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.

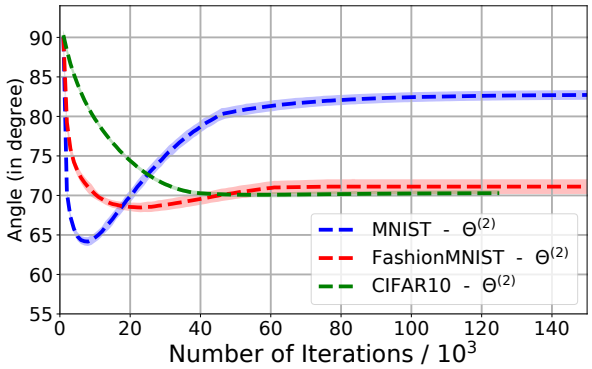


Figure 2: The angle between the feedforward and the transpose of the feedback weights between hidden and output layers (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) as a function of weight update iterations for CorInfoMax-$\mathcal{B}_{\infty,+}$.

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 $\mathcal{P} = \mathcal{B}_{\infty,+}$.

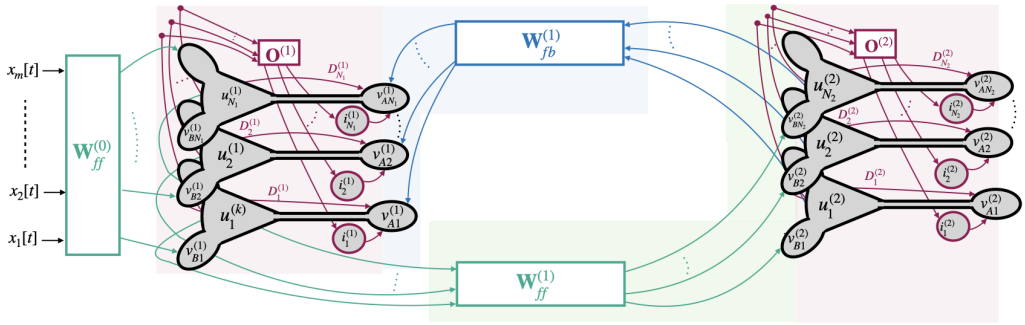


Figure 3: Two layer correlative information maximization based neural network with non-negative antisparsity constraint: $\mathbf{0} \preccurlyeq \mathbf{r}^{(l)} \preccurlyeq \mathbf{1}, l = 1, 2$.

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.

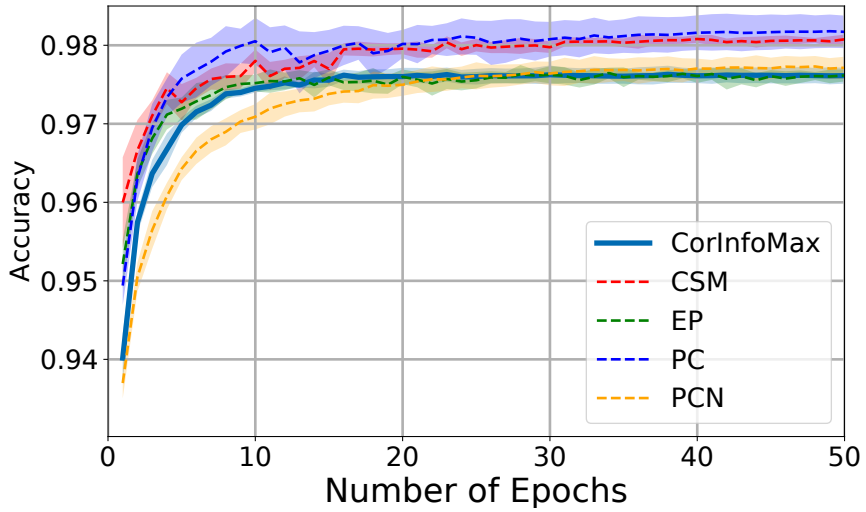


Figure 4: Test accuracy convergences of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks and other (CSM, EP, PC, and PC-Nudge) algorithms as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the MNIST dataset.

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.

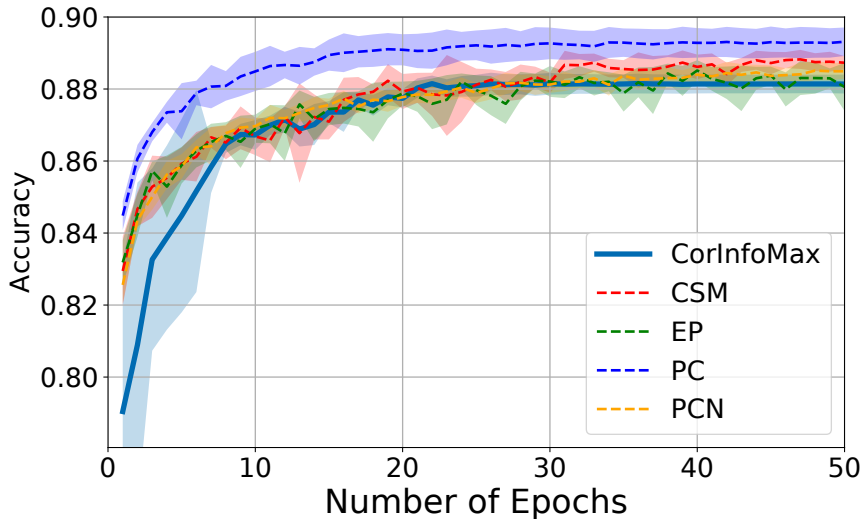


Figure 5: Test accuracy convergences of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks and other (CSM, EP, PC, and PC-Nudge) algorithms as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the Fashion MNIST dataset.

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.

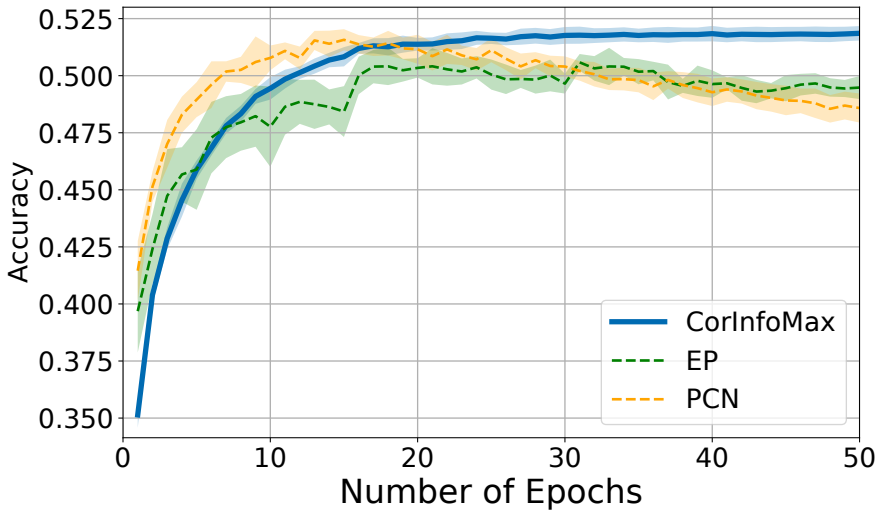


Figure 6: Test accuracy convergences of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks and other (CSM, EP, PC, and PC-Nudge) algorithms as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the CIFAR10 dataset.

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	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$	$\begin{cases} 0.99 & \text{ep.} < 20 \\ 0.9 & \text{O/W.} \end{cases}$

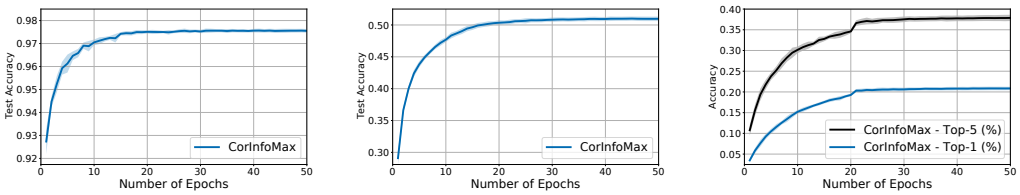
J.5.2 Test accuracy results

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.

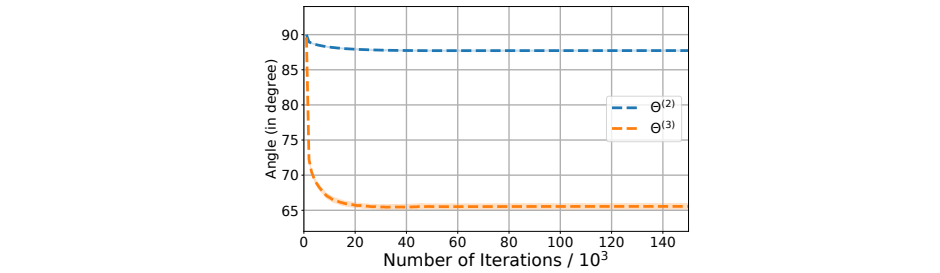


(a) (b) (c)

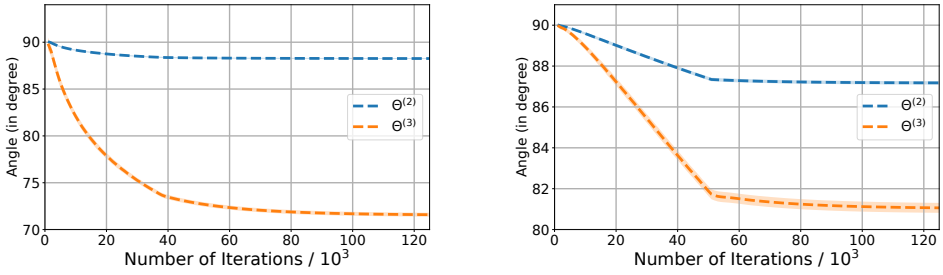
Figure 7: Test accuracy convergence of CorInfoMax-$\mathcal{B}_{\infty,+}$ network with two hidden layers as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the (a) MNIST dataset, (b) CIFAR10 dataset, and (c) CIFAR100 dataset (Top-1(%) and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

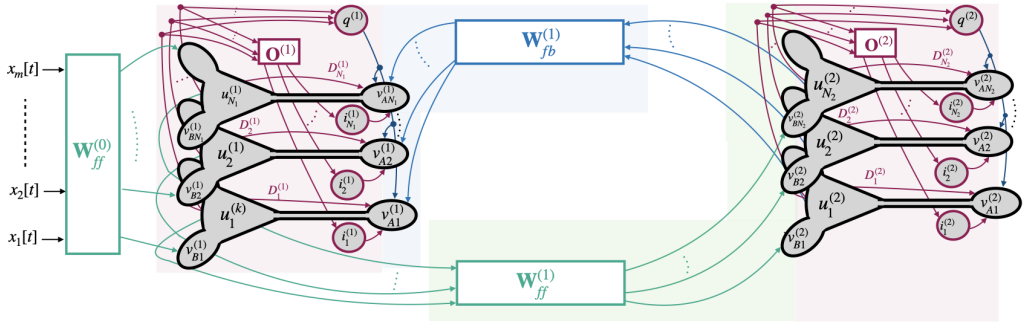


Figure 9: Two layer correlative information maximization based neural network with non-negative sparsity constraint: $\|\mathbf{r}^{(l)}\|_1 \leq 1,$ and $\mathbf{r}^{(l)} \succcurlyeq \mathbf{0}, l = 1, 2$. Interneurons with activations $q^{(1)}$ and $q^{(2)}$ are to impose ℓ_1-norm constraints for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

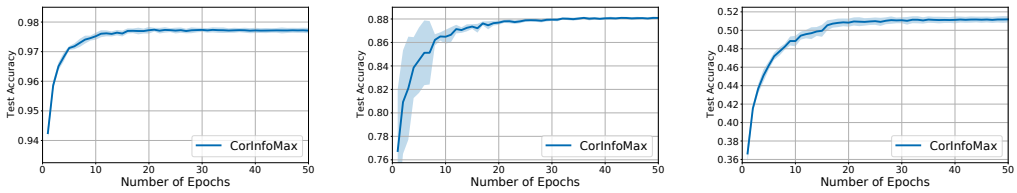
J.6.2 Hyperparameters

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.

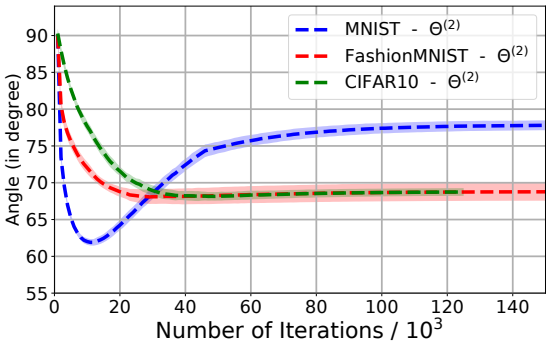


Figure 11: The angle between the feedforward and the transpose of the feedback weights between hidden and output layers (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) as a function of weight update iterations for CorInfoMax-$\mathcal{B}_{1,+}$.

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	$27.7/43.5$	$20.8/37.9$

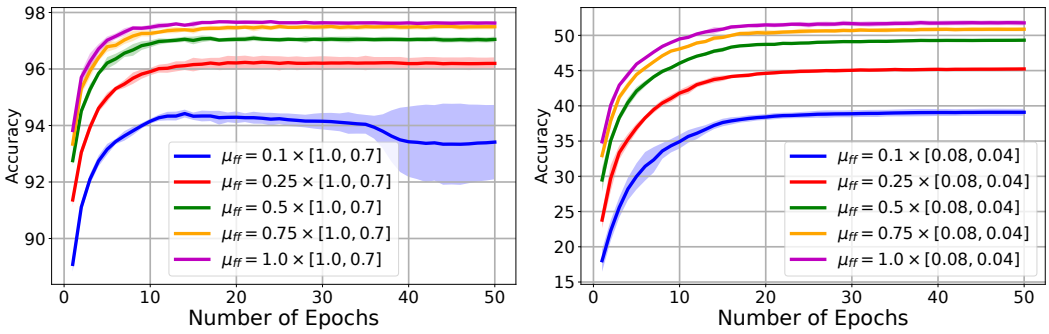
K.5 Larger variations on forward mapping learning rate and neural dynamic learning rate

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).

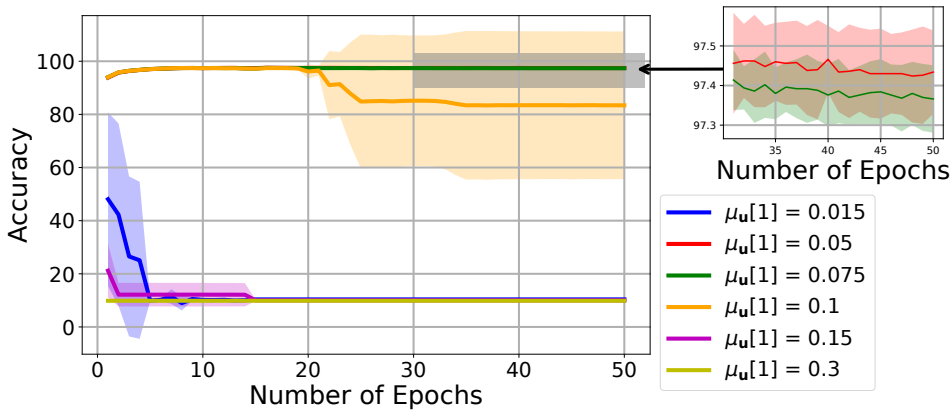


Figure 13: Ablation study on $\mu_{\mathbf{u}}[1]$ variations: Test accuracy curves in a two-layered CorInfoMax-$\mathcal{B}_{\infty,+}$ network for MNIST classification (averaged over $n = 5$ runs associated with the corresponding $\pm$ std envelopes). Convergence to approximately 97.5% accuracy is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

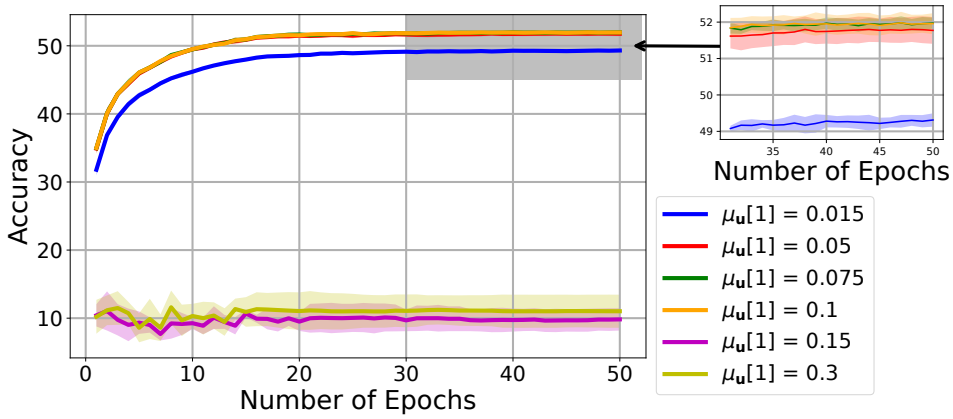


Figure 14: Ablation study on $\mu_{\mathbf{u}}[1]$ variations: Test accuracy curves in a two-layered CorInfoMax-$\mathcal{B}_{\infty,+}$ network for CIFAR10 classification (averaged over $n = 5$ runs associated with the corresponding $\pm$ std envelopes). Convergence to approximately 52% accuracy is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