of training epochs) with CSM, EP, PC, and PC-Nudge algorithms for the Fashion MNIST dataset.

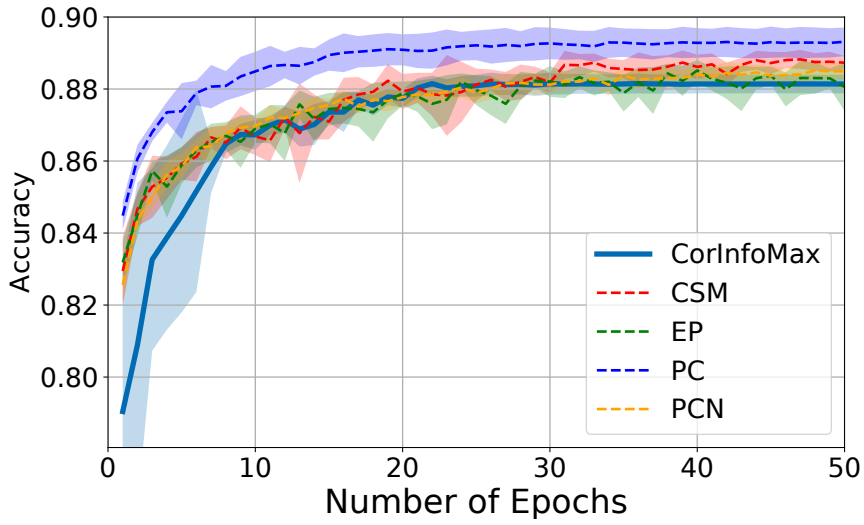


Figure 5: Test accuracy convergences of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks and other (CSM, EP, PC, and PC-Nudge) algorithms as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the Fashion MNIST dataset.

Figure 6 compares the test accuracy performance of CorInfoMax-$\mathcal{B}_{\infty,+}$ network (as a function of training epochs) with CSM, EP, PC, and PC-Nudge algorithms for the CIFAR10 dataset.

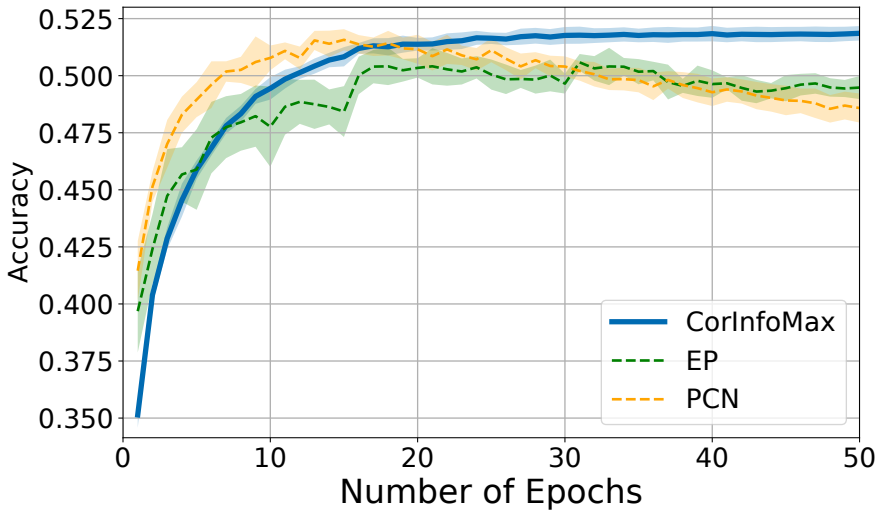


Figure 6: Test accuracy convergences of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks and other (CSM, EP, PC, and PC-Nudge) algorithms as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the CIFAR10 dataset.

J.5 Three layer CorInfoMax-$\mathcal{B}_{\infty,+}$ network

In this section, we consider the extension of the CorInfoMax-$\mathcal{B}_{\infty,+}$ network in Figure 3, obtained by inserting an additional hidden layer.

J.5.1 Hyperparameters

Table 4: Hyperparameters used to train CorInfoMax-$\mathcal{B}_{\infty,+}$ networks with two hidden layers.(In the row stating the lr decay, ep. and O/W means *epoch* and *otherwise*, respectively.)

Hyperparameter	MNIST	CIFAR10	CIFAR100
Batch size	20	20	20
Architecture	$[784, 500, 500, 10]$	$[3072, 1000, 500, 10]$	$[3072, 2000, 1000, 100]$
T_{free}	30	30	50
T_{nudged}	10	10	20
μ_{ff}	$[1.1, 0.75, 0.6]$	$[0.11, 0.06, 0.035]$	$[0.18, 0.15, 0.09]$
μ_{fb}	$[-, 0.17, 0.07]$	$[-, 0.045, 0.015]$	$[-, 0.08, 0.06]$
$\lambda_{\mathbf{r}}$	$1 - 10^{-5}$	$1 - 10^{-5}$	$1 - 10^{-5}$
ϵ_k	$0.15 \quad \forall k$	$0.15 \quad \forall k$	$0.15 \quad \forall k$
β'	1.0	1.0	1.0
g_{lk}	0.5	0.1	0.1
$\mu_{\mathbf{u}}$	0.05	0.05	0.06
lr decay	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$	$\begin{cases} 0.99 & \text{ep.} < 20 \\ 0.9 & \text{O/W.} \end{cases}$

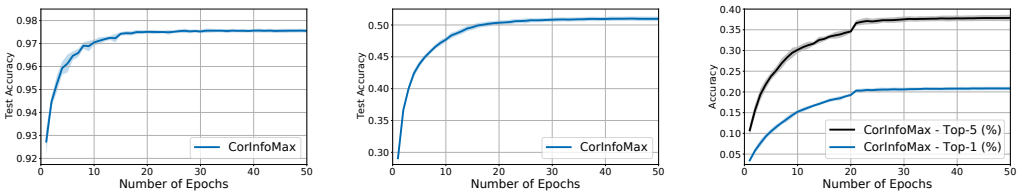
J.5.2 Test accuracy results

Table 5 displays the test accuracy results for the 3-layer CorInfoMax-$\mathcal{B}_{\infty,+}$ networks on MNIST, CIFAR10 and CIFAR100 datasets in comparison with Feedback Alignment and Backpropagation algorihtms.

Table 5: Test accuracy ($n = 10$ runs $\pm$ stddev) of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks with two hidden layers.

	MNIST	CIFAR10	CIFAR100
	Top-1 (%)	Top-1 (%)	Top-1 (%) / Top5 (%)
CorInfoMax-$\mathcal{B}_{\infty,+}$	97.58 ± 0.1	50.97 ± 0.4	20.84 ± 0.4 / 37.86 ± 0.8
Feedback Alignment (with MSE Loss)	98.18 ± 0.0	50.26 ± 1.4	- / -
Feedback Alignment (with CrossEntropy Loss)	97.96 ± 0.2	51.64 ± 0.6	- / -
BP (with MSE Loss)	97.74 ± 0.1	55.49 ± 0.4	26.56 ± 0.2 / 40.64 ± 0.4
BP (with CrossEntropy Loss)	98.28 ± 0.08	56.14 ± 0.3	28.93 ± 0.3 / 54.13 ± 0.4

Figure 7 reports the test accuracy performance of CorInfoMax-$\mathcal{B}_{\infty,+}$ network with two hidden layers as a function of training epochs for the MNIST, CIFAR10 and CIFAR100 image classification tasks.

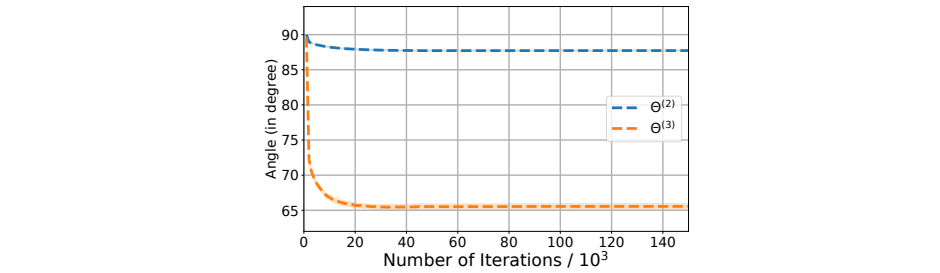


(a) (b) (c)

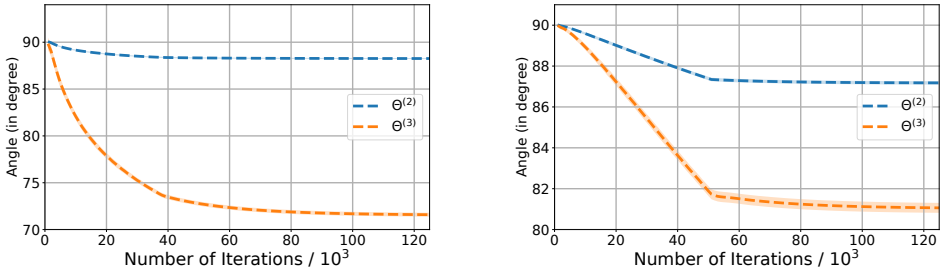
Figure 7: Test accuracy convergence of CorInfoMax-$\mathcal{B}_{\infty,+}$ network with two hidden layers as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the (a) MNIST dataset, (b) CIFAR10 dataset, and (c) CIFAR100 dataset (Top-1(%) and Top-5(%) test accuracy convergences)

J.5.3 Angle measurement results

Figure 8 provides the angle between feedforward and the transpose of the feedback weights (as defined in (A.39)) results for the 3-layer CorInfoMax-$\mathcal{B}_{\infty,+}$ network for the MNIST, CIFAR10 and CIFAR100 datasets. These results also confirm the asymmetry between the feedforward and feedback weights corresponding to the same segment.



(a)



(b) (c)

Figure 8: The angle between the feedforward and the transpose of the feedback weights (based on (A.39) for the 3-layer CorInfoMax-$\mathcal{B}_{\infty,+}$ network (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for (a) MNIST, (b) CIFAR10, and (c) CIFAR100 datasets.

J.6 Two layer CorInfoMax-$\mathcal{B}_{1,+}$ network

J.6.1 Network architecture

Figure 9 provides a depiction of a CorInfoMax network with a single hidden layer. In this instance, both the hidden and output layers have the same constraint set $\mathcal{P} = \mathcal{B}_{1,+}$. Relative to the CorInfoMax-$\mathcal{B}_{\infty,+}$ network structure in Figure 3, this network contains additional interneurons, namely $q^{(1)}$ and $q^{(2)}$ to impose sparsity constraint on hidden and output layer networks.

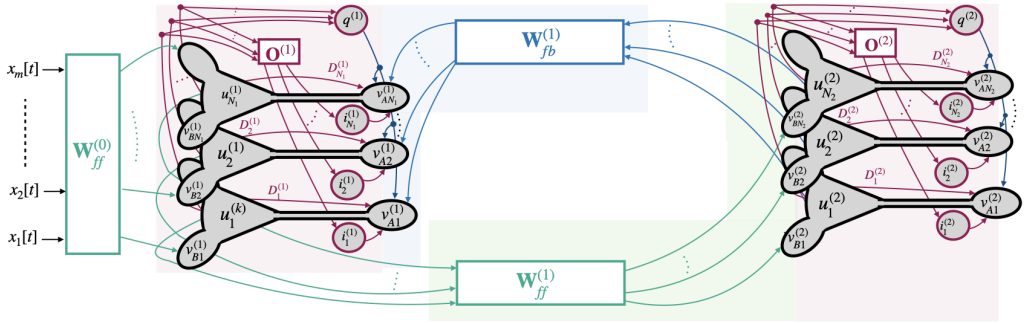


Figure 9: Two layer correlative information maximization based neural network with non-negative sparsity constraint: $\|\mathbf{r}^{(l)}\|_1 \leq 1,$ and $\mathbf{r}^{(l)} \succcurlyeq \mathbf{0}, l = 1, 2$. Interneurons with activations $q^{(1)}$ and $q^{(2)}$ are to impose ℓ_1-norm constraints for both layers.

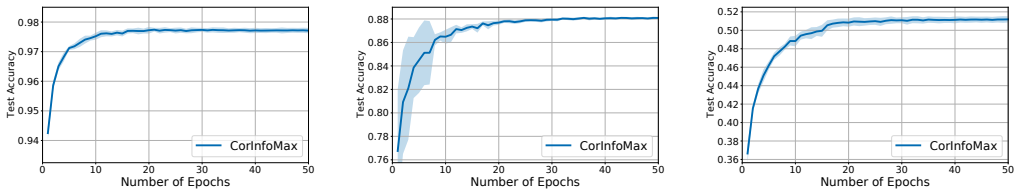
J.6.2 Hyperparameters

Table 6: Hyperparameters used to train two layers CorInfoMax-$\mathcal{B}_{1,+}$ network. (In the row stating the lr decay, ep. and O/W means *epoch* and *otherwise*, respectively.)

Hyperparameter	MNIST	FashionMNIST	CIFAR10
Batch size	20	20	20
Architecture	$[784, 500, 10]$	$[784, 500, 10]$	$[3072, 1000, 10]$
T_{free}	20	20	30
T_{nudged}	4	10	10
μ_{ff}	$[1.0, 0.7]$	$[0.35, 0.23]$	$[0.095, 0.075]$
μ_{fb}	$[-, 0.12]$	$[-, 0.06]$	$[-, 0.05]$
$\lambda_{\mathbf{r}}$	$1 - 10^{-5}$	$1 - 10^{-5}$	$1 - 10^{-5}$
ϵ_k	$0.15 \quad \forall k$	$0.15 \quad \forall k$	$0.15 \quad \forall k$
β'	1.0	1.0	1.0
g_{lk}	0.5	0.2	0.1
$\mu_{\mathbf{u}}$	0.05	$\max\{\frac{0.045}{s\times 10^{-2}+1}, 10^{-3}\}$	$\max\{\frac{0.025}{s\times 10^{-2}+1}, 10^{-3}\}$
μ_a	$[1e-6, 0.01]$	$[1e-6, 0.01]$	$[1e-5, 0.01]$
lr decay	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$	$\begin{cases} 0.95 & \text{ep.} < 11 \\ 0.9 & \text{O/W.} \end{cases}$	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$

J.6.3 Test accuracy results

Figure 10 reports the test accuracy performance of CorInfoMax-$\mathcal{B}_{1,+}$ network with single hidden layer as a function of training epochs for the MNIST, FashionMNIST, and CIFAR10 image classification tasks.



(a) (b) (c)

Figure 10: Test accuracy convergence of CorInfoMax-$\mathcal{B}_{1,+}$ network with single hidden layer as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the (a) MNIST dataset, (b) FashionMNIST dataset, and (c) CIFAR10 dataset.

J.6.4 Angle measurement results

The angle measurements between the feedforward and feedback weights demonstrated in Figure 11 confirm the typical asymmetry between these weights.

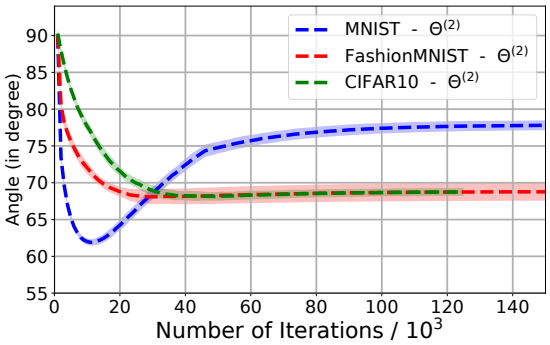


Figure 11: The angle between the feedforward and the transpose of the feedback weights between hidden and output layers (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) as a function of weight update iterations for CorInfoMax-$\mathcal{B}_{1,+}$.

J.7 Compute sources

All the simulations are carried out in an HPC cluster using either a single Tesla T4 or a single Tesla V100 GPU. Although the simulation times are dependent on the memory utilization and number of CPUs, the following list declares the approximate training periods for our experiments:

- One epoch training of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks with single hidden layer on MNIST, Fashion MNIST, and CIFAR10 datasets using batch size 20 takes approximately 1-2 minutes,
- One epoch training of CorInfoMax-$\mathcal{B}_{\infty,+}$ network with two hidden layers on CIFAR10 dataset using batch size 20 takes approximately 2-3 minutes,
- One epoch training of CorInfoMax-$\mathcal{B}_{\infty,+}$ network with two hidden layers on CIFAR100 dataset using batch size 20 takes approximately 3-4 minutes,
- One epoch training of CorInfoMax-$\mathcal{B}_{1,+}$ network with single hidden layer on MNIST, Fashion MNIST, and CIFAR10 datasets using batch size 20 takes approximately 1-2 minutes.

K Ablation studies: the effect of hyperparameters

In this section, we examine the influence of various hyperparameters on the performance of our proposed framework. To achieve this, we conducted a series of simulations over a selected grid of parameters. The outcomes, in terms of training and test accuracy, from these grid-based experiments are detailed here.

K.1 Neural dynamic's learning rate and leakage conductance g_{lk}

Table 7 presents the effect of the neural dynamic's learning rate and the value of g_{lk} on the training and test accuracies for CorInfoMax-$\mathcal{B}_{\infty,+}$ network with single hidden layer on the MNIST image classification task. We observe fairly stable performance over the selected range of values.

Table 7: Train and test accuracies (mean $\pm$ standard deviation of $n = 10$ runs) with respect to the combination of neural dynamic's learning rate and g_{lk} for the MNIST simulations with 2-layer CorInfoMax-$\mathcal{B}_{\infty,+}$ network. The other hyperparameters are as specified in Table 3.

$\mu_{\mathbf{u}}$	g_{lk}	Train accuracy	Test accuracy
$\max\{\frac{0.05}{s\times 10^{-2}+1}, 10^{-3}\}$	0.5	98.915 ± 0.06	97.613 ± 0.11
$\max\{\frac{0.05}{s\times 10^{-2}+1}, 10^{-3}\}$	0.2	98.919 ± 0.04	97.610 ± 0.10
0.05	0.5	98.930 ± 0.05	97.622 ± 0.12
0.05	0.2	98.929 ± 0.04	97.622 ± 0.10

Similarly, Table 8 demonstrates the impression of the learning rate for neural dynamics and the leakage conductance g_{lk} for CorInfoMax-$\mathcal{B}_{1,+}$ with single hidden layer. We note that the training and test accuracy results in the first row of Table 8 is low with high variance. This is due to the fact that one out of ten runs has diverged after epoch 35 for this hyperparameter combination.

Table 8: Train and test accuracies (mean $\pm$ standard deviation of $n = 10$ runs) with respect to the combination of neural dynamic's learning rate and g_{lk} for the MNIST simulations with 2-layer CorInfoMax-$\mathcal{B}_{1,+}$ network. The other hyperparameters are as specified in Table 6.

$\mu_{\mathbf{u}}$	g_{lk}	Train accuracy	Test accuracy
$\max\{\frac{0.05}{s\times 10^{-2}+1}, 10^{-3}\}$	0.5	89.793 ± 28.1	88.891 ± 27.79
$\max\{\frac{0.05}{s\times 10^{-2}+1}, 10^{-3}\}$	0.2	98.634 ± 0.13	97.634 ± 0.12
0.05	0.5	98.698 ± 0.06	97.711 ± 0.10
0.05	0.2	98.695 ± 0.05	97.724 ± 0.08

K.2 Learning rates for synapses and network dynamics

Table 9 reports performance variation for different choices of neural dynamic and synaptic learning rates for CorInfoMax-$\mathcal{B}_{\infty,+}$ with single hidden layer on the Fashion MNIST classification task. Better performance is achieved for relatively higher feedforward synapse learning rates and relatively smaller feedback synapse learning rates.

Table 9: Train and test accuracies (mean $\pm$ standard deviation of $n = 10$ runs) with respect to the combination of feedforward and feedback learning rates, and neural dynamic's learning rate for the Fashion MNIST simulations with 2-layer CorInfoMax-$\mathcal{B}_{\infty,+}$ network. The other hyperparameters are as specified in Table 3.

μ_{ff}	μ_{fb}	$\mu_{\mathbf{u}}$	Train accuracy	Test accuracy
$[0.3, 0.22]$	$[-, 0.07]$	$\max\{\frac{0.07}{s\times 10^{-2}+1}, 10^{-3}\}$	91.468 ± 0.22	88.138 ± 0.28
$[0.3, 0.22]$	$[-, 0.07]$	$\max\{\frac{0.05}{s\times 10^{-2}+1}, 10^{-3}\}$	91.230 ± 0.11	88.140 ± 0.16
$[0.25, 0.15]$	$[-, 0.09]$	$\max\{\frac{0.07}{s\times 10^{-2}+1}, 10^{-3}\}$	89.961 ± 2.79	87.215 ± 2.37
$[0.25, 0.15]$	$[-, 0.09]$	$\max\{\frac{0.05}{s\times 10^{-2}+1}, 10^{-3}\}$	90.530 ± 0.12	87.770 ± 0.23

K.3 Forgetting factor and learning rates

Table 10 shows the impact of forgetting factor $\lambda_{\mathbf{r}}$ together with synaptic learning rates for CorInfoMax-$\mathcal{B}_{\infty,+}$ with single hidden layer on the CIFAR10 classification accuracy. Overall, a slightly improved performance is observed when using relatively higher values of μ_{ff} and relatively smaller values of μ_{fb}.

Table 10: Train and test accuracies (mean $\pm$ standard deviation of $n = 10$ runs) with respect to the combination of feedforward and feedback learning rates, and $\lambda_{\mathbf{r}}$ for the CIFAR10 simulations with 2-layer CorInfoMax-$\mathcal{B}_{\infty,+}$ network. The other hyperparameters are as specified in Table 3.

μ_{ff}	μ_{fb}	$\lambda_{\mathbf{r}}$	Train accuracy	Test accuracy
$[0.08, 0.04]$	$[-, 0.04]$	$1 - 10^{-5}$	64.841 ± 0.17	51.732 ± 0.34
$[0.07, 0.03]$	$[-, 0.05]$	$1 - 10^{-5}$	62.427 ± 0.11	51.065 ± 0.37
$[0.08, 0.04]$	$[-, 0.04]$	$1 - 5 \times 10^{-5}$	64.848 ± 0.16	51.856 ± 0.33
$[0.07, 0.03]$	$[-, 0.05]$	$1 - 5 \times 10^{-5}$	62.456 ± 0.18	51.066 ± 0.32

Likewise, Table 11 illustrates the effect of the variations on the same set of hyperparameters for CorInfoMax-$\mathcal{B}_{1,+}$ with a single hidden layer for CIFAR10 classification task. Comparatively, the resulting performances exhibit a moderate level of stability when compared to the CorInfoMax-$\mathcal{B}_{\infty,+}$ results presented in Table 10.

Table 11: Train and test accuracies (mean $\pm$ standard deviation of $n = 10$ runs) with respect to the combination of feedforward and feedback learning rates, and $\lambda_{\mathbf{r}}$ for the CIFAR10 simulations with 2-layer CorInfoMax-$\mathcal{B}_{1,+}$ network. The other hyperparameters are as specified in Table 6.

μ_{ff}	μ_{fb}	$\lambda_{\mathbf{r}}$	Train accuracy	Test accuracy
$[0.09, 0.07]$	$[-, 0.045]$	$1 - 10^{-5}$	63.005 ± 0.48	51.047 ± 0.40
$[0.095, 0.075]$	$[-, 0.05]$	$1 - 10^{-5}$	63.719 ± 0.56	51.188 ± 0.36
$[0.09, 0.07]$	$[-, 0.045]$	$1 - 5 \times 10^{-5}$	63.003 ± 0.54	51.062 ± 0.32
$[0.095, 0.075]$	$[-, 0.05]$	$1 - 5 \times 10^{-5}$	63.716 ± 0.53	51.106 ± 0.38

K.4 Learning rate decay, free phase iterations, synaptic learning rates

Table 12 investigates the influence of learning rate decay, the number of free phase iterations, and synaptic learning rates on classification accuracy for CorInfoMax-$\mathcal{B}_{\infty,+}$ with two hidden layers on the CIFAR100 classification task. The best train and test results are acquired by utilizing higher values of μ_{ff} and μ_{fb}, along with implementing a learning rate decay during the initial epochs.

Table 12: Train and test accuracies (mean $\pm$ standard deviation of $n = 10$ runs) with respect to the combination of feedforward and feedback learning rates, learning rate decay, and number of iterations for the free phase (T_{free}) for the CIFAR100 simulations with 3-layer CorInfoMax-$\mathcal{B}_{\infty,+}$ network. The other hyperparameters are as specified in Table 4. (In the row stating the lr decay, ep. and O/W means *epoch* and *otherwise*, respectively.)

μ_{ff}	μ_{fb}	lr decay	T_{free}	Train acc.	Test acc.
				Top-1/5	Top-1/5
$[0.16, 0.13, 0.08]$	$[-, 0.06, 0.04]$	$\begin{cases} 1.0 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$	50	$23.9/40.9$	$19.1/36.4$
$[0.16, 0.13, 0.08]$	$[-, 0.06, 0.04]$	$\begin{cases} 0.99 & \text{ep.} < 20 \\ 0.9 & \text{O/W.} \end{cases}$	50	$26.0/42.8$	$20.2/37.6$
$[0.16, 0.13, 0.08]$	$[-, 0.06, 0.04]$	$\begin{cases} 1.0 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$	60	$23.9/41.0$	$19.1/36.5$
$[0.18, 0.15, 0.09]$	$[-, 0.08, 0.06]$	$\begin{cases} 0.99 & \text{ep.} < 20 \\ 0.9 & \text{O/W.} \end{cases}$	50	$27.7/43.5$	$20.8/37.9$

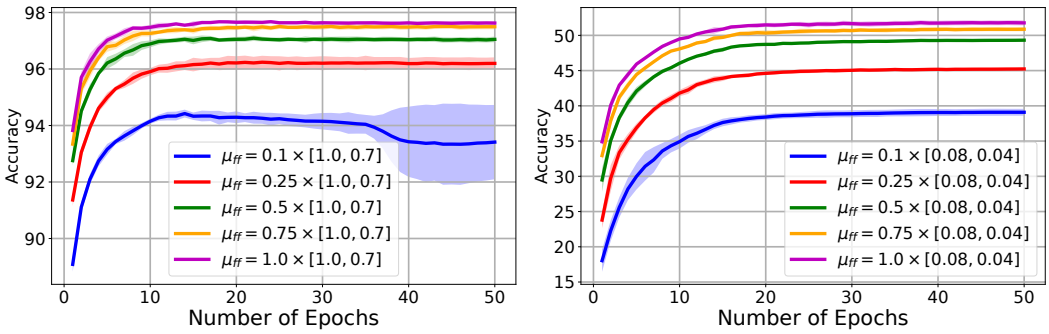
K.5 Larger variations on forward mapping learning rate and neural dynamic learning rate

To assess the impact of larger variations in certain hyperparameters on performance, we conducted two additional ablation studies using both the MNIST and CIFAR10 datasets. These experiments were conducted using a two-layered CorInfoMax-$\mathcal{B}_{\infty,+}$ network. Specifically, we explored variations in the selection of the learning rate for the forward mapping (μ_{ff}) and the initial learning rate for neural dynamics ($\mu_{\mathbf{u}}[1]$).

Figure 12a presents the test accuracy curves for the MNIST classification task, showcasing different selections of μ_{ff} averaged over five runs, with standard deviation envelopes ($\pm$std). The remaining hyperparameters of the CorInfoMax-$\mathcal{B}_{\infty,+}$ network were kept constant at the values reported in Table 3, including the learning rate decay. It is important to note that the specific value of $\mu_{ff} = [1.0, 0.7]$ corresponds to the value reported in Table 3 for our experiments. Our observations indicate that as the forward mapping learning rate approaches this reported value, the accuracy curve exhibits gradual improvement. Conversely, for relatively small values of μ_{ff}, the resulting accuracy is lower and exhibits higher variance.

Figure 12b presents the same experiment conducted for the CIFAR10 classification task. Once again, we observe that as μ_{ff} approaches the value reported in Table 3, the accuracy improves.

In addition, we conducted experiments to investigate the impact of varying the initial learning rate for neural dynamics, denoted as $\mu_{\mathbf{u}}[1]$. Figure 13 presents test accuracy curves for the MNIST classification task with two-layered CorInfoMax-$\mathcal{B}_{\infty,+}$, which were obtained by averaging results from five runs and displaying standard deviation envelopes ($\pm$std). The remaining hyperparameters were held constant at the values specified in Table 3. In Table 3, we define $\mu_{\mathbf{u}}[s]$ as $\max\{\frac{\mu_{\mathbf{u}}[1]}{s\times 10^{-2}+1}, 10^{-3}\}$. Consequently, for this experiment, we applied the same decay rule to all selected values of $\mu_{\mathbf{u}}[1]$. Notably, our results indicate that $\mu_{\mathbf{u}}[1]$ values of 0.05 or 0.075 consistently yield reliable accuracy outcomes, while other values fail to provide dependable accuracy estimates.



(a) (b)

Figure 12: Ablation study on μ_{ff} variations for two-layered CorInfoMax-$\mathcal{B}_{\infty,+}$ networks. (a) Convergence of test accuracy across different selections of μ_{ff} for the MNIST classification task (averaged over $n = 5$ runs associated with the corresponding $\pm$ std envelopes). (b) Convergence of test accuracy across different selections of μ_{ff} for the CIFAR-10 classification task (averaged over $n = 5$ runs associated with the corresponding $\pm$ std envelopes).

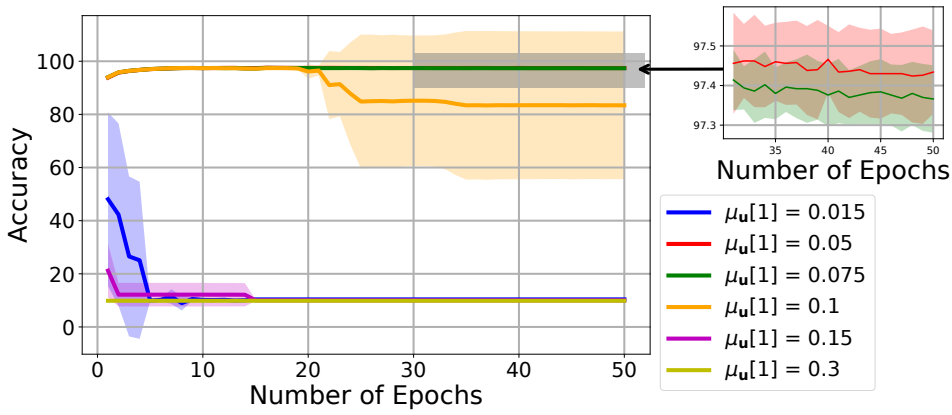


Figure 13: Ablation study on $\mu_{\mathbf{u}}[1]$ variations: Test accuracy curves in a two-layered CorInfoMax-$\mathcal{B}_{\infty,+}$ network for MNIST classification (averaged over $n = 5$ runs associated with the corresponding $\pm$ std envelopes). Convergence to approximately 97.5% accuracy is observed for $\mu_{\mathbf{u}}[1]$ values of 0.05 and 0.075.

Similarly, we conducted the same experiment using a two-layered CorInfoMax-$\mathcal{B}_{\infty,+}$ network for the CIFAR10 classification task. Figure 14 illustrates the test accuracy curves obtained for the CIFAR10 classification task, representing the average results of five runs with standard deviation envelopes ($\pm$std). The remaining hyperparameters were held constant at the values specified in Table 3. It is worth noting that, for the CIFAR10 task, we did not implement decay rule for the neural dynamics learning rate, i.e., $\mu_{\mathbf{u}}[s] = \mu_{\mathbf{u}}[1] \quad \forall s$. Figure 14 demonstrates that our method achieves convergence to approximately 52% accuracy when $\mu_{\mathbf{u}}[1]$ falls within the range of 0.05 to 0.1. However, when $\mu_{\mathbf{u}}[1]$ is set to 0.015, the accuracy slightly decreases, converging to around 49.5%. Notably, relatively higher values of $\mu_{\mathbf{u}}[1]$, such as 0.15 and 0.3, lead to divergence

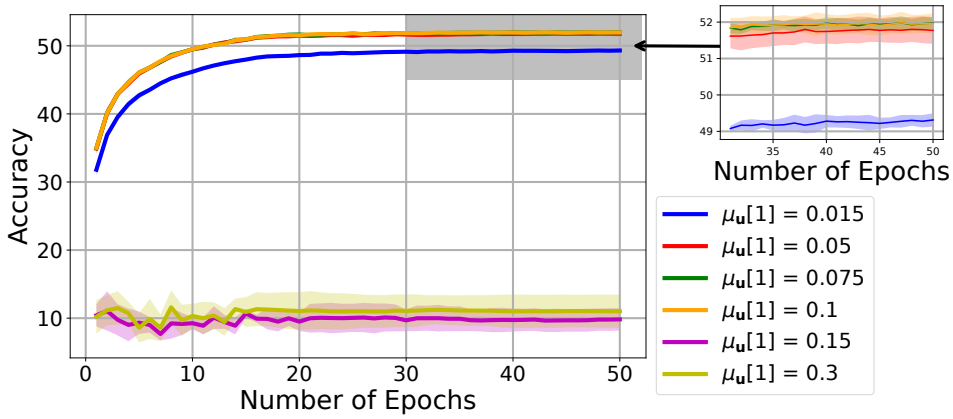


Figure 14: Ablation study on $\mu_{\mathbf{u}}[1]$ variations: Test accuracy curves in a two-layered CorInfoMax-$\mathcal{B}_{\infty,+}$ network for CIFAR10 classification (averaged over $n = 5$ runs associated with the corresponding $\pm$ std envelopes). Convergence to approximately 52% accuracy is observed for $\mu_{\mathbf{u}}[1]$ values between 0.05 and 0.1.