Determinant Maximization Criterion as a Normative Principle for Biologically Plausible Credit Assignment

by

Barışcan Bozkurt

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Electrical and Electronics Engineering

KOÇ ÜNİVERSİTESİ

July 26, 2023

Determinant Maximization Criterion as a Normative Principle for Biologically Plausible Credit Assignment

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Barışcan Bozkurt

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Alper T Erdoğan (Advisor)

_____

Prof. Alper Demir

_____

Prof. Burak Acar

Date: _____

To the science community

ABSTRACT

Determinant Maximization Criterion as a Normative Principle for Biologically Plausible Credit Assignment
Barışcan Bozkurt
Master of Science in Electrical and Electronics Engineering
July 26, 2023

We demonstrate that the determinant maximization (Det-Max) criterion arises as a normative principle to construct biologically-plausible learning algorithms for both unsupervised and supervised methods. The importance of determinant maximization lies in its central role within structured matrix factorization frameworks, enabling the extraction of hidden information from input data. We apply this criterion to formulate principled optimization problems that address specific issues in an online setting, leading to the development of neural networks with local learning updates. Our primary objectives involve solving two central problems in signal processing and machine learning: blind source separation and supervised learning. For blind source separation, we propose two normative biologically-plausible methods that incorporate the determinant maximization criterion and assume the source vectors are sufficiently scattered in their domains. The first method employs a cascade of two weighted similarity matching constraints. Each constraint aims to equalize the generalized inner products among output vectors and their corresponding input vectors. We prove that a batched optimization problem based on weighted similarity matching achieves the same global optimum as the determinant maximization optimization, resulting in perfect separation. The corresponding online optimization can be realized by a two-layered neural network with piece-wise linear activations and local Hebbian learning. The second method we propose involves maximizing correlative mutual information from input vectors to output vectors. This approach relies solely on second-order statistics of the inputs and outputs, eliminating the need for computationally expensive higher-order statistics or joint probability density function estimates. This novel framework serves as a unified approach for developing biologically-plausible neural networks for various unsupervised data decomposition methods, aiming to obtain structured latent representa-

tions. Furthermore, we demonstrate that augmenting the correlative information maximization objective with mean squared error for fitting to labeled data provides an alternative normative approach to explain signal propagation in supervised biological neural networks. The coordinate descent optimization of this novel objective establishes a neural network structure that mimics biologically realistic networks of multi-compartment pyramidal neurons with dendritic processing and lateral inhibitory neurons. Importantly, our approach naturally resolves the weight transport problem associated with the backpropagation algorithm, which has been a significant critique of training conventional neural networks from a biologically-plausible perspective. We achieve this by leveraging two alternative forms of correlative information measures. In brief, this thesis proposes several normative methods for establishing biologically-plausible neural networks for various tasks. We provide diverse numerical experiments to showcase the effectiveness of each proposed method in comparison to recent approaches.

ÖZETÇE

Biyolojik Olarak Makul Kredi Tahsisi için Normatif Bir İlke Olarak Determinant Maksimizasyon Kriteri

Barışcan Bozkurt

Elektrik ve Elektronik Mühendisliği, Yüksek Lisans

Temmuz 2023

Determinant maksimizasyon (Det-Max) kriterinin, hem denetimsiz hem de denetimli yöntemler için biyolojik olarak makul öğrenme algoritmaları oluşturmak için normatif bir ilke olarak ortaya çıktığını gösteriyoruz. Determinant maksimizasyonun önemi, yapılandırılmış matris çarpanlarına ayırma yöntemlerindeki merkezi rolünde yatmaktadır ve girdi verilerinden gizli bilgilerin çıkarılmasını sağlar. Bu kriteri, çevrimiçi bir ortamda belirli sorunları ele alan ve yerel öğrenme güncellemeleriyle sinir ağlarının geliştirilmesine yol açan ilkeli optimizasyon problemlerini formüle etmek için uyguluyoruz. Başlıca hedeflerimiz, sinyal işleme ve makine öğrenimindeki iki merkezi sorunu çözmeyi içerir: kör kaynak ayırma ve denetimli öğrenme. Kör kaynak ayrımı için, determinant maksimizasyon kriterini içeren ve kaynak vektörlerin etki alanlarında yeterince dağıldığını varsayan, biyolojik olarak makul iki normatif yöntem öneriyoruz. İlk yöntem, iki ağırlıklı benzerlik eşleştirme kısıtlamasından oluşan bir kaskad kullanır. Her kısıtlama, çıktı vektörleri ve karşılık gelen girdi vektörleri arasındaki genelleştirilmiş iç çarpımları eşitlemeyi amaçlar. Ağırlıklı benzerlik eşleşmesine dayalı bir optimizasyon probleminin, determinant maksimizasyon optimizasyonu ile aynı global optimuma ulaştığını ve mükemmel bir ayrımla sonuçlandığını kanıtladık. Karşılık gelen çevrimiçi optimizasyon, parçalı lineer aktivasyonlar ve yerel Hebbian öğrenme ile iki katmanlı bir sinir ağı tarafından gerçekleştirilebilir. Önerdiğimiz ikinci yöntem, girdi vektörlerinden çıktı vektörlerine karşılıklı korelatif bilgiyi maksimize etmeyi içerir. Bu yaklaşım, girdi ve çıktıların yalnızca ikinci dereceden istatistiklerine dayanır ve hesaplama açısından pahalı olan yüksek dereceli istatistiklere veya ortak olasılık yoğunluk fonksiyonu tahminlerine olan ihtiyacı ortadan kaldırır. Bu yeni çerçeve, yapılandırılmış gizli temsiller elde etmeyi amaçlayan çeşitli denetimsiz veri ayrıştırma yöntemleri için biyolojik olarak makul sinir ağları geliştirmeye yönelik birleşik bir yaklaşım olarak hizmet eder. Ayrıca, etiketli ver-

ilere uydurmak için korelatif bilgi maksimizasyon hedefini ortalama karesel hatayla birleştirmenin, denetimli biyolojik sinir ağlarında sinyal yayılımını açıklamak için alternatif bir normatif yaklaşım sağladığını gösteriyoruz. Bu yeni hedefin optimizasyonu, dendritik işleme ve yanal inhibitör nöronlara sahip çok bölmeli piramidal nöronların biyolojik olarak gerçekçi ağlarını taklit eden bir sinir ağı yapısı kurar. Daha da önemlisi, yaklaşımımız, geleneksel sinir ağlarını biyolojik olarak makul bir bakış açısıyla eğitmenin önemli bir eleştirisi olan geri yayılım algoritmasıyla ilişkili ağırlık taşıma problemini doğal olarak çözmektedir. Bunu, iki alternatif korelatif bilgi ölçüsü biçiminden yararlanarak başarıyoruz. Kısaca, bu tez, çeşitli görevler için biyolojik olarak makul sinir ağları kurmak için birkaç normatif yöntem önermektedir. Son yaklaşımlara kıyasla önerilen her yöntemin etkinliğini göstermek için çeşitli sayısal deneyler sağlıyoruz.

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my advisor Prof. Alper T Erdoğan for his never-ending support and invaluable guidance throughout our research journey. His mentorship has played a vital role in developing my disciplined research abilities during my master's study, and I am sincerely thankful for his continuous encouragement and our close friendship. Furthermore, Prof. Erdoğan's profound knowledge, passion for research, and dedication to academic excellence have been a remarkable model not only for my eagerness to pursue knowledge but also for my personal and professional growth.

I am also deeply grateful for the tremendous support provided by Assistant Prof. Cengiz Pehlevan. We collaborated as co-authors on three publications, which are highlighted in Chapter 4, 5, and 6 of this thesis. His invaluable contributions greatly enhanced the research content and guided us in all aspects of presenting our work and advancements.

I extend heartfelt thanks to jury members Prof. Alper Demir and Prof. Burak Acar for accepting the invitation to serve on the jury and providing valuable feedbacks and questions on this work.

I would like to express my sincere appreciation to Deniz Aktaş for her lovely and invaluable support throughout my research progress.

Furthermore, I would like to acknowledge the support received from the KUIS AI Center Research Award. Lastly, I would like to thank Ateş İsfendiyaroğlu for his contribution in [Bozkurt et al., 2023a].

TABLE OF CONTENTS

List of Tables **xv**

List of Figures **xviii**

Abbreviations **xxvii**

Chapter 1: Introduction **1**

Chapter 2: Related Work **5**

2.1 Blind Source Separation 5

2.2 Correlative Information Maximization Based BSS Approach 10

2.3 Biologically Plausible Neural Networks for BSS 13

2.4 Information Maximization for Unsupervised and Supervised Learning 15

2.5 Biologically Plausible Supervised Learning 17

Chapter 3: Identifiable Polytope Characterization for Polytopic Matrix Factorization **20**

3.1 Abstract 20

3.2 Introduction 20

3.3 Polytopic Matrix Factorization and Identifiability 21

3.4 Identifiable Polytope Characterization 26

3.5 Numerical Example 30

3.6 Conclusion 31

Chapter 4: Biologically Plausible Determinant Maximization Neural Networks for Blind Separation of Correlated Sources **32**

4.1 Abstract . . . 32
4.2 Introduction . . . 32
4.3 Problem Statement: Blind Source Separation . . . 34
4.4 Weighted Similarity Matching-Based Optimization Formulation of Determinant Maximization Criterion . . . 37
4.5 Biologically-plausible Neural Networks for WSM-based BSS . . . 40
4.5.1 Online Optimization Setting for WSM-based BSS . . . 41
4.5.2 Description of Network Dynamics for Bounded Anti-sparse Sources . . . 43
4.5.3 Description of Network Dynamics for Bounded Nonnegative Anti-sparse Sources . . . 46
4.5.4 Description of Network Dynamics for Nonnegative Sparse Sources 48
4.5.5 Description of Network Dynamics for Sparse Sources . . . 49
4.6 Description of Network Dynamics for Unit Simplex Sources . . . 50
4.7 Description of Network Dynamics for Sources with Mixed Attributes . 52
4.8 Numerical Experiments . . . 53
4.8.1 On the Computation of Neural Dynamics . . . 54
4.8.2 Synthetically Correlated Source Separation with Nonnegative Anti-sparse Sources . . . 55
4.8.3 Synthetically Correlated Source Separation with Anti-sparse Sources . . . 56
4.8.4 Image Separation . . . 58
4.8.5 Sparse Source Separation . . . 60
4.8.6 Sparse Dictionary Learning . . . 61
4.8.7 Source Separation with Mixed Latent Attributes . . . 63
4.8.8 Digital Communication Example: 4-PAM Modulation Scheme 67
4.9 Conclusion . . . 68

Chapter 5: Correlative Information Maximization Based Biologically Plausible Neural Networks for Correlated Source

Separation **69**

5.1 Abstract 69

5.2 Introduction 70

5.3 Blind Source Separation Setting 72

5.4 Method: Biologically Plausible Neural Networks for Correlative Information Maximization 74

5.4.1 Online Optimization Setting for Correlative Mutual Information Maximization 74

5.4.2 Description of the Network Dynamics for Sparse Sources . . . 75

5.4.3 Description of the Network Dynamics for a Canonical Polytope Representation 78

5.4.4 Description of the Network Dynamics for Antisparse Sources . 80

5.4.5 Description of the Network Dynamics for Nonnegative Antisparse Sources 81

5.4.6 Description of the Network Dynamics for Nonnegative Sparse Sources 82

5.4.7 Description of the Network Dynamics for Unit Simplex Sources 83

5.4.8 Description of the Network Dynamics for Feature Based Specified Polytopes 84

5.5 Numerical Experiments 87

5.5.1 Synthetically Correlated Source Separation with Antisparse Sources 88

5.5.2 Video Separation 89

5.5.3 Image Separation 91

5.5.4 Sparse Source Separation 94

5.5.5 Nonnegative Sparse Source Separation 97

5.5.6 Simplex Source Separation 98

5.5.7 Source Separation for Polytopes with Mixed Latent Attributes 99

5.5.8 Sparse Dictionary Learning 100

5.5.9 Digital Communication Example: 4-PAM Modulation Scheme 102
5.6 Conclusion . . . 103

Chapter 6: Correlative Information Maximization: A Biologically Plausible Approach to Supervised Deep Neural Networks without Weight Symmetry 105

6.1 Abstract . . . 105
6.2 Introduction . . . 106
6.3 Deep Correlative Information Maximization . . . 108
6.3.1 Network Data Model . . . 108
6.3.2 Correlative Information Maximization Based Signal Propagation 109
6.3.3 Neural Network Formulation based on Information Maximization . . . 113
6.3.4 Learning Dynamics . . . 116
6.4 Alternative Expression for the Feedforward Weight Updates . . . 117
6.5 Employing Sparsity Assumption on Neuronal Activities . . . 119
6.6 On the Asymmetry of Feedforward and Feedback Weights . . . 121
6.6.1 The Importance of the Choice of Alternative CMI Expressions 121
6.6.2 The Analytical Comparison of Forward and Backward Prediction Coefficients . . . 123
6.7 Numerical Experiments . . . 123
6.7.1 On the Computation of Neural Dynamics . . . 124
6.7.2 Image Classification with Two Layered Networks . . . 125
6.7.3 Empirical Angle Between Forward and Backward Weights . . 129
6.7.4 Three Layered CorInfoMax-$\mathcal{B}_{\infty,+}$ Network . . . 130
6.8 Conclusion . . . 133

Chapter 7: Conclusion 134

Bibliography 136

Appendix A: Supplementary on Det-Max WSM Networks 153

A.1 The Simplification of the Similarity Matching Cost Functions 153

A.2 Derivatives of the WSM Cost Function 155

A.3 Hyperparameters Used in the Numerical Experiments 157

A.3.1 Synthetically Correlated Source Separation with Nonnegative Anti-sparse Sources 157

A.3.2 Synthetically Correlated Source Separation with Anti-sparse Sources 158

A.3.3 Image Separation 158

A.3.4 Sparse Source Separation 159

A.3.5 Source Separation with Mixed Latent Attributes 159

A.3.6 Digital Communication Example: 4-PAM Modulation Scheme 161

A.4 Ablation Study on Hyperparameter Selection for Nonnegative Sparse Sources 161

A.5 Discussion on the Complexity of the Proposed Approach 164

Appendix B: Supplementary on CorInfoMax-Based BSS Networks 167

B.1 Gradient Derivations for Online Optimization Objective 167

B.2 Online CorInfoMax Algorithm Implementations for Special Source Domains 170

B.3 Hyperparameter Selections 173

B.4 Ablation Studies on Hyperparameter Selections 173

B.5 Ablation Studies on the Effect of Number of Mixtures 176

B.6 Computational Complexity of the Proposed Approach 177

Appendix C: Supplementary on CorInfoMax-Based Supervised Networks 180

C.1 Test Accuracy Results of CorInfoMax-$\mathcal{B}_{1,+}$ Network 180

C.2 Hyperparameters 182

C.3 Ablation Studies on the Effect of Hyperparameters 183

C.3.1 Neural Dynamic's Learning Rate and Leakage Conductance g_{lk} 183
C.3.2 Learning Rates for Synapses and Network Dynamics 185
C.3.3 Forgetting Factor and Learning Rates 185
C.3.4 Learning Rate Decay, Free Phase Iterations, and Synaptic Learning Rates 187
C.4 Compute Sources 188

LIST OF TABLES

4.1 Illustrative source domains depicted in Figure 2.1 and their respective output dynamics. 47

4.2 Averaged SINR results for sparse source separation using the WSM, PMF, and LD-InfoMax algorithms. 62

5.1 Source separation averaged SINR results on $\mathcal{P}_{\text{ex}}$ for the CorInfoMax (ours), CorInfoMax Canonical (ours) LD-InfoMax, and PMF algorithms (averaged for 50 realizations). 100

6.1 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 [Qin et al., 2021], while the remaining results stem from our own simulations. 126

6.2 Test accuracy ($n = 10$ runs $\pm$ stddev) of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks with two hidden layers. 130

B.1 Example source domains and the corresponding CorInfoMax network dynamics. 171

B.2 CorInfoMax network hyperparameter selections for special source domains in 5.3. 174

B.3 CorInfoMax network hyperparameter selections for the polytope with mixed latent attributes in (5.24) and application examples sparse dictionary learning, digital communication, video separation, and image separation. 175

C.1 Description of the hyperparameter notations. 182

C.2 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.) 183
C.3 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.) 184
C.4 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.) 185
C.5 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 C.2. 186
C.6 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 C.4. 186
C.7 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 C.2. 187
C.8 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 C.2. 187

C.9 Train and test accuracies (mean ± 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 C.4. 188

C.10 Train and test accuracies (mean ± 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 C.3. (In the row stating the lr decay, ep. and O/W means *epoch* and *otherwise*, respectively.) 189

LIST OF FIGURES

2.1 Provided are examples of source domains that result in identifiable generative models, as presented in the aforementioned list. 7
2.2 The three-dimensional visualization of the geometry depicting the sufficient scattering conditions for the unit simplex and polytopes. . . 8

3.1 $\mathcal{B}_1$ and Corresponding $G_{\mathcal{P}}$ 27
3.2 Execution time of the proposed Algorithm 1 and brute-force search. As expected, Algorithm 1 scales exponentially with respect to the number of vertices. 31

4.1 Det-Max WSM neural networks for blind source separation. The neural network accepts a mixed input $\mathbf{x}$ and generates latent components $\mathbf{y}$ as its output. The configuration of the output layer is influenced by the selected source domain. Inhibitory neurons establish connections between mutually sparse components within the output layer. 34
4.2 Nonnegative clipping function utilized for elementwise projection onto the set $\mathcal{B}_{\infty,+}$. 48
4.3 Det-Max WSM neural network for the extraction of nonnegative anti-sparse sources ($\mathcal{P} = \mathcal{B}_{\infty,+}$). 48
4.4 Det-Max WSM neural network for the extraction of nonnegative sparse sources ($\mathcal{P} = \mathcal{B}_{1,+}$). 50
4.5 Det-Max WSM neural network for the extraction of sparse sources ($\mathcal{P} = \mathcal{B}_1$). 51
4.6 Det-Max WSM neural network for the extraction of unit-simplex sources ($\mathcal{P} = \Delta$). 52
4.7 Det-Max WSM neural network corresponding to the polytope in (4.21). 54

4.8 The SINR performances of the WSM, NSM, ICA, LD-InfoMax, and PMF algorithms, evaluated with respect to the correlation parameter ρ 56
4.9 The SINR performances of the WSM, BSM, ICA, LD-InfoMax, and PMF algorithms, evaluated with respect to the correlation parameter ρ 57
4.10 (a) Original RGB images. 58
4.10 (b) mixture RGB images, (c) ICA outputs, (d) NSM outputs, (e) WSM outputs, (f) LD-InfoMax Outputs. 59
4.10 (g) PMF Outputs. 60
4.11 The convergence curve of the sparse-WSM algorithm for the SINR, considering an input SNR level of 30dB. The curve is represented by the mean (solid line) along with the 25th and 75th percentile envelopes. 60
4.12 The output SINR as a function of the input SNR level for the sparse-WSM algorithm, displayed using the mean (solid line) along with the 25th and 75th percentile envelopes. 61
4.13 Dictionary obtained from the natural image patches through the sparse-WSM Network. 62
4.14 Example behaviors of SINR, component SNR values, and diagonal weights of $\mathbf{D}_1$ and $\mathbf{D}_2$ for a single experiment corresponding to the polytope in (4.21) 64
4.15 The convergence curve of SINR for the experiments corresponding to the polytope defined in (4.21), represented by the mean (solid line) along with the 25th and 75th percentile envelopes. 64
4.16 The convergence curve of SINR for the experiments corresponding to the polytope defined in (4.23), represented by the mean (solid line) along with the 25th and 75th percentile envelopes. 65
4.17 The convergence curve of SINR for the experiments corresponding to the polytope defined in Equation (4.24), represented by the mean (solid line) along with the 25th and 75th percentile envelopes. 66

4.18 The SINR convergence curve for 4-PAM digital communication signals, depicted by the mean (solid line) along with the 25th and 75th percentile envelopes. 67

5.1 CorInfoMax BSS neural networks designed for two distinct canonical source domain representations. In both representations, x_i's and y_i's denote the input (mixtures) and output (separator) signals, respectively. The feedforward weights are denoted by $\mathbf{W}$, while e_i's represent the errors between the transformed inputs and outputs. The lateral weights at the output, $\mathbf{B_y}$, correspond to the inverse of the output autocorrelation matrix. For canonical form (a), the Lagrangian interneurons, λ_i, impose source domain constraints. The matrix $\mathbf{A}_{\mathcal{P}}$ ($\mathbf{A}_{\mathcal{P}}^T$) represents the feedforward (feedback) connections between the outputs and interneurons. In canonical form (b), the interneurons on the right enforce sparsity constraints on specific subsets of outputs. . 72

5.2 Sparse CorInfoMax Network: (a) three-layer (b) two-layer. x_i's and y_i's denote the input (mixtures) and output (separator) signals, respectively. The feedforward weights are denoted by $\mathbf{W}$, while e_i's in the three-layer implementation (on the left) represent the errors between the transformed inputs and outputs. $\mathbf{B_Y}$ in (a) correspond to the inverse of the output autocorrelation matrix and represents lateral weights at the output. In representation (b), the output lateral weights, denoted by $\mathbf{M_Y}$, are a diagonally modified form of the inverse of the output correlation. In both representations, the rightmost interneuron enforces the sparsity constraint. 79

5.3 Two-Layer antisparse CorInfoMax Network. x_i's represent the inputs (mixtures) and y_i's represent the output (separator) signals. $\mathbf{W}$ denotes the feedforward weights. e_i's represent the errors between the transformed inputs and outputs. $\mathbf{B_Y}$ represents the lateral weights at the output, which is the inverse of the output autocorrelation matrix. The output nonlinearities are clipping functions. 81

5.4 Two-Layer nonnegative antisparse CorInfoMax Network. x_i's represent the inputs (mixtures) and y_i's represent the output (separator) signals. $\mathbf{W}$ denotes the feedforward weights. e_i's represent the errors between the transformed inputs and outputs. $\mathbf{B_Y}$ represents the lateral weights at the output, which is the inverse of the output autocorrelation matrix. The output nonlinearities are nonnegative clipping functions. 82

5.5 Three-Layer nonnegative sparse CorInfoMax Network. x_i's represent the inputs (mixtures) and y_i's represent the output (separator) signals. $\mathbf{W}$ denotes the feedforward weights. e_i's represent the errors between the transformed inputs and outputs. $\mathbf{B_Y}$ represents the lateral weights at the output, which is the inverse of the output autocorrelation matrix. The output nonlinearities are ReLU functions. The leftmost interneuron enforces sparsity constraints on the outputs. . . 83

5.6 Three-Layer CorInfoMax Network for unit simplex sources. x_i's represent the inputs (mixtures) and y_i's represent the output (separator) signals. $\mathbf{W}$ denotes the feedforward weights. e_i's represent the errors between the transformed inputs and outputs. $\mathbf{B_Y}$ represents the lateral weights at the output, which is the inverse of the output autocorrelation matrix. The output nonlinearities are ReLU functions. The leftmost interneuron enforces sparsity constraints on the outputs through inhibition. 85

5.7 The averaged SINR performances of CorInfoMax (ours), LD-InfoMax, PMF, ICA-InfoMax, NSM, and BSM, obtained from 100 realizations, are plotted on the y-axis against the correlation factor ρ on the x-axis. The SINR vs. ρ curves are depicted for two different source domain configurations: (a) nonnegative antisparse ($\mathcal{B}_{\ell_\infty,+}$), (b) antisparse ($\mathcal{B}_{\ell_\infty}$) source domains. 89
5.8 Video separation example: the final frames of (a) sources, (b) mixtures, and (c) the outputs of the nonnegative antisparse CorInfoMax network. 90
5.9 Image separation example: (a) Original RGB images, (b) mixture RGB images, (c) ICA outputs. 93
5.9 (d) NSM outputs, (e) WSM outputs, (f) LD-InfoMax Outputs, (g) CorInfoMax (ours) Outputs. 94
5.10 SINR performances of CorInfoMax (ours), LD-InfoMax, PMF, and WSM (averaged over 50 realizations) for sparse sources: (a) output SINR results (vertical axis) with respect to input SNR levels (horizontal axis), (b) SINR (vertical axis) convergence plot as a function of iterations (horizontal axis) of sparse CorInfoMax for the 30 dB SNR level (mean solid line and standard deviation envelopes). 96
5.11 SINR performances of CorInfoMax (ours), LD-InfoMax, and PMF with different distribution selections for the mixing matrix entries and for sparse sources. The horizontal axis represents different distribution choices, and the vertical axis represents the SINR levels. . . 96
5.12 SINR performances of CorInfoMax (ours), LD-InfoMax, and WSM for nonnegative sparse sources: (a) the output SINR (vertical axis) results with respect to the input SNR levels (horizontal axis), (b) the SINR (vertical axis) convergence plot as a function of iterations (horizontal axis) of nonnegative sparse CorInfoMax for the 30dB SNR level (mean solid line and standard deviation envelopes). 97

5.13 SINR performances of CorInfoMax (ours) and LD-InfoMax (averaged over 50 realizations) for unit simplex sources: (a) the output SINR (vertical axis) results with respect to the input SNR levels (horizontal axis), (b) SINR (vertical axis) convergence plot as a function of iterations (horizontal axis) of simplex-CorInfoMax for 30dB SNR level (mean solid line and standard deviation envelopes). 98
5.14 SINR performances of CorInfoMax (ours) and LD-InfoMax with different distribution selections for the mixing matrix entries (averaged over 50 realizations) and unit simplex sources. The horizontal axis represents different distribution choices, and the vertical axis represents the SINR levels. 99
5.15 SINR performances of CorInfoMax networks on $\mathcal{P}_{\text{ex}}$ with mean-solid line with standard deviation envelope (averaged over 50 realization) for the polytope with mixed latent attributes: (a) SINR convergence curve for CorInfoMax feature-based polytope formulation with 30dB mixture SNR, (b) SINR convergence curve for CorInfoMax canonical formulation with 30dB mixture SNR, (c) SINR convergence curve for CorInfoMax feature-based polytope formulation with 40dB mixture SNR, (d) SINR convergence curve for CorInfoMax canonical formulation with 40dB SNR. 101
5.16 Sparse dictionary learned by sparse CorInfoMax from natural image patches. 102
5.17 The SINR (vertical axis) convergence curve for the 4-PAM digital communication example as a function of iterations (horizontal axis) which is averaged over 100 realizations: mean-solid line with standard deviation envelope. 103

6.1 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 $H^{(\epsilon_k)}(\mathbf{r}^{(k+1)})$ ($H^{(\epsilon_k)}(\mathbf{r}^{(k)})$). 108

6.2 Two layer correlative information maximization based neural network with non-negative antisparsity constraint: $0 \leq \mathbf{r}^{(l)} \leq 1, l = 1, 2$. 119

6.3 Two layer correlative information maximization based neural network with non-negative sparsity constraint: $\|\mathbf{r}^{(l)}\|_1 \leq 1$, and $\mathbf{r}^{(l)} \geq \mathbf{0}, l = 1, 2$. Interneurons with activations $q^{(1)}$ and $q^{(2)}$ are to impose ℓ_1-norm constraints for both layers. 122

6.4 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. 127

6.5 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. 128

6.6 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. 128

6.7 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 a) CorInfoMax-$\mathcal{B}_{\infty,+}$, b) CorInfoMax-$\mathcal{B}_{1,+}$. 129

6.8 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 MNIST dataset. 131

6.9 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 CIFAR10 dataset. 131

6.10 Top-1(%) and Top-5(%) test accuracy convergences 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 CIFAR100 dataset. 132

6.10 The angle between the feedforward and the transpose of the feedback weights (based on (6.46) 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. 133

A.1 SINR performances of WSM Det-Max networks for different hyper-parameter selections (averaged over 50 realizations, mean solid lines with 25/75-percentile envelopes): (a) averaged SINR performance with respect to $1 - \lambda_{\text{SM}}$, (b) averaged SINR performance with respect to initial $\mathbf{D}_1$, (c) averaged SINR performance with respect to $\mu_{\mathbf{D}_1}$, (d) averaged SINR performance with respect to initial $1 - \gamma^2$. . 164

B.1 CorInfoMax ablation studies on the hyperparameter selections of $\mu_{\mathbf{W}}$ and $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}$: mean SINR with standard deviation envelopes corresponding to 50 realizations. (a) ablation study of $\mu_{\mathbf{W}}$ on uncorrelated nonnegative antisparse sources, (b) ablation study of $\mu_{\mathbf{W}}$ on correlated (with $\rho = 0.6$) nonnegative antisparse sources, (c) ablation study of $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}$ on uncorrelated nonnegative antisparse sources, (d) ablation study of $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}$ on correlated (with $\rho = 0.6$) nonnegative antisparse sources, ... 177

B.2 CorInfoMax ablation studies on the impact of number of mixtures with fixed number of sources (5 sources). (a) illustrates the SINR performances of nonnegative antisparse CorInfoMax network as a function of number of mixtures, (b) illustrates the SINR performances of sparse CorInfoMax network as a function of number of mixtures. . . . 178

C.1 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 MNIST dataset. 180

C.2 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 Fashion MNIST dataset. 181

C.3 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 CIFAR10 dataset. 181

ABBREVIATIONS

BSS	Blind Source Separation
PMF	Polytopic Matrix Factorization
SSC	Sufficiently Scattered Condition
MVIE	Maximum Volume Inscribed Ellipsoid
CMI	Correlative Mutual Information
ICA	Independent Component Analysis
BCA	Bounded Component Analysis
SCA	Sparse Component Analysis
NMF	Nonnegative Matrix Factorization
SSMF	Simplex Structured Matrix Factorization
BPTT	Backpropagation Through Time
SST	Somatostatin-expressing

Chapter 1

INTRODUCTION

In computational neuroscience, a fundamental question revolves around understanding the brain's circuit motifs and plasticity rules that facilitate learning. It also investigates whether neurons approximate gradient-based algorithms akin to machine learning approaches [Richards et al., 2019]. This critical matter, known as the **credit assignment** problem in both neuroscience and machine learning, involves determining the contribution of neural network parameters to the output and how they should be updated to enhance performance. While the specific algorithms employed by the brain to tackle complex problems remain largely elusive, numerous methods have been proposed in the literature based on normative methods and anatomical observations. **Normative principles** are mathematically-motivated methods and they involve formulating principled objectives that adhere to biological constraints such as local learning. While many of these normative methods do not comprehensively encompass experimental biological findings, they offer rigorous theoretical explanations that incorporate essential evidences, thus fostering the development of more biologically-realistic frameworks. As a result, the literature on normative principles for proposing biologically-plausible learning frameworks is rapidly expanding, with the proposed approaches demonstrating closer alignment with experimental observations of the learning in the brain.

For a neural network to be considered a **biologically plausible** model, it must meet certain important requirements, namely operating in an online setting [Pehlevan and Chklovskii, 2019] and adapting its synaptic connections based on local learning rules [Dayan and Abbott, 2005]. The term "online setting" refers to the processing of input data in the neural network one data point at a time, producing

the corresponding output before proceeding to the next input vector. Additionally, local learning involves updating the weights of the network based on the activity of the pre- and post-synaptic neurons, meaning a weight is updated using the activations of the two neurons that it connects. Recent artificial neural networks do not fulfill these requirements and exhibit significant differences from natural biological networks. If artificial neural networks can achieve superior performance, it raises the question: *why should we care about computational models of the brain and develop biologically plausible algorithms*? A concise answer to these questions can be approached from two perspectives. Firstly, artificial neural networks have been inspired by the learning process of the brain [Rosenblatt, 1958], driving significant advancements in artificial intelligence for several decades [Zador et al., 2022]. Secondly, the development of such biologically plausible frameworks enables faster and energy-efficient implementations in future analog neuromorphic systems like Loihi [Davies et al., 2021].

This thesis demonstrates that the **determinant-maximization** (Det-Max) criterion, which has proven to be a powerful tool for structured matrix factorization frameworks, arises as a normative approach to develop both unsupervised and supervised biologically-plausible neural networks. Before delving into the problem of biological plausibility, the thesis addresses a geometric issue in Polytopic Matrix Factorization (PMF) [Tatli and Erdogan, 2021b] which also employs the Det-Max objective. Specifically, an algorithmic approach is proposed to tackle the identifiable polytope characterization problem in Chapter 3, and this method is subsequently utilized in the following chapters dealing with the blind source separation problem. The remaining chapters of the thesis focus on biologically-plausible neural networks for blind source separation and supervised learning problems. The proposed approaches begin with a principled optimization based on determinant maximization for the specific problem, which is then formulated in an online manner to derive neural networks with local learning rules. The contributions of this thesis can be summarized as follows:

- We propose an algorithm for the identifiable polytope characterization problem

using an existing graph automorphism framework. We prove that only checking the generator set of the linear automorphism group of the polytope, which can be obtained by graph algorithms, is sufficient to determine the identifiability of the polytope. Through numerical experiments, we demonstrate the efficiency and superiority of our method compared to brute-force approaches (presented in Chapter 3 and published in [Bozkurt and Erdogan, 2022]).

- We introduce biologically-plausible neural networks for blind separation of uncorrelated and correlated sources based on weighted similarity matching constraints in combination with the Det-Max criterion. We prove that the batch optimization problem with two weighted similarity matching constraints achieves the same global optimum with the determinant maximization optimization. Hence, when the source samples are sufficiently scattered within their domains, the optimization process leads to perfect separation. Our normative approach provides a versatile framework to generate biologically-plausible neural networks that are capable of handling various source domain constraints (presented in Chapter 4 and published in [Bozkurt et al., 2022]).

- We formulate the problem of correlative mutual information maximization (CorInfoMax) for the blind source separation problem. By deriving two- and three-layered neural networks through an online formulation, we naturally incorporate local learning rules, ensuring biologically-plausible network structures. This novel framework solely relies on the second order statistics of the input and output vectors while offering an information-theoretic interpretation for neural networks used in blind source separation. We present a unified derivation method for developing neural networks that can separate sources from identifiable polytopes (presented in Chapter 5 and published in [Bozkurt et al., 2023a]).

- We demonstrate that augmenting the CorInfoMax objective with a mean squared loss function for labeled data leads to the development of supervised biologically-plausible neural networks with multi-compartment pyramidal neu-

ron models. Furthermore, by employing two alternative yet equivalent expressions for correlative information, we offer a natural solution to the weight transport problem, a primary critique regarding the biological plausibility of the backpropagation algorithm for training neural networks. Our proposed framework presents a normative principle that elucidates the signal propagation in both the forward and backward paths of biological neural networks (presented in Chapter 6 and published in [Bozkurt et al., 2023b]).

Each contribution chapter (including Chapter 3, 4, 5, and 6) provides a more elaborated discussion on the specific advancements made in that chapter. Additionally, Chapter 2 is dedicated to discussing the related work on blind source separation, information maximization, and biologically-plausible neural networks for both unsupervised and supervised tasks.

Chapter 2

RELATED WORK

2.1 Blind Source Separation

Blind source separation (BSS) is the problem of extracting the original sources from their linear mixtures in an unsupervised manner, i.e., without knowing the ground truth mixing function and sources. This unsupervised retrieval of latent components from their linear mixtures is a fundamental challenge [Comon and Jutten, 2010, Cichocki et al., 2009], with wide-ranging applications in machine learning, signal processing, and computational neuroscience [Bell and Sejnowski, 1995, Makeig et al., 1996, Kreutz-Delgado et al., 2003, Mukamel et al., 2009, Pehlevan et al., 2017b]. If we represent the source vectors by $\mathbf{s} \in \mathbb{R}^n$, the mixtures, which are the observations, can be expressed as a linear combination of the sources using an unknown mixing matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \in \mathbb{R}^m. \tag{2.1}$$

Throughout this thesis, we will only consider the overdetermined case with $m \geq n$ and presume that the mixing matrix is full column rank. The objective of blind source separation (BSS) frameworks is to learn a separator matrix $\mathbf{W} \in \mathbb{R}^{n \times m}$ to estimate the original sources:

$$\mathbf{y} = \mathbf{W}\mathbf{x} \approx \mathbf{s}. \tag{2.2}$$

For a data-driven approach, we can assume that we have N observations $\mathbf{x}(i)$, $i = 1, 2, \ldots, N$ corresponding to N ground truth source vectors $\mathbf{s}(i)$, $i = 1, 2, \ldots, N$. Therefore, we can express the mixing and un-mixing equations in the matrix equation form as

$$\mathbf{X} = \mathbf{A}\mathbf{S} \quad \text{and} \quad \mathbf{Y} = \mathbf{W}\mathbf{X}, \tag{2.3}$$

where $\mathbf{X} = \begin{bmatrix} \mathbf{x}(1) & \mathbf{x}(2) & \dots & \mathbf{x}(N) \end{bmatrix} \in \mathbb{R}^{m \times N}$, $\mathbf{S} = \begin{bmatrix} \mathbf{s}(1) & \mathbf{s}(2) & \dots & \mathbf{s}(N) \end{bmatrix} \in \mathbb{R}^{n \times N}$, and $\mathbf{Y} = \begin{bmatrix} \mathbf{y}(1) & \mathbf{y}(2) & \dots & \mathbf{y}(N) \end{bmatrix} \in \mathbb{R}^{n \times N}$. While the formulation of the BSS problem in this manner may initially appear ill-posed, it is mathematically provable to solve this seemingly impossible problem through some assumptions regarding the generative model for the observations (mixtures).

The most widely used assumption is the mutual independence of components, which is the basis for the independent component analysis (ICA) framework [Comon, 1994, Bell and Sejnowski, 1995, Hyvärinen et al., 2001, Comon and Jutten, 2010]. This stochastic criterion has its roots in the computational modeling of the vertebrate nervous system. Besides its biological inspiration, ICA offers a tractable venue for algorithm development and analysis based on the mutual information cost function or its surrogates.

In scenarios where the assumption of independence is not applicable, alternative forms of side information regarding the generative model can be leveraged. Numerous algorithms have been proposed in the literature for separating linearly mixed and correlated sources, and these algorithms rely on specific assumptions about the domains of latent vectors, resulting in alternative criteria for their extraction. For example, the assumptions about the domains of latent vectors give rise to distinct principles for their extraction.

- Sparse component analysis (SCA) methods frequently utilize the assumption that the latent vectors in the generative model are contained within the unit ℓ_1-norm ball [Georgiev et al., 2005, Donoho, 2006, Rozell et al., 2007, Elad, 2010, Babatas and Erdogan, 2018, Babatas and Erdogan, 2020].

- The identifiability of latent variables in antisparse bounded component analysis (BCA) approaches relies on the assumption that the latent variables belong to the ℓ_∞-norm ball [Cruces, 2010, Erdogan, 2013, Inan and Erdogan, 2014].

- Simplex structured matrix factorization (SSMF) methods leverage the assumption that the latent vectors belong to the unit simplex for their retrieval [Chan et al., 2011, Lin et al., 2018, Lin et al., 2015].

- Nonnegative matrix factorization (NMF) algorithms utilize the assumption that the latent vectors belong to the nonnegative orthant to identify them from their linear transformations [Chen, 1984, Paatero and Tapper, 1994, Cichocki et al., 2009, Fu et al., 2019].

- The recently introduced polytopic matrix factorization (PMF) framework expands the identifiability-enabling domains to include infinitely many polytopes that satisfy a specific symmetry restriction [Tatli and Erdogan, 2021a, Tatli and Erdogan, 2021b, Bozkurt and Erdogan, 2022].

(a) $\mathcal{B}_{\ell_1}$ (sparse)

(b) $\mathcal{B}_{\ell_\infty}$ (anti-sparse)

(c) Δ (unit simplex)

(d) $\mathcal{B}_{\ell_\infty,+}$ (nonnegative anti-sparse)

(e) $\mathcal{B}_{\ell_1,+}$ (nonnegative sparse)

Figure 2.1: Provided are examples of source domains that result in identifiable generative models, as presented in the aforementioned list.

The set-membership-based approaches listed above rely on the assumption that the latent vectors in the generative model exhibit sufficient spread across their domain to capture its shape, thus ensuring identifiability [Lin et al., 2015, Tatli and Erdogan, 2021b]. The Det-Max approach is a common algorithmic method that leverages these assumptions by dispersing latent vector estimates across their presumed domain [Schachtner et al., 2011, Lin et al., 2015, Erdogan, 2013, Inan and Erdogan, 2014, Babatas and Erdogan, 2018].

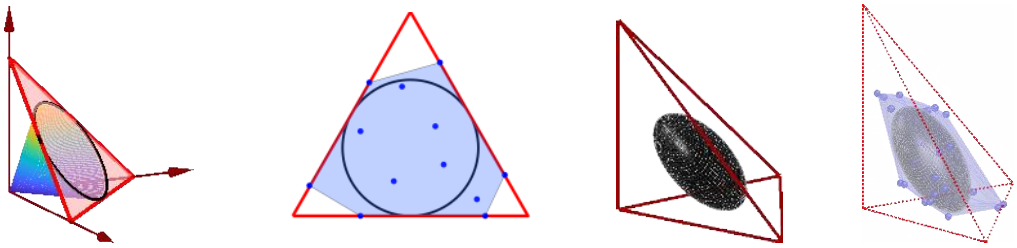
This is achieved by training the separator to maximize the (log)-determinant of the sample correlation matrix for the separator outputs. Therefore, the corresponding objective function can be written as $J(\mathbf{W}) = \log(\det(\hat{\mathbf{R}}_\mathbf{y}))$, where $\hat{\mathbf{R}}_\mathbf{y}$ is the sample correlation matrix $\hat{\mathbf{R}}_\mathbf{y} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{y}(i)\mathbf{y}(i)^T = \frac{1}{N}\mathbf{Y}\mathbf{Y}^T$. During the training process, an additional constraint is imposed to ensure that the separator

outputs lie within the presumed source domain, i.e. $\mathbf{y}(i) \in \mathcal{P} \quad \forall i$. Consequently, the corresponding optimization problem can be formulated as described in [Fu et al., 2019, Tatli and Erdogan, 2021b]

$$\underset{\mathbf{Y} \in \mathbb{R}^{n \times N}}{\text{maximize}} \quad \log(\det(\mathbf{Y}\mathbf{Y}^T)) \tag{2.4a}$$

$$\text{subject to} \quad \mathbf{y}(i) \in \mathcal{P}, i = 1, \ldots, N, \tag{2.4b}$$

where constant $\frac{1}{N}$ term is ignored. In this context, the determinant of the correlation matrix serves as a measure of spread for the output samples. When the original source samples $\mathbf{s}(1), \ldots, \mathbf{s}(N)$ are sufficiently scattered within the source domain $\mathcal{P}$, it has been proved that the global solution of this optimization problem achieves perfect separation [Fu et al., 2018, Fu et al., 2019, Tatli and Erdogan, 2021b]. Both the polytopic matrix factorization and simplex structured matrix factorization frameworks put forward specific sufficient scattering conditions for the latent vectors, ensuring their identifiability under the determinant maximization criterion. In the following, we provide a review of these underlying conditions to ensure a thorough discussion.



(a) The unit simplex, denoted by Δ, and the second-order cone, denoted by $\mathcal{C}$.

(b) An illustration showcasing the sufficient scattering condition for samples within the unit simplex, Δ.

(c) An illustration of the polytope $\mathcal{B}_{1,+}$ and its MVIE $\mathcal{E}_{\mathcal{B}_1}$.

(d) An example showing the sufficient scattering condition for $\mathcal{B}_{1,+}$ samples.

Figure 2.2: The three-dimensional visualization of the geometry depicting the sufficient scattering conditions for the unit simplex and polytopes.

Sufficient scattering condition for unit simplex sources: An earlier theory for la-

tent factor identifiability in the context of SSMF required that the generative latent vector samples include the vertices of the unit simplex. [Donoho and Stodden, 2003]. This assumption, so-called "separability" or "local dominance" in the context of SSMF, was later replaced by a weaker condition known as the "sufficiently scattered condition" (SSC) in [Fu et al., 2018]. This new condition utilizes the second-order cone.

$$\mathcal{C} = \{\mathbf{s} \,|\, \mathbf{1}^T\mathbf{s} \geq \sqrt{n-1}\|\mathbf{s}\|_2, \mathbf{s} \in \mathbb{R}^n\},$$

which is illustrated in Figure 2.2a alongside the unit simplex Δ used as a reference object for defining the SSC. The SSC proposed in [Fu et al., 2018] for SSMF requires the conic hull of the simplex samples to contain $\mathcal{C}$, i.e.,

$$\text{cone}(\{\mathbf{s}(1), \mathbf{s}(2), \ldots, \mathbf{s}(N)\}) \supseteq \mathcal{C}.$$

Let $\mathcal{A}_\Delta$ denote the affine hull of Δ. Figure 2.2b provides an illustration of this requirement specifically applied to $\mathcal{A}_\Delta$: the red triangle represents the boundary of Δ, the blue dots represent sufficiently scattered samples from Δ, the black circle and the blue polyhedral region represent the boundary of $\mathcal{C}$ and the conic hull of sufficiently scattered samples from Δ restricted to $\mathcal{A}_\Delta$, respectively. Furthermore, there is an additional requirement that the boundaries of Δ and $\text{cone}(\{\mathbf{s}(1), \mathbf{s}(2), \ldots, \mathbf{s}(N)\}) \cap \mathcal{A}_\Delta$ intersect the boundary of $\mathcal{C} \cap \mathcal{A}_\Delta$ at the same points.

Sufficient scattering condition for polytopic sources: The reference [Tatli and Erdogan, 2021b] presents a similar SSC for polytopic sources, where the reference object for SSC is the maximum volume inscribed ellipsoid (MVIE) denoted by $\mathcal{E}_\mathcal{P}$ of the polytope $\mathcal{P}$. Figure 2.2c provides an illustration of the MVIE (black ellipsoid) for the polytope selection $\mathcal{P} = \mathcal{B}_{1,+}$, where the edges of the polytope are represented by red lines. The SSC for polytopic sources requires that the convex hull of the samples from polytope contains the MVIE of the polytope, i.e.,

$$\text{conv}(\{\mathbf{s}(1), \mathbf{s}(2), \ldots, \mathbf{s}(N)\}) \supseteq \mathcal{E}_\mathcal{P}.$$

This condition is illustrated in Figure 2.2d, where the dots are sufficiently scattered samples and the blue polyhedral region represents their convex hull. Addi-

tionally, the SSC in [Tatli and Erdogan, 2021b] requires that the boundaries of $\text{conv}(\{\mathbf{s}(1), \mathbf{s}(2), \ldots, \mathbf{s}(N)\})$ and $\mathcal{P}$ intersect $\mathcal{E}_{\mathcal{P}}$ at the exact same points.

As discussed earlier in this section, PMF extends the domains that enable identifiability to polytopes that meet specific symmetry restrictions. However, determining whether a given polytope satisfies these symmetry restrictions poses a challenging problem. In Chapter 3, we tackle this challenge by presenting an efficient algorithm that utilizes graph-theoretic tools to characterize identifiable polytopes. This algorithm provides a means to verify and identify the polytopes that satisfy the necessary symmetry constraints. Furthermore, in Chapter 4 and 5, we present neural network frameworks for solving the blind source separation problem. These frameworks are designed based on the determinant maximization criterion. Consequently, our proposed methods enable the blind separation of correlated sources from an identifiable domain, particularly under the assumption of sufficient scattering conditions.

2.2 Correlative Information Maximization Based BSS Approach

In this section, we provide a review of the correlative entropy and correlative mutual information measures for the BSS setting, as introduced in [Erdogan, 2022]. For a finite set of vectors $\mathbb{X} = \{\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(N)\} \subset \mathbb{R}^m$ with a sample correlation matrix $\hat{\mathbf{R}}_{\mathbf{x}} = \frac{1}{N}\mathbf{X}\mathbf{X}^T$, where $\mathbf{X}$ is defined as $\mathbf{X} = \begin{bmatrix} \mathbf{x}(1) & \mathbf{x}(2) & \ldots & \mathbf{x}(N) \end{bmatrix}$, the deterministic correlative entropy is defined in [Erdogan, 2022] as

$$H^{(\epsilon)}(\mathbf{X}) = \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{x}} + \epsilon\mathbf{I}) + \frac{m}{2}\log(2\pi e). \tag{2.5}$$

Here, $\epsilon > 0$ is a small number introduced to avoid the expression approaching $-\infty$. When the sample correlation $\hat{\mathbf{R}}_{\mathbf{x}}$ in this expression is replaced with the true covariance, then $H^{(0)}(\mathbf{x})$ coincides with the Shannon differential entropy for a Gaussian vector $\mathbf{x}$ [Cover, 1999]. Furthermore, the deterministic joint correlative entropy of

two sets of vectors $\mathbb{X} \subset \mathbb{R}^m$ and $\mathbb{Y} \subset \mathbb{R}^n$ can be defined as

$$\begin{aligned}
H^{(\epsilon)}(\mathbf{X}, \mathbf{Y}) &= \frac{1}{2} \log\det(\hat{\mathbf{R}}_{\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}} + \epsilon\mathbf{I}) + \frac{m+n}{2}\log\det(2\pi e) \\
&= \frac{1}{2}\log\det\left(\begin{bmatrix}\hat{\mathbf{R}}_{\mathbf{x}} + \epsilon\mathbf{I} & \hat{\mathbf{R}}_{\mathbf{xy}} \\ \hat{\mathbf{R}}_{\mathbf{yx}} & \hat{\mathbf{R}}_{\mathbf{y}} + \epsilon\mathbf{I}\end{bmatrix}\right) + \frac{m+n}{2}\log\det(2\pi e) \\
&= \frac{1}{2}\log\Big(\det(\hat{\mathbf{R}}_{\mathbf{x}} + \epsilon\mathbf{I})\det(\hat{\mathbf{R}}_{\mathbf{y}} + \epsilon\mathbf{I} - \hat{\mathbf{R}}_{\mathbf{xy}}^T(\hat{\mathbf{R}}_{\mathbf{x}} + \epsilon\mathbf{I})^{-1}\hat{\mathbf{R}}_{\mathbf{xy}})\Big) \\
&+ \frac{m+n}{2}\log\det(2\pi e) \\
&= \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{x}} + \epsilon\mathbf{I}) + \frac{m}{2}\log(2\pi e) + \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{e}} + \epsilon\mathbf{I}) + \frac{n}{2}\log(2\pi e) \\
&= H^{(\epsilon)}(\mathbf{X}) + H^{(\epsilon)}(\mathbf{Y}|_L\mathbf{X}) \qquad (2.6)
\end{aligned}$$

where $\hat{\mathbf{R}}_{\mathbf{e}} = \hat{\mathbf{R}}_{\mathbf{y}} - \hat{\mathbf{R}}_{\mathbf{xy}}^T(\hat{\mathbf{R}}_{\mathbf{x}} + \epsilon\mathbf{I})^{-1}\hat{\mathbf{R}}_{\mathbf{xy}}$, $\hat{\mathbf{R}}_{\mathbf{xy}} = \frac{1}{N}\boldsymbol{X}\boldsymbol{Y}^T = \hat{\mathbf{R}}_{\mathbf{yx}}^T$, and and the third equality is obtained by writing the determinant in terms of a principle submatrix and its Schur's complement [Kailath et al., 2000]. In (2.6), the notation $\mathbf{Y}|_L\mathbf{X}$ is used to signify $H^{(\epsilon)}(\mathbf{Y}|_L\mathbf{X}) \neq H^{(\epsilon)}(\mathbf{Y}|\mathbf{X})$ as the latter requires the use of $\hat{\mathbf{R}}_{\mathbf{y}|\mathbf{x}}$ instead of $\hat{\mathbf{R}}_{\mathbf{e}}$. Moreover, $H^{(\epsilon)}(\mathbf{Y}|_L\mathbf{X})$ corresponds to the log-determinant of the error sample correlation of the best linear minimum mean squared estimate (MMSE) of $\mathbf{y}$ from $\mathbf{x}$. We can simply verify this in the zero-mean and noiseless case by considering the MMSE estimate $\hat{\mathbf{y}} = \mathbf{W}\mathbf{x}$ for which the solution is provided by $\mathbf{W} = \mathbf{R}_{\mathbf{yx}}\mathbf{R}_{\mathbf{x}}^{-1} = \mathbf{R}_{\mathbf{xy}}^T\mathbf{R}_{\mathbf{x}}^{-1}$ [Kailath et al., 2000]. As a result, $\mathbf{R}_{\hat{\mathbf{y}}} = \mathbb{E}[\hat{\mathbf{y}}\hat{\mathbf{y}}^T] = \mathbb{E}[\mathbf{R}_{\mathbf{xy}}^T\mathbf{R}_{\mathbf{x}}^{-1}\mathbf{x}\mathbf{x}^T\mathbf{R}_{\mathbf{x}}^{-1}\mathbf{R}_{\mathbf{xy}}] = \mathbf{R}_{\mathbf{xy}}^T\mathbf{R}_{\mathbf{x}}^{-1}\mathbf{R}_{\mathbf{xy}}$. Therefore, the error vector is defined as $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$ and its correlation matrix can be found as desired, i.e., $\mathbf{R}_{\mathbf{e}} = \mathbf{R}_{\mathbf{y}} - \mathbf{R}_{\hat{\mathbf{y}}} = \mathbf{R}_{\mathbf{y}} - \mathbf{R}_{\mathbf{xy}}^T\mathbf{R}_{\mathbf{x}}^{-1}\mathbf{R}_{\mathbf{xy}}$. Furthermore, note that this derivation naturally leads to the expressions for the conditional correlative entropy measures for which we can express as

$$H^{(\epsilon)}(\mathbf{Y}|_L\mathbf{X}) = \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{y}} - \hat{\mathbf{R}}_{\mathbf{xy}}^T(\hat{\mathbf{R}}_{\mathbf{x}} + \epsilon\mathbf{I})^{-1}\hat{\mathbf{R}}_{\mathbf{xy}} + \epsilon\mathbf{I}) + \frac{n}{2}\log(2\pi e) \qquad (2.7)$$

$$H^{(\epsilon)}(\mathbf{X}|_L\mathbf{Y}) = \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{x}} - \hat{\mathbf{R}}_{\mathbf{yx}}^T(\hat{\mathbf{R}}_{\mathbf{y}} + \epsilon\mathbf{I})^{-1}\hat{\mathbf{R}}_{\mathbf{yx}} + \epsilon\mathbf{I}) + \frac{m}{2}\log(2\pi e) \qquad (2.8)$$

Utilizing the alternative Schur's complement in (2.6), we can also show 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 informa-

tion for $\mathbf{X}$ and $\mathbf{Y}$ can be defined as follows

$$I^{(\epsilon)}(\mathbf{X},\mathbf{Y}) = H^{(\epsilon)}(\mathbf{Y}) - H^{(\epsilon)}(\mathbf{Y}|_L\mathbf{X}) \tag{2.9}$$

$$= H^{(\epsilon)}(\mathbf{X}) - H^{(\epsilon)}(\mathbf{X}|_L\mathbf{Y}) \tag{2.10}$$

$$= H^{(\epsilon)}(\mathbf{X}) + H^{(\epsilon)}(\mathbf{Y}) - H^{(\epsilon)}(\mathbf{X},\mathbf{Y}) \tag{2.11}$$

In more explicit terms, the correlative mutual information can be expressed using two alternative yet equivalent expressions:

$$\begin{aligned} I^{(\epsilon)}(\mathbf{X},\mathbf{Y}) &= \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{y}} + \epsilon\mathbf{I}) - \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{y}} - \hat{\mathbf{R}}_{\mathbf{xy}}^T(\hat{\mathbf{R}}_{\mathbf{x}} + \epsilon\mathbf{I})^{-1}\hat{\mathbf{R}}_{\mathbf{xy}} + \epsilon\mathbf{I}) \\ &= \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{x}} + \epsilon\mathbf{I}) - \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{x}} - \hat{\mathbf{R}}_{\mathbf{yx}}^T(\hat{\mathbf{R}}_{\mathbf{y}} + \epsilon\mathbf{I})^{-1}\hat{\mathbf{R}}_{\mathbf{yx}} + \epsilon\mathbf{I}). \end{aligned} \tag{2.12}$$

The correlative mutual information $I^{(\epsilon)}(\mathbf{X},\mathbf{Y})$ serves as a measure of the degree of correlation or linear association between the random vectors $\mathbf{X}$ and $\mathbf{Y}$ [Erdogan, 2022]. It is also worth mentioning that the reference also defines the stochastic correlative entropy and mutual information measures for the random vectors using true correlation matrices, i.e.,

$$H^{(\epsilon)}(\mathbf{x}) = \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}} + \epsilon\boldsymbol{I}) + \frac{m}{2}\log(2\pi e), \tag{2.13}$$

$$H^{(\epsilon)}(\mathbf{x}|_L\mathbf{y}) = \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}} - \mathbf{R}_{\mathbf{yx}}^T(\mathbf{R}_{\boldsymbol{y}} + \epsilon\boldsymbol{I})^{-1}\mathbf{R}_{\mathbf{yx}} + \epsilon\boldsymbol{I}) + \frac{m}{2}\log(2\pi e), \tag{2.14}$$

$$I^{(\epsilon)}(\mathbf{x},\mathbf{y}) = H^{(\epsilon)}(\mathbf{x}) - H^{(\epsilon)}(\mathbf{x}|_L\mathbf{y}), \tag{2.15}$$

where $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^n$ are random vectors, $\mathbf{R}_{\mathbf{x}} = \mathbb{E}[\mathbf{x}\mathbf{x}^T]$, $\mathbf{R}_{\mathbf{xy}} = \mathbb{E}[\mathbf{x}\mathbf{y}^T]$, and $\mathbb{E}[.]$ denotes the expectation operation. We will utilize the stochastic entropy and information definitions in (2.13), (2.14), (2.15) in Chapter 6.

Based on the correlative information measure, [Erdogan, 2022] brings a statistical interpretation to the PMF framework while relating the source separation problem with maximum information transfer from mixtures to the source estimate outputs. According to this approach, the correlative mutual information between the input and output is maximized, under the constraint that the outputs are in the presumed source domain. The corresponding optimization setting is given by

$$\underset{\mathbf{Y} \in \mathbb{R}^{n \times N}}{\text{maximize}} \quad \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{y}} + \epsilon\mathbf{I}) - \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{y}} - \hat{\mathbf{R}}_{\mathbf{yx}}(\epsilon\mathbf{I} + \hat{\mathbf{R}}_{\mathbf{x}})^{-1}\hat{\mathbf{R}}_{\mathbf{yx}}^{T} + \epsilon\mathbf{I}) \tag{2.16a}$$

$$\text{subject to} \quad \mathbf{y}(i) \in \mathcal{P}, i = 1, \ldots, N, \tag{2.16b}$$

which has been named the (batched) CorInfoMax-based BSS framework. Similar to the PMF framework, the CorInfoMax approach assumes that the source vectors are drawn from a presumed polytope $\mathcal{P}$. Furthermore, the CorInfoMax-based BSS optimization problem in (2.16) is capable of separating correlated sources, since it does not assume any statistical independence or uncorrelatedness on the source vectors.

Chapter 5 and Chapter 6 in this thesis both employ the concept of correlative information to develop biologically plausible neural networks for different tasks. In Chapter 5, correlative information is utilized to design a neural network framework for blind source separation. On the other hand, in Chapter 6, correlative information is leveraged to construct a neural network model for supervised learning, specifically in the context of a multi-compartmental neuron model.

2.3 Biologically Plausible Neural Networks for BSS

Blind source separation (BSS) is a vital process through which the brain extracts latent causes, known as sources, from complex stimuli received by sensory organs. This unsupervised extraction of sources plays a fundamental role in our understanding of the surrounding world [Bell and Sejnowski, 1995, Olshausen and Field, 1996, Bronkhorst, 2000, Lewicki, 2002, Asari et al., 2006, Narayan et al., 2007, Bee and Micheyl, 2008, McDermott, 2009, Mesgarani and Chang, 2012, Golumbic et al., 2013, Isomura et al., 2015]. While the brain consistently performs blind source separation (BSS), the specific algorithms it utilizes remain largely unknown. As a result, researchers have been developing biologically plausible neural network algorithms to shed light on these processes.

Influential studies have demonstrated that visual and auditory cortical receptive

fields can emerge through the application of BSS to natural scenes and sounds [Bell and Sejnowski, 1995, Olshausen and Field, 1996, Lewicki, 2002]. The widespread utilization and ubiquity of BSS in the brain indicates the presence of generic circuit motifs dedicated to this task [Sharma et al., 2000]. Consequently, the field of biologically plausible neural network algorithms for BSS has witnessed significant growth, with contributions from various researchers [Eagleman et al., 2001, Pehlevan et al., 2017b, Isomura and Toyoizumi, 2018, Erdogan and Pehlevan, 2020, Bahroun et al., 2021]. BSS algorithms tackle the challenge of an underdetermined inverse problem by making generative assumptions on the observation signals. Within the context of biologically-plausible BSS algorithms, it is common to assume that the complex stimuli can be represented as linear mixtures of the latent sources. This assumption has proven to be highly valuable and has been applied successfully in modeling various scenarios, such as natural images [Olshausen and Field, 1997, Bell and Sejnowski, 1995] and the responses of the olfactory neurons to complex odorants [Zhang and Sharpee, 2016, Krishnamurthy et al., 2017, Singh et al., 2021]. However, it is important to note that linear mixing alone is insufficient for source identifiability, necessitating the incorporation of additional assumptions to achieve successful separation. Previous biologically plausible algorithms for BSS of linear mixtures have primarily focused on assuming statistical independence or uncorrelatedness of sources [Isomura and Toyoizumi, 2018, Bahroun et al., 2021, Lipshutz et al., 2022, Pehlevan et al., 2017b, Erdogan and Pehlevan, 2020]. While these assumptions have been effective in certain scenarios, they limit the domain of applicability of these algorithms. Real-world situations often involve potentially dependent or correlated sources, which require novel approaches to achieve successful separation. An instance of biologically plausible neural networks capable of separating correlated sources can be found in the work of [Simsek and Erdogan, 2019]. In their study, the determinant maximization criterion is employed to separate antisparse sources, i.e, the source samples are assumed to lie inside the $\mathcal{B}_{\ell_\infty}$, by employing the online derivation of the batch BCA approach in [Erdogan, 2013].

In this thesis, we propose two distinct biologically-plausible neural network frameworks for blind separation of potentially correlated sources. Chapter 4 introduces

determinant maximization neural networks that utilize a weighted similarity matching objective. This framework differs from the approach presented in [Simsek and Erdogan, 2019] in terms of network structure and learning dynamics. For instance, in [Simsek and Erdogan, 2019], the feedforward weights represent a linear mapping from inputs to outputs, whereas in our framework described in Chapter 4, the feedforward connections capture cross-correlations between inputs and outputs. In Chapter 5, we leverage an information-theoretic principle to construct biologically-plausible neural networks capable of separating correlated sources. Although the network dynamics in this new framework resemble those in [Simsek and Erdogan, 2019], our approach is primarily distinguished by its foundation in information theory. Furthermore, both frameworks developed in Chapter 4 and Chapter 5 are capable of handling an infinite number of identifiable source types, while online-BCA in [Simsek and Erdogan, 2019] is specifically designed for sources within $\mathcal{B}_{\ell_\infty}$.

2.4 Information Maximization for Unsupervised and Supervised Learning

The use of information maximization, particularly Shannon's mutual information, has been a prominent approach in various unsupervised learning tasks for several decades. One of the pioneering applications of mutual information maximization is found in Linsker's work on self-organizing networks, where the normative approach involves maximizing the mutual information between input and its latent representation [Linsker, 1988]. In the context of Gaussian assumptions, this objective simplifies to determinant maximization for the output covariance matrix. In the blind source separation (BSS) literature, the ICA-Infomax approach by [Bell and Sejnowski, 1995] is a well-known application of information maximization. This approach aims to maximize the information transferred from the input to the output while enforcing the constraint of mutual independence among the output components. However, for potentially correlated sources, the assumption of mutual independence may not hold. To address this, [Erdogan, 2022] introduced a second-order statistics-based correlative (or log-determinant) mutual information measure for BSS, which we re-

viewed in Section 2.2. This measure replaces the mutual independence assumption with source domain information, enabling the separation of both independent and dependent sources. Moreover, it provides an information-theoretic interpretation for the determinant maximization criterion used in various unsupervised structured matrix factorization frameworks, such as NMF (or simplex structured matrix factorization), SCA, BCA, and PMF [Chan et al., 2011, Lin et al., 2015, Fu et al., 2018, Fu et al., 2019].

Furthermore, the information maximization has been employed as a guiding principle in various neuroscience and machine learning frameworks for different tasks. In the context of feature extraction or factorization, methods such as independent component analysis (ICA-InfoMax) [Bell and Sejnowski, 1995] aim to maximize the information between the input and its internal representation. This approach has also found application in neuroscience research, providing normative explanations for cortical activity observed in experiments, such as the selectivity of primary visual cortex neurons to orientation and visual stimuli length [Hubel and Wiesel, 1959, Bell and Sejnowski, 1997]. The same principle has been also extended in the field of machine learning, with approaches like Deep Infomax that seek to transfer maximum information from the input of a deep network to its final layer while respecting prior distribution constraints on the output representations [Hjelm et al., 2019]. Self-supervised learning methods have also leveraged information maximization by matching representations derived from different augmentations or modalities of the same input [Becker and Hinton, 1992].

Recently, correlative mutual information maximization has emerged as an alternative to Shannon Mutual Information (SMI) due to its desirable properties [Erdogan, 2022]. Correlative mutual information maximization, also known as CorInfoMax, captures linear dependence and is based solely on second-order statistics, making it relatively easier to optimize compared to more complex nonlinear dependence measures. After it has been applied in the context of blind source separation (BSS) to retrieve potentially correlated components from their mixtures [Erdogan, 2022], maximizing the correlative mutual information between representations derived from different augmentations of the same input has been proposed as an ef-

fective approach in the field of self-supervised learning [Ozsoy et al., 2022]. These advancements highlight the relevance and versatility of information maximization principles in both unsupervised and supervised learning settings.

Recent work [Bozkurt et al., 2023a] has further extended the use of correlative mutual information for generating biologically plausible neural networks in BSS with infinitely many domain selections. In this thesis, we present the contributions of [Bozkurt et al., 2023a] in Chapter 5. Furthermore, [Bozkurt et al., 2023b] demonstrated that CorInfoMax principle emerges as a normative principle for biologically-plausible supervised learning with multi-compartmental neuron model. The contributions in [Bozkurt et al., 2023b] has been presented in Chapter 6 of this thesis.

2.5 Biologically Plausible Supervised Learning

Understanding how biological neural networks learn in a supervised manner has been a longstanding pursuit within the research community. The backpropagation algorithm, renowned for its efficacy in training large-scale artificial neural networks and its intuitive design, has played a pivotal role in driving the development of biologically plausible neural network frameworks [Whittington and Bogacz, 2019]. However, certain aspects of this algorithm, coupled with the oversimplified representation of artificial neurons, have posed challenges in creating biologically plausible network models based on this inspiration. Despite the widespread utilization of back-propagation of error algorithm in current state-of-the-art deep learning models, its biological plausibility has been questioned based on the following fundamental reasons: 1) it requires a special computational circuit to propagate the downstream errors to upstream neurons [Scellier and Bengio, 2017], 2) the feed-forward and feedback synaptic weights are shared (weight transport problem) [Grossberg, 1987a, Xiao et al., 2019], and 3) it violates the locality in the synaptic weight updates [Dayan and Abbott, 2005, Kunin et al., 2020].

Considering these limitations, alternative approaches have been proposed to develop biologically plausible learning algorithms that are more aligned with the principles observed in biological neural systems. Experimental studies grounded in em-

pirical evidence, such as those conducted by [Larkum, 2013] and [Petreanu et al., 2009], have had a significant impact on considering the role of dendritic processing in learning and credit assignment within multi-compartmental neurons [Richards and Lillicrap, 2019]. Subsequent research has explored biologically plausible models that incorporate supervised learning functionality. For instance, [Urbanczik and Senn, 2014] proposed a two-compartment neuron model, while [Sacramento et al., 2018] presented a three-compartment pyramidal neuron model. Both models incorporate non-Hebbian learning and spike-time dependent plasticity, with the latter also including SST (somatostatin) interneurons [Urban-Ciecko and Barth, 2016]. Similar frameworks have been proposed by [Guerguiev et al., 2017] and [Golkar et al., 2022], with the latter introduces a normative framework based on the multi-compartmental neuron structure, involving top-down feedback, lateral and feedforward connections, and incorporating both Hebbian and non-Hebbian learning rules. These models emerge from the optimization of a prediction error objective while adhering to a whitening constraint on co-layer neurons.

A significant avenue of research for advancing biologically plausible neural networks also involves the exploration of normative methods such as Equilibrium Propagation [Scellier and Bengio, 2017], Contrastive Similarity Matching [Qin et al., 2021], Predictive Coding [Rao and Ballard, 1999, Whittington and Bogacz, 2017], and others. These approaches can be combined within the framework of energy-based models [Millidge et al., 2023], providing an important parallel line of development for biologically realistic neural networks. A notable characteristic common to these energy-based models is that the contrastive or explicit optimization of the proposed energy function naturally gives rise to local update rules. Importantly, these updates can be mathematically proven to align with the gradient of the supervised loss function, thereby effectively following the backpropagation through time (BPTT) [Scellier and Bengio, 2017, Whittington and Bogacz, 2019, Millidge et al., 2023] while providing local synaptic update rules. However, these biologically plausible frameworks also face a central criticism observed in backpropagation, known as the weight transport problem. This issue arises from the fact that synaptic weights in the feedback path for error backpropagation are transposed versions of those employed

in the forward inference path [Whittington and Bogacz, 2019, Crick, 1989, Grossberg, 1987b]. This phenomenon can be explained by the transpose symmetry of the Hebbian update in relation to inputs and outputs. To address this issue, an extended Equilibrium Propagation framework described in [Laborieux et al., 2021] decouples the forward and backward connections, drawing inspiration from [Scellier et al., 2018] and [Kolen and Pollack, 1994]. This modification leads to only minor performance degradation. Another recent approach, proposed by [Golkar et al., 2022], tackles this challenge by incorporating two alternative forward prediction error loss function terms associated with the same network layer. Additionally, they leverage presumed whitening constraints to eliminate shared feedback coefficients.

In Chapter 6, we introduce an alternative normative framework based on the principle of correlative information maximization. In this framework, the three-compartment structure and the associated forward, top-down, and lateral synaptic connections emerge from the maximization of correlative mutual information (CMI) between adjacent layers, resulting in local learning rules. The key contribution of our framework lies in the adoption of two alternative expressions for the correlative mutual information between consecutive network layers as the central normative approach. By utilizing these two alternatives, our framework naturally incorporates both forward and backward prediction paths with asymmetric weights, facilitating information flow in both the feedforward and top-down directions. Notably, unlike the approach proposed by [Golkar et al., 2022], our method circumvents the need for layer whitening constraints and additional forward prediction terms to achieve the desired asymmetric weights

Chapter 3

IDENTIFIABLE POLYTOPE CHARACTERIZATION FOR POLYTOPIC MATRIX FACTORIZATION

3.1 Abstract

Polytopic Matrix Factorization (PMF) is a novel matrix decomposition technique that models data vectors as linear transformations of samples from a polytope. The successful recovery of the original factors in the generative PMF model relies on the "identifiability" of the chosen polytope. In this chapter, we delve into the problem of determining the identifiability of a polytope and present an efficient solution. The identifiability condition stipulates that the polytope must exhibit permutation and/or sign-only invariance. To ascertain whether a given polytope meets this criterion, we propose leveraging a graph automorphism algorithm. We show that examining the generating set of the linear automorphism group associated with the polytope, which corresponds to the automorphism group of an edge-colored complete graph, is sufficient to verify its identifiability. This approach alleviates the need to exhaustively check all elements of the permutation group, which would require factorial algorithm complexity. Through the numerical experiment on practical polytopes, we demonstrate the feasibility and effectiveness of our method in determining the identifiability of polytopes.

The contributions presented in this chapter has been published in [Bozkurt and Erdogan, 2022] as a conference paper at IEEE ICASSP 2022.

3.2 Introduction

Polytopic Matrix Factorization (PMF) is a structured matrix decomposition method introduced in [Tatli and Erdogan, 2021b]. It represents the columns of an input ma-

trix as linear transformations of latent vectors drawn from a polytope. The choice of the polytope reflects assumptions about the attributes and relationships of the latent components. While there are infinitely many possible polytopes to choose from, not all of them guarantee identifiability. For a polytope to be identifiable, its vertex set should remain unchanged when subjected to certain linear transformations, such as permutations or sign alterations of the components [Tatli and Erdogan, 2021a]. This chapter explores an efficient approach for determining polytope identifiability by analyzing the algebraic structure of polytope symmetries.

The study of combinatorial and geometric symmetries of polytopes has attracted significant attention due to its mathematical significance and its practical applications in optimization and engineering problems. The computation of linear, projective, and combinatorial symmetries of polyhedra has been extensively studied in the literature, with notable investigations presented in [Bremner et al., 2014] and [Bremner et al., 2009]. Moreover, the symmetry groups of polyhedra have significant implications in the fields of linear and integer linear programming [Bremner et al., 2014, Bödi and Herr, 3329, Margot, 2010]. The reference [Danielson and Borrelli, 2014] studies the symmetry groups of linear systems based on bounded polyhedral constraints to find the state-space and input-space automorphisms. In this chapter, in order to determine the identifiability of polytopes for Polytopic Matrix Factorization, we examine the linear automorphism groups of polytopes. Specifically, we demonstrate that it is adequate to examine solely the generating set of the linear automorphism group, whose size is smaller than the number of vertices of the polytope. This approach alleviates the need to consider all possible permutations of the vertices, which would result in factorial complexity. Moreover, the members of the generating set can be acquired by utilizing existing efficient graph automorphism algorithms.

3.3 Polytopic Matrix Factorization and Identifiability

An H-representation refers to the representation of a nonempty polyhedral set in $\mathbb{R}^n$ in the form of $\{\mathbf{s} \in \mathbb{R}^n | \mathbf{A}_{\mathcal{P}}\mathbf{s} \preceq \mathbf{b}\}$. If the polyhedron is bounded in the Euclidean

Space, then it is called polytope. By Minkowski-Weyl theorem [Ziegler, 1995], there exist two equivalent representations of a given polytope $\mathcal{P}$, so-called H and V representations, i.e.,

$$\mathcal{P} = \{\mathbf{s} \in \mathbb{R}^n | \mathbf{A}_{\mathcal{P}}\mathbf{s} \preceq \mathbf{b}\} \iff \mathcal{P} = conv\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$$

where the set $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ denotes the vertices of $\mathcal{P}$, and $conv\{.\}$ is the convex hull of a given set. The matrix $\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_m \end{bmatrix} \in \mathbb{R}^{n \times m}$ will be referred to as the vertex matrix of the polytope.

Throughout the article, we make the assumption that the vertex matrix $\mathbf{V}$ of a given polytope $\mathcal{P}$ has full column rank, denoted as $rank(\mathbf{V}) = n$, where $n \leq m$. Consequently, the set $\{\mathbf{v}_1, \ \ldots \ , \mathbf{v}_m\}$ can be reduced to a basis, and spans $\mathbb{R}^n$. In PMF, the input data matrix $\mathbf{X}$ is assumed to have the form $\mathbf{X} = \mathbf{A}_{\text{mix}}\mathbf{S}$ as in (2.3), where $\mathbf{A}_{\text{mix}}$ is a mixing matrix with full column rank, and the columns of $\mathbf{S}$ are the original sources that are contained within the polytope $\mathcal{P}$. The identifiability of the original sources from $\mathbf{X}$, considering potential scaling and permutation ambiguities, is contingent upon the identifiability of $\mathcal{P}$ [Tatli and Erdogan, 2021a]. The following definition describes the identifiability of $\mathcal{P}$:

Definition 1. *Let $\mathcal{P} = conv\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ be a given polytope in $\mathbb{R}^n$. Also, let $\mathbf{V} \in \mathbb{R}^{n \times m}$ be the corresponding vertex matrix. A polytope is identifiable [Tatli and Erdogan, 2021a] if and only if for any permutation matrix $\mathbf{\Pi} \in \mathbb{R}^{m \times m}$, the solution to the equation*

$$\mathbf{GV} = \mathbf{V\Pi} \tag{3.1}$$

holds for only $\mathbf{G} = \mathbf{D}\bar{\mathbf{\Pi}}$, where $\bar{\mathbf{\Pi}} \in \mathbb{R}^{n \times n}$ is a permutation matrix and $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with entries $\{+1, -1\}$.

The matrices in the form of $\mathbf{G} = \mathbf{D}\bar{\mathbf{\Pi}}$ are pointed out as the signed permutation matrices when $\mathbf{D}$ is constrained to be a diagonal matrix with entries ± 1. It can be observed that any solution matrix $\mathbf{G}$ to equation (3.1) only permutes the vertices of the polytope. Therefore, it is provable that the linear transformation of the convex polytope $\mathcal{P}$ using $\mathbf{G}$ results in the polytope itself, i.e., $\mathbf{G}(\mathcal{P}) = \mathcal{P}$.

Now, we present some fundamental observations regarding the matrices that satisfy equation (3.1).

Lemma 1. *If* $\mathbf{G}$ *satisfying (3.1) exists for a given* $\mathbf{\Pi} \in \mathbb{R}^{m\times m}$*, then it is unique.*

Proof. Let $\mathbf{G}$ and $\mathbf{G}'$ be two solutions of (3.1). Then we can write,

$$\begin{aligned} \mathbf{GV} &= \mathbf{V\Pi} = \mathbf{G'V} \\ \mathbf{GV} - \mathbf{G'V} &= \mathbf{0_{n\times m}} \\ (\mathbf{G} - \mathbf{G'})\mathbf{V} &= \mathbf{0_{n\times m}}. \end{aligned}$$

The full column rank property for $\mathbf{V}$ implies $\mathbf{G} - \mathbf{G}' = \mathbf{0_{n\times n}}$. Therefore, $\mathbf{G}$ matrix that satisfy (3.1) is unique for a fixed $\mathbf{\Pi}$. □

Lemma 2. $det(\mathbf{G}) = \pm 1$ *for any solution matrix* $\mathbf{G}$ *of (3.1).*

Proof. Let $\mathbf{G} \in \mathbb{R}^{n\times n}$, $\mathbf{\Pi} \in \mathbb{R}^{m\times m}$ such that $\mathbf{GV} = \mathbf{V\Pi}$. Multiply both sides of the equation by $\mathbf{G}$ from left, and observe that

$$\mathbf{G^2V} = \mathbf{GV\Pi} = \mathbf{V\Pi^2}.$$

By induction, one can show that $\mathbf{G}^k\mathbf{V} = \mathbf{V\Pi}^k$. For any permutation matrix $\mathbf{\Pi} \in \mathbb{R}^{m\times m}$, there exist an integer $k \leq m$ [Fraleigh and Katz, 2020], such that $\mathbf{\Pi}^k = \mathbf{I}$, where $\mathbf{I} \in \mathbb{R}^{m\times m}$ is the identity matrix. Then,

$$\mathbf{G}^k\mathbf{V} = \mathbf{V\Pi}^k = \mathbf{V}.$$

Due to the full column rank property of $\mathbf{V}$, $\mathbf{G}^k = \mathbf{I}$.

$$det(\mathbf{G}^k) = 1 \iff det(\mathbf{G})^k = 1 \iff det(\mathbf{G}) = \pm 1.$$

□

We first provide the definition of a group below for clarity of presentation [Fraleigh and Katz, 2020] before presenting results about the group structure of the set of all matrices satisfying equation (3.1):

Definition 2. *A group is a set* $\mathscr{G}$ *equipped with a binary operation* $*$ *from* $\mathscr{G}\times\mathscr{G} \longrightarrow \mathscr{G}$ *satisfying the following four axioms:*

(A1) $\forall x, y, z \in \mathscr{G} \quad (x * y) * z = x * (y * z)$ *(associative),*

(A2) $\exists e \in \mathscr{G}$ *s.t.* $\forall x \in \mathscr{G},\ e * x = x * e = x$, *(identity exist),*

(A3) $\forall x \in \mathscr{G},\ \exists x' \in \mathscr{G}$ *s.t.* $x * x' = e = x' * x$, *(inverse exists),*

(A4) $\forall x, y \in \mathscr{G},\ x * y \in \mathscr{G}$ *(closure).*

Lemma 3. *The set of all* $\mathbf{G}$ *matrices satisfying (3.1) forms a group structure together with its binary operation as matrix multiplication.*

Proof. Let $\mathscr{G}(\mathcal{P})$ denote the set of all such matrices for a given polytope $\mathcal{P}$. We illustrate that $\mathscr{G}(\mathcal{P})$ satisfies all the group axioms in Definition 2:

(A1) The set fulfills the associativity axiom as matrix multiplication is associative.

(A2) The identity element of this set is the identity matrix.

(A4) Let $\mathbf{G}_1, \mathbf{G}_2 \in \mathscr{G}(\mathcal{P})$, such that

$$\mathbf{G}_1\mathbf{V} = \mathbf{V}\mathbf{\Pi}_1, \ \textit{and}\ \mathbf{G}_2\mathbf{V} = \mathbf{V}\mathbf{\Pi}_2.$$

Therefore, we can write

$$\mathbf{G}_1\mathbf{G}_2\mathbf{V} = \mathbf{G}_1\mathbf{V}\mathbf{\Pi}_2 = \mathbf{V}\mathbf{\Pi}_1\mathbf{\Pi}_2.$$

Since $\mathbf{\Pi_1\Pi_2}$ is another permutation matrix, $\mathbf{G}_1\mathbf{G}_2 \in \mathscr{G}(\mathcal{P})$. $\mathbf{G}_2\mathbf{G}_1 \in \mathscr{G}(\mathcal{P})$ similarly follows.

(A3) $\forall \mathbf{G} \in \mathscr{G}(\mathcal{P})$, $\mathbf{G}^{-1} \in \mathscr{G}(\mathcal{P})$ follows from (A4) and the proof of Lemma 2.

□

We refer to the set $\mathscr{G}(\mathcal{P})$ as the linear automorphism group of the polytope $\mathcal{P}$ since each element in $\mathscr{G}(\mathcal{P})$ represents a linear map that preserves $\mathcal{P}$. Furthermore, according to the relation given in equation (3.1), $\mathscr{G}(\mathcal{P})$ is isomorphic to a subgroup of the permutation group S_m, which consists of $m \times m$ permutation matrices.

A generating set $Gen(\mathscr{G})$ of a group $\mathscr{G}$ is defined as a subset of the group where every element of the group can be expressed as a product of elements from $Gen(\mathscr{G})$ and their inverses. The following theorem demonstrates that characterizing the identifiability of polytope $\mathcal{P}$ can be simplified by determining if all elements of the generating set of group $\mathscr{G}(\mathcal{P})$ are sign/permutation matrices.

Theorem 1. *Let $Gen(\mathscr{G}(\mathcal{P})) = \{\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_r\}$ be the generating set of linear automorphisms for a given polytope $\mathcal{P}$. If all the elements of $Gen(\mathscr{G}(\mathcal{P}))$ are signed permutation matrices, then $\mathcal{P}$ is identifiable.*

Proof. We present the sketch of the proof, but the idea simply follows by induction. Let $\{\mathbf{\Pi}_1, \mathbf{\Pi}_2, \ldots, \mathbf{\Pi}_r\}$ represent the corresponding permutation matrices for $Gen(\mathscr{G}(\mathcal{P}))$, i.e.,
$\mathbf{G}_i\mathbf{V} = \mathbf{V}\mathbf{\Pi}_i$ for all $i = 1, \ldots, r$. Now consider the pair $\mathbf{G}_1$ and $\mathbf{G}_2$, and assume that both are signed permutation matrices, i.e., $\exists\ \mathbf{D}_1, \bar{\mathbf{\Pi}}_1, \mathbf{D}_2, \bar{\mathbf{\Pi}}_2 \in \mathbb{R}^{n\times n}$ such that

$$\mathbf{G}_1 = \mathbf{D}_1\bar{\mathbf{\Pi}}_1, \quad \mathbf{G}_2 = \mathbf{D}_2\bar{\mathbf{\Pi}}_2.$$

Based on (3.1), we obtain

$$\mathbf{G}_1\mathbf{G}_2\mathbf{V} = \mathbf{G}_1\mathbf{V}\mathbf{\Pi}_2 = \mathbf{V}\mathbf{\Pi}_1\mathbf{\Pi}_2.$$

Therefore, $\mathbf{\Pi}_1\mathbf{\Pi}_2$ is the permutation matrix corresponding to $\mathbf{G}_1\mathbf{G}_2$. Moreover, writing the product of the pair $\mathbf{G}_1$,$\mathbf{G}_2$ explicitly,

$$\begin{aligned}\mathbf{G}_1\mathbf{G}_2 &= \mathbf{D}_1\bar{\mathbf{\Pi}}_1\mathbf{D}_2\bar{\mathbf{\Pi}}_2 \\ &= \mathbf{D}_1\bar{\mathbf{\Pi}}_1\mathbf{D}_2\bar{\mathbf{\Pi}}_1^T\bar{\mathbf{\Pi}}_1\bar{\mathbf{\Pi}}_2 \\ &= \mathbf{D}_1\mathbf{D}_2'\bar{\mathbf{\Pi}}_1\bar{\mathbf{\Pi}}_2,\end{aligned}$$

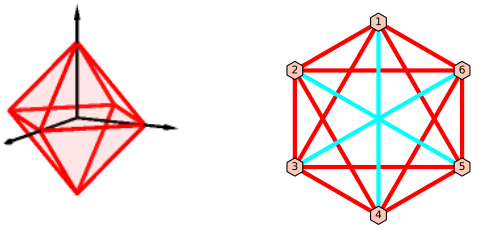
where $\mathbf{D}_2' = \bar{\mathbf{\Pi}}_1\mathbf{D}_2\bar{\mathbf{\Pi}}_1^T$ is another diagonal matrix. Therefore, $\mathbf{G}_1\mathbf{G}_2$ is a signed permutation matrix since $\mathbf{D}_1\mathbf{D}_2'$ is a diagonal and $\bar{\mathbf{\Pi}}_1\bar{\mathbf{\Pi}}_2$ is a permutation matrix. The rest of the proof follows from induction since each element of $\mathscr{G}(\mathcal{P})$ can be represented as the product of the generating set members. □

3.4 Identifiable Polytope Characterization

Determining the automorphism group of a convex polytope and identifying its generating set is known to be a challenging problem [Bremner et al., 2014, Bremner et al., 2009]. As a solution approach, we can construct an edge-colored complete graph $G_{\mathcal{P}}$ with m nodes, where each node corresponds to a vertex of the given polytope $\mathcal{P}$ with m vertices. This construction allows us to establish an isomorphism between the automorphism group of $\mathcal{P}$ and the automorphism group of $G_{\mathcal{P}}$ [Bremner et al., 2014, Bremner et al., 2009]. Moreover, there exist various coloring techniques of the edge-graph of a polytope such that the combinatorial automorphism groups of the corresponding colored edge-graphs are isomorphic to the orthogonal or linear symmetries of the polytope [Winter, 3483]. We will adopt the edge-colored complete graph construction method based on a polytope, as outlined in [Bremner et al., 2014, Bremner et al., 2009], to determine the generating set of the linear automorphism group of the polytope. For this purpose, let $\mathcal{P} \in \mathbb{R}^n$ be the given polytope with the vertex matrix $\mathbf{V} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \dots \ \mathbf{v}_m] \in \mathbb{R}^{n\times m}$. Also, let $\mathbf{Q} = \mathbf{V}\mathbf{V}^T \in \mathbb{R}^{n\times n}$ and $\mathbf{C} = \mathbf{V}^T\mathbf{Q}^{-1}\mathbf{V} \in \mathbb{R}^{m\times m}$. To construct the edge-colored complete graph $G_{\mathcal{P}}$ for a given polytope $\mathcal{P} \in \mathbb{R}^n$ with m vertices, we create a complete graph with m nodes, where each pair of distinct nodes is connected. The color of an edge connecting the i-th and j-th nodes is determined by the inner product $\mathbf{v}_i^T\mathbf{Q}^{-1}\mathbf{v}_j = \mathbf{C}_{i,j}$. The matrix $\mathbf{C}$ is called the colored (or weighted) adjacency matrix of the graph $G_{\mathcal{P}}$. For instance, in Figure 3.1a, we can observe the three-dimensional ℓ_1-norm ball ($\mathcal{B}_1$) and its corresponding edge-colored complete graph. To illustrate, we construct a complete graph with 6 nodes as depicted in Figure 3.1b. In this graph, every pair of distinct vertices is connected by an edge. For the $\mathcal{B}_1 \in \mathbb{R}^3$ illustrated in Figure 3.1a, the vertex matrix can be expressed as $\mathbf{V} = \begin{bmatrix} \mathbf{I} & -\mathbf{I} \end{bmatrix}$, and the corresponding coloring matrix $\mathbf{C}$ can be found as

$$\mathbf{C}_{i,j} = \begin{cases} 0.5 & i = j, \\ -0.5 & |i-j| = 3, \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

Thus, in Figure 3.1b, the edge connecting node-1 to node-4 is denoted as blue,



(a) $\mathcal{B}_1$.

(b) Edge-colored complete graph corresponding to $\mathcal{B}_1$.

Figure 3.1: $\mathcal{B}_1$ and Corresponding $G_{\mathcal{P}}$

while the other vertices connected to node-1 are depicted in red. This is due to the fact that $\mathbf{C}_{1,4} = -0.5$, and $\mathbf{C}_{1,j} = 0$ for $j = 2, 3, 5, 6$. The color assignments for the remaining vertices follow a similar pattern. For the graph $G_{\mathcal{P}}$, the permutation matrices which satisfy $\mathbf{C} = \mathbf{\Pi}^T \mathbf{C} \mathbf{\Pi}$ correspond to the automorphisms of the graph. These automorphisms are bijective mappings from the nodes of the graph to themselves while preserving the colored adjacency.

As we will demonstrate algebraically in Theorem 2, the linear automorphism group of a given polytope $\mathcal{P}$ is isomorphic to the automorphism group of the constructed graph $G_{\mathcal{P}}$. The graph automorphism problem, which involves determining graph isomorphisms from a graph to itself, is classified as an NP problem. However, it is currently unknown whether it is NP-complete or not [McKay and Piperno, 2014, Junttila and Kaski, 2007]. Indeed, there exist widely-used graph isomorphism algorithms that are applied in various practical applications. Although these algorithms may have exponential worst-case runtime, they are still valuable for solving graph isomorphism problems efficiently in many practical scenarios [Bremner et al., 2014, Bremner et al., 2009, McKay and Piperno, 2014, Junttila and Kaski, 2007]. For the problem of characterizing identifiable polytopes, we have adapted the graph automorphism algorithm described in [Junttila and Kaski, 2007]. The numerical example presented in Section 3.5 shows the improved efficiency of our adapted graph automorphism algorithm compared to the brute-force approach that involves search-

ing through all possible permutations of vertices, which has a factorial runtime. Furthermore, as the graph automorphism algorithm provides us with the generating set of the automorphism group, which consists of at most $(m-1)$ elements for m vertices [Junttila and Kaski, 2007, Danielson and Borrelli, 2014], the complexity of the proposed algorithm for polytope identifiability characterization will essentially be determined by the complexity of the graph isomorphism problem.

In the following, we present the algebraic proof establishing the isomorphism between the linear automorphism group of a given polytope $\mathcal{P}$ and the automorphism group of the edge-colored complete graph $G_{\mathcal{P}}$. For further graph-theoretical proofs and detailed explanations, we refer to [Bremner et al., 2014, Bremner et al., 2009]. By combining this observation with Theorem 1, we will propose a method to characterize the identifiability of a polytope.

Theorem 2. *Let* $\mathbf{\Pi} \in \mathbb{R}^{m \times m}$ *be a permutation matrix. Then,*

$$\mathbf{GV} = \mathbf{V\Pi} \iff \mathbf{C} = \mathbf{\Pi}^T \mathbf{C\Pi}$$

In that case, $\mathbf{G} = \mathbf{V\Pi V}^T\mathbf{Q}^{-1} = \mathbf{V\Pi V}^{\dagger}$.

Proof. ($\Rightarrow$ "if part") Assume $\mathbf{GV} = \mathbf{V\Pi}$. We note,

$$\mathbf{GQG}^T = \mathbf{GVV}^T\mathbf{G}^T = \mathbf{V\Pi\Pi}^T\mathbf{V}^T = \mathbf{VV}^T = \mathbf{Q}.$$

Therefore, $\mathbf{G}^T\mathbf{Q}^{-1}\mathbf{G} = \mathbf{Q}^{-1}$. Then, we obtain

$$\mathbf{\Pi}^T\mathbf{C\Pi} = \mathbf{\Pi}^T\mathbf{V}^T\mathbf{Q}^{-1}\mathbf{V\Pi} = \mathbf{V}^T\mathbf{G}^T\mathbf{Q}^{-1}\mathbf{GV}$$

$$= \mathbf{V}^T\mathbf{Q}^{-1}\mathbf{V} = \mathbf{C}.$$

($\Leftarrow$ "only if part") Assume $\mathbf{C} = \mathbf{\Pi}^T\mathbf{C\Pi}$ for a permutation matrix. Let $\mathbf{G} = \mathbf{V\Pi V}^T\mathbf{Q}^{-1}$. Then,

$$\mathbf{GV} = \mathbf{V\Pi V}^T\mathbf{Q}^{-1}\mathbf{V} = \mathbf{V\Pi C} = \mathbf{VC\Pi}$$

$$= \mathbf{VV}^T\mathbf{Q}^{-1}\mathbf{V\Pi} = \mathbf{V\Pi}.$$

□

Algorithm 1 presents a summarized approach based on Theorem 1 and Theorem 2 for characterizing the identifiability of a polytope.

Algorithm 1 Identifiability Decision

Input: $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2\ \ldots\ \mathbf{v}_m] \in \mathbb{R}^{n \times m}$ (vertex matrix)

Output: *Identifiable* (Boolean)

$\mathbf{Q} = \mathbf{V}\mathbf{V}^T$

$\mathbf{C} = \mathbf{V}^T\mathbf{Q}^{-1}\mathbf{V}$

Construct an edge-colored complete graph G_P with the edge colors from $\mathbf{C}$.

Calculate $Gen(Aut(G_{\mathcal{P}}))$, the generator set of the automorphism group of G_P with a graph isomorphism algorithm.

Identifiable = *True*

for $\mathbf{\Pi}$ in $Gen(Aut(G_{\mathcal{P}}))$ **do**

  $\mathbf{G} = \mathbf{V}\mathbf{\Pi}\mathbf{V}^\dagger$ ($\mathbf{V}^\dagger$ is pseudoinverse)

  if ($\mathbf{G}$ is not signed permutation) **then**

    Identifiable = *False* ($\mathcal{P}$ is not identifiable)

    break the loop

  else

    continue

  end if

end for

3.5 Numerical Example

To implement the proposed Algorithm 1, we utilized the *SageMath* software [The Sage Developers, 2021] in *Python 3.7*. Specifically, we employed the graph automorphism functionality provided by *SageMath*, which implements the algorithm described in [Junttila and Kaski, 2007].

To conduct experiments in the context of Polytopic Matrix Factorization, we generated random polytopes that closely resemble the example provided in [Tatli and Erdogan, 2021a]. These random polytopes were used to assess and evaluate the proposed identifiability characterization method. For any dimension $n \geq 3$, we first randomly decide for each component to be non-negative, i.e., $s_j \in [0, 1]$, or signed, i.e., $s_j \in [-1, 1]$. Then, we select q random subvectors to apply sparsity constraints in the form $\left\| \begin{bmatrix} s_{j_1^{(i)}} & s_{j_2^{(i)}} & \dots & s_{j_{l_i}^{(i)}} \end{bmatrix} \right\|_1 \leq 1$, where $q \in \{2, \dots, n\}$ is randomly selected, $i \in \{1, \dots, q\}$, $l_i \in \{2, \dots, n\}$ is the random length of the subvector-i and $\{j_1^{(i)}, \dots, j_{l_i}^{(i)}\}$ are the randomly selected indices of the subvector-i. We selected the number of relative sparsity constraint randomly. With this way, we obtain the H-representation of the polytope. Subsequently, we calculated the vertices of the generated polytopes. The resulting dataset comprises 6000 randomly generated polytopes, with dimensions ranging up to 10.

Figure 3.2 presents a comparison of the execution time between Algorithm 1 and the brute-force approach. The execution time is plotted against the number of vertices of the polytope. As anticipated, the execution time of Algorithm 1 increases exponentially with the number of vertices of the polytope. Notably, in our experiments, we found that 96% of the randomly generated polytopes were classified as identifiable using this algorithm. Figure 3.2 include the brute-force algorithm execution time only for the polytopes up to 11 vertices. It is worth mentioning that for polytopes with more than 12 vertices, the identifiability decision using the brute-force approach becomes impractical. This is due to the significantly higher complexity of the brute-force algorithm, which scales much faster compared to Algorithm 1. Therefore, Algorithm 1 provides a more efficient and feasible solution for determining identifiability in such cases.

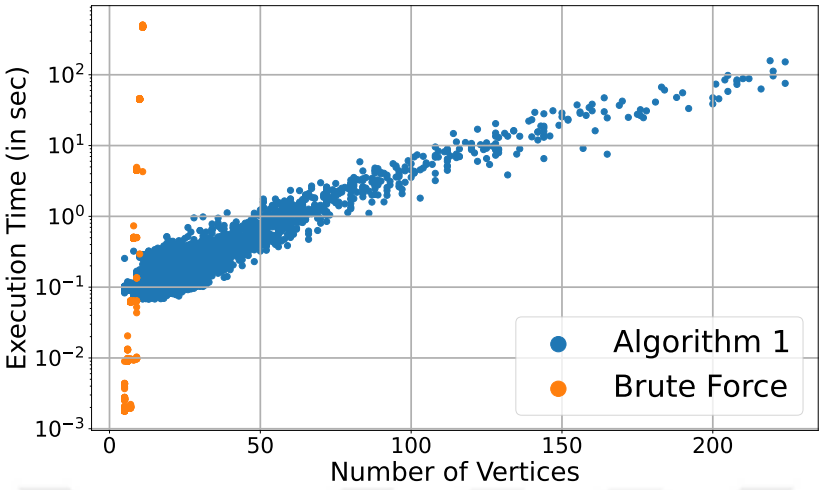


Figure 3.2: Execution time of the proposed Algorithm 1 and brute-force search. As expected, Algorithm 1 scales exponentially with respect to the number of vertices.

3.6 Conclusion

In this chapter, we have introduced a computationally tractable algorithm for making polytope identifiability decisions by leveraging graph automorphism. We have demonstrated the efficiency of this approach through our numerical experiment. While the worst-case asymptotic complexity of the graph isomorphism algorithm might appear discouraging, our experimental result confirms that the proposed approach offers a practical and feasible solution to the problem of determining polytope identifiability.

Chapter 4

BIOLOGICALLY PLAUSIBLE DETERMINANT MAXIMIZATION NEURAL NETWORKS FOR BLIND SEPARATION OF CORRELATED SOURCES

4.1 Abstract

We propose a novel biologically-plausible neural network for the blind separation of potentially dependent/correlated sources. We assume that the source vectors are sufficiently scattered in their domains and use the determinant maximization for the output correlation matrix to capture a similar spread for the source estimates. In addition, we employ the weighted similarity matching criterion, which aims to equalize the generalized inner products among output vectors to their counterparts at the input. Cascading of system blocks based on this principle can implement general linear transformations trainable by local learning rules. As an illustration of this framework, we provide two-layer neural networks corresponding to particular source domain choices. The numerical examples highlight the potential correlated source separation capability for both synthetic sources and natural images.

The contributions presented in this chapter has been published in [Bozkurt et al., 2022] as conference paper at 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

4.2 Introduction

In this chapter, we aim to overcome the limitations of existing literature, which often rely on independence assumptions, by introducing biologically-plausible BSS (Blind Source Separation) neural networks capable of separating potentially correlated sources. Instead of relying on statistical assumptions such as independence

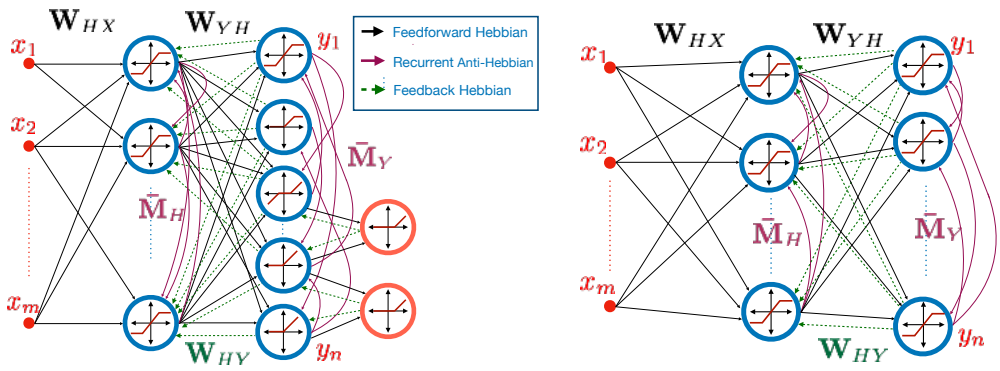
or uncorrelatedness, we focus on general geometric identifiability conditions of the sources. Specifically, 1) we make natural assumptions on the domains of source vectors, such as nonnegativity, sparsity, anti-sparsity, or boundedness (Figure 2.1) and 2) we additionally assume that the latent source vectors are sufficiently spread in their domain [Lin et al., 2015, Tatli and Erdogan, 2021b]. Because the identifiability conditions we consider are not stochastic in nature, our neural networks have the capability to separate both independent and dependent sources.

We derive our biologically-plausible algorithms based on a normative principle. One common approach to leverage our geometric identifiability conditions is by dispersing latent vector estimates across their presumed domain. This is achieved by maximizing the determinant of their sample correlation matrix, known as the Det-Max approach [Schachtner et al., 2011, Lin et al., 2015, Erdogan, 2013, Inan and Erdogan, 2014, Babatas and Erdogan, 2018]. Starting with a Det-Max objective function and considering constraints that define the domain of source vectors, we utilize mathematical tools that enable us to map optimization algorithms to adaptive Hebbian neural networks [Pehlevan and Chklovskii, 2015, Pehlevan and Chklovskii, 2019, Erdogan and Pehlevan, 2020], and we derive two-layered neural networks capable of separating potentially correlated sources from their linear mixtures (Figure 4.1). The neural networks we propose consist of feedforward, recurrent, and feedback synaptic connections that are updated using Hebbian or anti-Hebbian update rules. The structure of the output layer of the neural network is determined by the domain of the latent sources (Figure 4.1, Table B.1).

In summary, we can list our main contributions in this chapter as follows:

- We introduce a normative framework for generating biologically plausible neural networks capable of separating correlated sources from their mixtures. These networks are derived from a Det-Max objective function with source domain constraints.

- Our framework is versatile and can handle a wide range of source types by leveraging the topology of their source domains.



(a) General source domain with sparse components

(b) Antisparse sources

Figure 4.1: Det-Max WSM neural networks for blind source separation. The neural network accepts a mixed input $\mathbf{x}$ and generates latent components $\mathbf{y}$ as its output. The configuration of the output layer is influenced by the selected source domain. Inhibitory neurons establish connections between mutually sparse components within the output layer.

- Through simulations with both synthetic and realistic data, we demonstrate the efficacy of our neural networks and their ability to successfully separate correlated sources.

4.3 Problem Statement: Blind Source Separation

SOURCES: In the blind source separation (BSS) setting, we consider a finite set of n-dimensional source vectors denoted as the set $\mathbb{S} = \{\mathbf{s}(1), \mathbf{s}(2), \ldots, \mathbf{s}(t)\} \subset \mathcal{P}$. Here, $\mathcal{P}$ denotes a specific subset of $\mathbb{R}^n$. The choice of the source domain $\mathcal{P}$ is critical in determining the identifiability of the sources from their mixtures, as well as the properties and relationships among the individual sources [Tatli and Erdogan, 2021b, Erdogan, 2022]. Structured unsupervised matrix factorization frameworks are commonly defined based on the source or latent domain, taking into consideration the specific characteristics or constraints of the sources, such as

- *Bounded sparse sources*: A natural choice of domain for sparse sources is the unit ℓ_1 norm ball, denoted as $\mathcal{B}_{\ell_1} = \{\mathbf{s} \mid \|\mathbf{s}\|_1 \leq 1\}$ (Figure 2.1a). The

use of ℓ_1-norm as a convex (non)sparsity measure has been widely successful in numerous applications including sparse dictionary learning and component analysis [Donoho, 2006, Elad, 2010, Kreutz-Delgado et al., 2003, Li et al., 2004, Babatas and Erdogan, 2018] and modeling of V1 receptive fields [Olshausen and Field, 1996].

- *Bounded anti-sparse sources*: A common domain choice for anti-sparse sources is the unit ℓ_∞-norm-ball: $\mathcal{B}_{\ell_\infty} = \{\mathbf{s} \mid \|\mathbf{s}\|_\infty \leq 1\}$ (Figure 2.1b). If vectors drawn from $\mathcal{B}_{\ell_\infty}$ are well-spread inside this set, it is possible for some samples to contain near-peak magnitude values simultaneously at all their components. This equal spreading of values among the components justifies the term "anti-sparse" [Elvira et al., 2016] or "democratic" [Studer et al., 2014] component representations. This choice is well-suited for various applications, including those in natural images and digital communication constellations [Cruces, 2010, Erdogan, 2013].

- *Normalized nonnegative sources*: Simplex structured matrix factorization [Chan et al., 2011, Lin et al., 2015, Lin et al., 2018] utilizes the unit simplex [Donoho and Stodden, 2003, Fu et al., 2019] $\Delta = \{\mathbf{s} \mid \mathbf{s} \geq 0, \mathbf{1}^T\mathbf{s} = 1\}$ (Figure 2.1c) as the source domain. Nonnegativity of sources is a natural assumption in various biological contexts, such as demixing olfactory mixtures [Grabska-Barwińska et al., 2017]. This domain set, corresponding to ℓ_1-norm normalized versions of nonnegative orthant vectors, is also commonly used in NMF algorithms [Fu et al., 2019].

- *Nonnegative bounded anti-sparse sources*: A non-degenerate polytopic choice of the nonnegative sources can be obtained by combining nonnegativity and anti-sparseness constraints. This corresponds to the intersection of $\mathcal{B}_{\ell_\infty}$ and the nonnegative orthant $\mathbb{R}^n_+$, represented as $\mathcal{B}_{\ell_\infty,+} = \mathcal{B}_{\ell_\infty} \cap \mathbb{R}^n_+$ [Tatli and Erdogan, 2021b] (Figure 2.1d).

- *Nonnegative bounded sparse sources*: Another polytopic choice for nonnegative sources can be obtained by intersecting the unit ℓ_1-norm ball $\mathcal{B}_{\ell_1}$ with the nonnegative orthant $\mathbb{R}_+$, [Tatli and Erdogan, 2021b]: $\mathcal{B}_{\ell_1,+} = \mathcal{B}_{\ell_1} \cap \mathbb{R}_+^n$ (Figure 2.1e).

Except for the unit simplex Δ, the examples mentioned above belong to an infinite set of identifiable polytopes that satisfy a symmetry restriction that we formalized in Definition 1. The identifiability property of these polytopes are proven in [Tatli and Erdogan, 2021b].

Further, instead of uniformly imposing features such as sparsity and nonnegativity on all elements of the component vector, we can assign these attributes at the subvector level and still achieve identifiable polytopes. For example, in [Tatli and Erdogan, 2021b], the authors provides the set

$$\mathcal{P}_{ex} = \left\{ \mathbf{s} \in \mathbb{R}^3 \;\middle|\; s_1, s_2 \in [-1,1],\ s_3 \in [0,1],\ \left\| \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \right\|_1 \leq 1,\ \left\| \begin{bmatrix} s_2 \\ s_3 \end{bmatrix} \right\|_1 \leq 1 \right\},$$

as a simple demonstration of such polytopes with heterogeneous structure where s_3 is nonnegative, s_1, s_2 are signed, and $\begin{bmatrix} s_1 & s_2 \end{bmatrix}^T$, $\begin{bmatrix} s_2 & s_3 \end{bmatrix}^T$ are sparse subvectors, while sparsity is not imposed on the whole vector $\begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix}^T$. In this chapter, we focus on specific source domains, including the unit simplex and a subset of identifiable polytopes. In these polytopes, attributes such as sparsity and nonnegativity are defined at the subvector level in a general form that can be stated as

$$\mathcal{P} = \left\{ \mathbf{s} \in \mathbb{R}^n \;\middle|\; s_i \in [-1,1]\, \forall i \in \mathcal{I}_s,\ s_i \in [0,1]\, \forall i \in \mathcal{I}_+,\ \|\boldsymbol{s}_{\mathcal{J}_k}\|_1 \leq 1,\ \mathcal{J}_k \subseteq \mathbb{Z}_n,\ k \in \mathbb{Z}_L \right\} \quad (4.1)$$

where $\mathcal{I}_+ \subseteq \mathcal{Z}_n$ is the index set for nonnegative sources, and I_s is its complement, $\boldsymbol{s}_{\mathcal{J}_k}$ is the subvector constructed from the elements with indices in $\mathcal{J}_k$, and L is the number of sparsity constraints imposed in the subvector level. Those polytopes that can be represented in the form of (4.1) will be referred as *feature-based* polytopes.

The Det-Max criterion for BSS relies on the assumption that the source samples are well-spread within their respective domains. The papers [Fu et al., 2018] and [Tatli and Erdogan, 2021b] present specific conditions on the scattering of source

samples that ensure their identifiability for the unit simplex and polytopes, respectively. We briefly reviewed these conditions in Section 2.1.

It is important to note that our assumptions about the sources are deterministic. Therefore, we do not rely on any stochastic assumptions such as independence or uncorrelatedness of the sources. As a result, our algorithms are capable of separating both independent and dependent (potentially correlated) sources.

MIXING: The sources $\mathbf{s}(t)$ are mixed through a random matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$.

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t), \qquad t \in \mathbb{Z}. \tag{4.2}$$

In our study, we focus on the (over)determined case where the number of observations m is greater than or equal to the number of sources n, and we assume that the mixing matrix is full-rank. While we consider noiseless mixtures to achieve perfect separability in our analysis, we take into account the potential presence of noise in the optimization setting of our online algorithm. We use $\mathbf{S}(t) = \begin{bmatrix} \mathbf{s}(1) & \dots & \mathbf{s}(t) \end{bmatrix} \in \mathbb{R}^{n \times t}$ and $\mathbf{X}(t) = \begin{bmatrix} \mathbf{x}(1) & \dots & \mathbf{x}(t) \end{bmatrix} \in \mathbb{R}^{m \times t}$ to represent data snapshot matrices, at time t, for sources and mixtures, respectively.

SEPARATION: The goal of source separation is to estimate the source signals $\mathbf{S}(t)$ from the mixture measurements $\mathbf{X}(t)$ in the case where the mixing matrix $\mathbf{A}$ is unknown. In our approach, we denote the estimated source signals as $\mathbf{y}(t)$, which are obtained as linear transformations of the observed mixtures, i.e., $\boldsymbol{y}(i) = \mathbf{W}\mathbf{x}(i)$, where $\mathbf{W} \in \mathbb{R}^{n \times m}$. We define $\mathbf{Y}(t) = \begin{bmatrix} \mathbf{y}(1) & \mathbf{y}(2) & \dots & \mathbf{y}(t) \end{bmatrix} \in \mathbb{R}^{n \times t}$ as the output snapshot matrix. "Ideal separation" refers to the condition where the output signals are scaled and/or permuted versions of the original sources, meaning they satisfy $\mathbf{y}(i) = \mathbf{\Pi}\mathbf{\Lambda}\mathbf{s}(i)$, where $\mathbf{\Pi}$ is a permutation matrix, and $\mathbf{\Lambda}$ is a full rank diagonal matrix.

4.4 Weighted Similarity Matching-Based Optimization Formulation of Determinant Maximization Criterion

Here, we present a reformulation of the Det-Max problem (2.4) that enables the derivation of biologically-plausible neural networks for the linear BSS setup discussed in Section 4.3.

We propose to use the following optimization problem:

$$\underset{\substack{\mathbf{Y}(t),\mathbf{H}(t),\\ D_{1,11}(t),\dots D_{1,nn}(t),\mathbf{D}_1(t)\\ D_{2,11}(t),\dots D_{2,nn}(t),\mathbf{D}_2(t)}}{\text{minimize}} \quad \sum_{i=1}^{n}\log(D_{1,ii}(t))+\sum_{i=1}^{n}\log(D_{2,ii}(t)) \tag{4.3a}$$

$$\text{subject to} \quad \mathbf{X}(t)^T\mathbf{X}(t)-\mathbf{H}(t)^T\mathbf{D}_1(t)\mathbf{H}(t)=0, \tag{4.3b}$$

$$\mathbf{H}(t)^T\mathbf{H}(t)-\mathbf{Y}(t)^T\mathbf{D}_2(t)\mathbf{Y}(t)=0, \tag{4.3c}$$

$$\mathbf{y}(i)\in\mathcal{P}, i=1,\dots,t, \tag{4.3d}$$

$$\mathbf{D}_l(t)=\text{diag}(D_{l,11}(t),\dots,D_{l,nn}(t)), \quad l=1,2, \tag{4.3e}$$

$$D_{l,11}(t),D_{l,22}(t),\dots,D_{l,nn}(t)>0, \quad l=1,2 \tag{4.3f}$$

In this problem, $\mathbf{X}(t)\in\mathbb{R}^{m\times t}$ is the matrix containing input (mixture) vectors, $\mathbf{Y}(t)\in\mathbb{R}^{n\times t}$ is the matrix containing output vectors, $\mathbf{H}(t)\in\mathbb{R}^{n\times t}$ is a slack variable matrix containing an intermediate signal $\{\mathbf{h}(i)\in\mathbb{R}^n, i=1,\dots,t\}$, corresponding to the hidden layer of the neural network implementation discussed in Section 4.5, in its columns $\mathbf{H}(t)=\begin{bmatrix}\mathbf{h}(1) & \mathbf{h}(2) & \dots & \mathbf{h}(t)\end{bmatrix}$. In the proposed optimization problem, we introduce the following nonnegative slack variables: $D_{l,11}(t),D_{l,22}(t),\dots,D_{l,nn}(t)$ for $l=1,2$. These variables will be further described below. We also define $\mathbf{D}_l$ for $l=1,2$ as a diagonal matrix containing the weights $D_{l,ii}$ for $i=1,\dots,n$. The constraint equation 4.3d ensures that the outputs lie within the presumed domain of sources.

The optimization problem presented is related to the weighted similarity matching (WSM) objective introduced in [Erdogan and Pehlevan, 2020]. The constraints equation 4.3b and equation 4.3c correspond to two separate WSM conditions. The equality constraint in equation 4.3b represents a WSM constraint between the input $\mathbf{X}(t)$ and the intermediate signal $\mathbf{H}(t)$. This constraint enforces that the pairwise weighted correlations between the elements of the signal $\{\mathbf{h}(i), i=1,\dots,t\}$ are equal to the correlations among the elements of the input signal $\{\mathbf{x}(i), i=1,\dots,t\}$. In other words, it ensures that $\mathbf{x}(i)^T\mathbf{x}(j)=\mathbf{h}(i)^T\mathbf{D}_1(t)\mathbf{h}(j)$ for all $i,j\in\{1,\dots,t\}$. $D_{1,11}(t),D_{1,22}(t),\dots,D_{1,nn}(t)$ are the inner product weights used in these equalities. Likewise, the equality constraint in (4.3c) establishes a WSM constraint between the

intermediate signal $\mathbf{H}(t)$ and outputs $\mathbf{Y}(t)$. This equality constraint can be written as $\mathbf{h}(i)^T\mathbf{h}(j) = \mathbf{y}(i)^T\mathbf{D}_2(t)\mathbf{y}(j)$, $i, j \in \{1, \ldots, t\}$, and $D_{2,11}(t), D_{2,22}(t), \ldots, D_{2,nn}(t)$ correspond to the inner product weights used in these equalities. The optimization objective involves minimizing the logarithm of the determinant of the weighting matrices $\mathbf{D}_1(t)$ and $\mathbf{D}_2(t)$.

The following theorem establishes the relationship between our Weighted Similarity Matching (WSM)-based objective and the original Determinant Maximization (Det-Max) criterion:

Theorem 3. *If $\mathbf{X}(t)$ is full column-rank, then global optimal $\mathbf{Y}(t)$ solutions of equation (2.4) and equation (4.3) coincide.*

The proof of Theorem 3 relies on the following lemma, which is established from equality constraints in (4.3b) and (4.3c):

Lemma 4. *Given the mixing model $\mathbf{x}(i) = \mathbf{A}\mathbf{s}(i)$, for sufficiently large sample sizes enabling full column-rank condition on $\mathbf{X}(t)$, the constraints in (4.3b) and (4.3c) define an arbitrary linear mapping between input and output vectors in the form*

$$\mathbf{y}(i) = \mathbf{W}(t)\mathbf{x}(i), \qquad i \in \{1, \ldots, t\}. \tag{4.4}$$

where $\mathbf{W}(t)$ is full-rank.

Proof of Lemma 4. To understand the relationship between $\mathbf{h}(i)$ and $\mathbf{x}(i)$, we can consider the mixing relation equation 4.2 and the equality constraint equation 4.3b, which imply that: $\mathbf{S}(t)^T\mathbf{A}^T\mathbf{A}\mathbf{S}(t) = \mathbf{H}(t)^T\mathbf{D}_1(t)\mathbf{H}(t)$. We define $\mathbf{A} = \mathbf{U}_\mathbf{A}\mathbf{\Sigma}_\mathbf{A}\mathbf{V}_\mathbf{A}^T$ as the reduced SVD decomposition for $\mathbf{A}$ with $\mathbf{U}_\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{\Sigma}_\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{V}_\mathbf{A} \in \mathbb{R}^{n\times n}$. Then, we can write $\mathbf{A}^T\mathbf{A} = \mathbf{V}_\mathbf{A}\mathbf{\Sigma}_\mathbf{A}^2\mathbf{V}_\mathbf{A}^T$. For sufficiently large sample sizes that enable full rank $\mathbf{X}(t)$, these conditions imply $\mathbf{h}(i) = \mathbf{D}_1(t)^{-1/2}\mathbf{Q}_1^T(t)\mathbf{\Sigma}_\mathbf{A}\mathbf{V}_\mathbf{A}^T\mathbf{s}(i)$, $i \in \{1, \ldots, t\}$, for some real-orthogonal matrix $\mathbf{Q}_1(t)$. Using this expression, we can also express $\mathbf{h}(i) = \mathbf{D}_1(t)^{-1/2}\mathbf{Q}_1^T(t)\mathbf{U}_\mathbf{A}^T\mathbf{x}(i) = \mathbf{D}_1(t)^{-1/2}\mathbf{\Theta}_1^T(t)\mathbf{x}(i)$, $i \in \{1, \ldots, t\}$, where $\mathbf{\Theta}_1(t) = \mathbf{U}_\mathbf{A}\mathbf{Q}_1(t) \in \mathbb{R}^{m\times n}$ is a matrix with orthonormal columns.

The weighted inner product matching condition in equation 4.3c implies a relationship between the output vectors $\mathbf{y}(i)$ and the slack vectors $\mathbf{h}(i)$, given by

$\mathbf{y}(i) = \mathbf{D}_2(t)^{-1/2}\mathbf{\Theta}_2(t)\mathbf{h}(i)$, $i \in \{1, \ldots, t\}$, where $\mathbf{\Theta}_2(t)$ is another real-orthogonal matrix. As a result, $\mathbf{y}(i) = \mathbf{D}_2(t)^{-1/2}\mathbf{\Theta}_2(t)\mathbf{D}_1(t)^{-1/2}\mathbf{\Theta}_1^T(t)\mathbf{x}(i)$, $i \in \{1, \ldots, t\}$. Here, the multiplier of $\mathbf{x}(i)$ takes the form of a Singular Value Decomposition (SVD) of a full rank matrix, specifically $\mathbf{\Theta}_2(t)\mathbf{D}_1(t)^{-1/2}\mathbf{\Theta}_1^T(t)$, which is then left multiplied by a full rank diagonal matrix $\mathbf{D}_2(t)^{-1/2}$. Therefore, this implies that the equality constraints (4.3b) and (4.3c), when considered together, define an arbitrary linear mapping between input and output vectors. □

Now we can prove Theorem 3.

Proof of Theorem 3. By Lemma 4, $\mathbf{y}(i) = \mathbf{W}(t)\mathbf{x}(i)$, $i = 1, \ldots t$, where $\mathbf{W}(t)$ admits the form $\mathbf{W}(t) = \mathbf{D}_2(t)^{-1/2}\mathbf{\Theta}_2(t)\mathbf{D}_1(t)^{-1/2}\mathbf{\Theta}_1^T(t)$. Therefore, using the mixing relationship $\mathbf{x}(i) = \mathbf{A}\mathbf{s}(i)$, we can write $\mathbf{y}(i) = \mathbf{G}(t)\mathbf{s}(i)$, where $\mathbf{G}(t) = \mathbf{W}(t)\mathbf{A}$ is the linear mapping between sources and outputs. We plug this into Det-Max objective in (2.4) to obtain $\log(\det(\mathbf{Y}(t)\mathbf{Y}(t)^T)) = \log\det(\mathbf{G}(t)\mathbf{S}(t)\mathbf{S}(t)^T\mathbf{G}(t)^T) = 2\log(|\det(\mathbf{G}(t))|) + \log(\det(\mathbf{S}(t)\mathbf{S}(t)^T))$.Defining $\mathbf{A} = \mathbf{U}_\mathbf{A}\mathbf{\Sigma}_\mathbf{A}\mathbf{V}_\mathbf{A}^T$ as the reduced SVD for matrix $\mathbf{A}$, we obtain $\mathbf{G}(t) = \mathbf{D}_2(t)^{-1/2}\mathbf{\Theta}_2(t)\mathbf{D}_1(t)^{-1/2}\mathbf{\Theta}_1(t)^T\mathbf{U}_\mathbf{A}\mathbf{\Sigma}_\mathbf{A}\mathbf{V}_\mathbf{A}^T$. As a result, $2\log(|\det(\mathbf{G}(t))|) = -\log(\det(\mathbf{D}_1)) - \log(\det(\mathbf{D}_2)) + 2\log(\det(\mathbf{\Sigma}_\mathbf{A}))$. Consequently, minimizing $\log(\det(\mathbf{D}_1))+\log(\det(\mathbf{D}_2))$ with some additional constant terms is equivalent to maximizing the objective in (2.4). The equivalent function can be written as $\sum_{i=1}^n \log(D_{1,ii}(t)) + \sum_{i=1}^n \log(D_{2,ii}(t))$ as the objective function in (4.3a) due to the fact that $\mathbf{D}_1$ and $\mathbf{D}_2$ are diagonal. □

4.5 Biologically-plausible Neural Networks for WSM-based BSS

In this section, we shift our focus from the offline setting to the online setting, which better aligns with the dynamics of biological systems. In the online scenario, inputs are observed and outputs are produced sequentially, one datum at a time, before the next input is observed. Consequently, we introduce an online version of the batch WSM problem equation 4.3 for which we show that the corresponding gradient descent algorithm leads to a biologically-plausible two-layer neural network with local update rules.

4.5.1 Online Optimization Setting for WSM-based BSS

We propose an online extension of WSM-based BSS (4.3), taking into account the constraints imposed by the sequential nature of the problem. In the online setting, past outputs cannot be altered, but the information contained in past inputs and outputs is still valuable for solving the BSS problem. Therefore, we formulate an optimization problem that aims to estimate the sources $\mathbf{y}(t)$ based on the current mixture $\mathbf{x}(t)$, while leveraging information from all previous fixed inputs and outputs.

We will now introduce our notation. In order to account for potential nonstationarity in the data, we adopt exponential weighting of the signals. We define the weighted input data snapshot matrix at time t as follows,

$$\boldsymbol{\mathcal{X}}(t) = \begin{bmatrix} \gamma^{t-1}\mathbf{x}(1) & \dots & \gamma\mathbf{x}(t-1) & \mathbf{x}(t) \end{bmatrix} = \mathbf{X}(t)\boldsymbol{\Gamma}(t),$$

where γ is the forgetting factor and $\boldsymbol{\Gamma}(t) = \text{diag}(\gamma^{t-1}, \dots, \gamma, 1)$. The use of exponential weighting allows us to place more emphasis on recent mixtures while reducing the impact of past samples. Similarly, we define the corresponding weighted output snapshot matrix for output as

$$\boldsymbol{\mathcal{Y}}(t) = \begin{bmatrix} \gamma^{t-1}\mathbf{y}(1) & \dots & \gamma\mathbf{y}(t-1) & \mathbf{y}(t) \end{bmatrix} = \mathbf{Y}(t)\boldsymbol{\Gamma}(t),$$

and the hidden layer vectors as

$$\boldsymbol{\mathcal{H}}(t) = \begin{bmatrix} \gamma^{t-1}\mathbf{h}(1) & \dots & \gamma\mathbf{h}(t-1) & \mathbf{h}(t) \end{bmatrix} = \mathbf{H}(t)\boldsymbol{\Gamma}(t).$$

Furthermore, we define τ as the effective time window length for sample correlation calculations based on the exponential weights, given by $\tau = \lim_{t\to\infty} \sum_{k=0}^{t-1} \gamma^{2k} = \frac{1}{1-\gamma^2}$.

We convert equality constraints in (4.3b) and (4.3c) to similarity matching cost functions to derive an online cost function as follows

$$J_1(\mathbf{H}(t), \mathbf{D}_1(t)) = \frac{1}{2\tau^2}\|\boldsymbol{\mathcal{X}}(t)^T\boldsymbol{\mathcal{X}}(t) - \boldsymbol{\mathcal{H}}(t)^T\mathbf{D}_1(t)\boldsymbol{\mathcal{H}}(t)\|_F^2,$$

$$J_2(\mathbf{H}(t), \mathbf{D}_2(t), \mathbf{Y}(t)) = \frac{1}{2\tau^2}\|\boldsymbol{\mathcal{H}}(t)^T\boldsymbol{\mathcal{H}}(t) - \boldsymbol{\mathcal{Y}}(t)^T\mathbf{D}_2(t)\boldsymbol{\mathcal{Y}}(t)\|_F^2.$$

Then, we combine the similarity matching costs with the objective function in (4.3a) using a weighted combination, resulting in the final cost function

$$\begin{aligned}&\mathcal{J}(\mathbf{H}(t),\mathbf{D}_1(t),\mathbf{D}_2(t),\mathbf{Y}(t))=\\&\lambda_{SM}[\beta J_1(\mathbf{H}(t),\mathbf{D}_1(t))+(1-\beta)J_2(\mathbf{H}(t),\mathbf{D}_2(t),\mathbf{Y}(t))]\\&+(1-\lambda_{SM})[\textstyle\sum_{k=1}^{n}\log(D_{1,kk}(t))+\sum_{k=1}^{n}\log(D_{2,kk}(t))].\end{aligned}\tag{4.5}$$

Here, $\beta\in[0,1]$ and $\lambda_{SM}\in[0,1]$ are parameters that control the convex combination of similarity matching costs and the objective function. We can now formulate the online optimization problem, which aims to determine the current output $\mathbf{y}(t)$, the corresponding hidden state $\mathbf{h}(t)$, and update the gain parameters $\mathbf{D}_l(t)$ for $l=1,2$. The problem is given by:

$$\underset{\substack{\mathbf{y}(t),\mathbf{h}(t),\\ D_{1,11}(t),\dots D_{1,nn}(t),\mathbf{D}_1(t)\\ D_{2,11}(t),\dots D_{2,nn}(t),\mathbf{D}_2(t)}}{\text{minimize}} \quad \mathcal{J}(\mathbf{H}(t),\mathbf{D}_1(t),\mathbf{D}_2(t),\mathbf{Y}(t)) \tag{4.6a}$$

$$\text{subject to} \quad \mathbf{y}(t)\in\mathcal{P}, \tag{4.6b}$$

$$\mathbf{D}_l(t)=\operatorname{diag}(D_{l,11}(t),\dots,D_{l,nn}(t)),\quad l=1,2, \tag{4.6c}$$

$$D_{l,11}(t),D_{l,22}(t),\dots,D_{l,nn}(t)>0,\quad l=1,2 \tag{4.6d}$$

As we show in Appendix A.1, part of $\mathcal{J}$ that depends on the hidden state $\mathbf{h}_t$ and the output $\mathbf{y}_t$ can be expressed as follows

$$\begin{aligned}C(\mathbf{h}(t),\mathbf{y}(t)) \;=\;& 2\mathbf{h}(t)^T\mathbf{D}_1\mathbf{M}_H(t)\mathbf{D}_1(t)\mathbf{h}(t)-4\mathbf{h}(t)^T\mathbf{D}_1(t)\mathbf{W}_{HX}(t)\mathbf{x}(t)\\&+2\mathbf{y}(t)^T\mathbf{D}_2(t)\mathbf{M}_Y(t)\mathbf{D}_2(t)\mathbf{y}(t)-4\mathbf{y}(t)^T\mathbf{D}_2(t)\mathbf{W}_{YH}(t)\mathbf{h}(t)\\&+2\mathbf{h}(t)^T\mathbf{M}_H(t)\mathbf{h}(t),\end{aligned}\tag{4.7}$$

where the dependence on past inputs and outputs appear in the weighted correlation matrices:

$$\begin{aligned}\mathbf{M}_H(t)&=\tfrac{1}{\tau}\textstyle\sum_{k=1}^{t-1}(\gamma^2)^{t-1-k}\mathbf{h}(k)\mathbf{h}(k)^T,\\ \mathbf{W}_{HX}(t)&=\tfrac{1}{\tau}\textstyle\sum_{k=1}^{t-1}(\gamma^2)^{t-1-k}\mathbf{h}(k)\mathbf{x}(k)^T,\\ \mathbf{W}_{YH}(t)&=\tfrac{1}{\tau}\textstyle\sum_{k=1}^{t-1}(\gamma^2)^{t-1-k}\mathbf{y}(k)\mathbf{h}(k)^T,\\ \mathbf{M}_Y(t)&=\tfrac{1}{\tau}\textstyle\sum_{k=1}^{t-1}(\gamma^2)^{t-1-k}\mathbf{y}(k)\mathbf{y}(k)^T.\end{aligned}\tag{4.8}$$

4.5.2 Description of Network Dynamics for Bounded Anti-sparse Sources

We can interpret the gradient-descent minimization of the online WSM cost function in (4.6) as the dynamics of a neural network with local learning rules. In this section, we focus on the domain choice $\mathcal{P} = \mathcal{B}_\infty$ as an illustrative example. The specific choice of this domain determines the exact network architecture for our source separation task, and we will use it as a basis to demonstrate the neural network's dynamics and learning rules. However, it's important to note that the proposed approach is general and can be extended to other identifiable source domains, each of which can lead to a different neural network architecture. In the following sections, we will discuss how to generalize the results of this section by modifying the output layer for different identifiable source domains. To begin, let's write the update expressions for the optimization variables based on the gradients of the cost function $\mathcal{J}(\mathbf{h}(t), \mathbf{y}(t), \mathbf{D}_1(t), \mathbf{D}_2(t))$:

Update dynamics for $\mathbf{h}(t)$: Utilizing the gradient of equation 4.7 in (A.7) with respect to $\mathbf{h}(t)$, and following previous work [Rozell et al., 2008, Pehlevan, 2019], we can express update dynamics for $\mathbf{h}(t)$ as follows:

$$\begin{aligned} \frac{d\mathbf{v}(\tau)}{d\tau} &= -\mathbf{v}(\tau) - \lambda_{SM}[((1-\beta)\bar{\mathbf{M}}_H(t) + \beta\mathbf{D}_1(t)\bar{\mathbf{M}}_H(t)\mathbf{D}_1(t))\mathbf{h}(\tau) \\ &\quad +\beta\mathbf{D}_1(t)\mathbf{W}_{HX}(t)\mathbf{x}(\tau) + (1-\beta)\mathbf{W}_{YH}(t)^T\mathbf{D}_2(t)\mathbf{y}(\tau)] \qquad (4.9) \\ \mathbf{h}(t;\tau)_i &= \sigma_A\left(\frac{\mathbf{v}(\tau)_i}{\lambda_{SM}\Gamma_{Hii}(t)((1-\beta)+\beta D_{1,ii}(t)^2)}\right), \quad \text{for } i = 1,\dots n, \qquad (4.10) \end{aligned}$$

where $\mathbf{\Gamma}_H(t)$ is the diagonal matrix containing the diagonal elements of $\mathbf{M}_H(t)$, $\bar{\mathbf{M}}_H(t) = \mathbf{M}_H(t) - \mathbf{\Gamma}_H(t)$, and $\sigma(\cdot)$ is the clipping function which is defined as

$$\sigma_A(x) = \begin{cases} x & -A \le x \le A, \\ A\text{sign}(x) & \text{otherwise.} \end{cases}$$

This dynamics minimize the equation equation 4.7 [Rozell et al., 2008]. Here $\mathbf{v}(\tau)$ is an internal variable that could be interpreted as the voltage dynamics of a biological neuron. It is defined as a linear transformation of $\mathbf{h}(t)$ in (A.8). Equation (4.9) defines the dynamics of $\mathbf{v}(\tau)$ based on the gradient of equation equation 4.7 with

respect to $\mathbf{h}(t)$ as given in equation (A.7). The positive definite linear map in equation (A.8) ensures that the expression in equation (A.7) serves as a descent direction for $\mathbf{v}(\tau)$. Furthermore, the function $\sigma(\cdot)$ represents the projection onto the set $A\mathcal{B}_\infty$, where the interval $[-A, A]$ defines the presumed dynamic range for the components of $\mathbf{h}(t)$. It is important to note that in the online optimization setting described in Section 4.5.1, there is no explicit constraint set for $\mathbf{h}(t)$, allowing A to be chosen arbitrarily large in the actual implementation. The inclusion of the nonlinearity in equation (4.10) is intended to model the limited dynamic range of an actual (biological) neuron.

Update dynamics for output $\mathbf{y}(t)$: Using (A.10), the update dynamics for the output $\mathbf{y}(t)$ can be written as

$$\frac{d\mathbf{u}(\tau)}{d\tau} = -\mathbf{u}(\tau) + \mathbf{W}_{YH}(t)\mathbf{h}(\tau) - \bar{\mathbf{M}}_Y(t)\mathbf{D}_2(t)\mathbf{y}(\tau), \tag{4.11}$$

$$\mathbf{y}(t;\tau)_i = \sigma_1\left(\frac{\mathbf{u}(\tau)_i}{\Gamma_{Yii}(t)D_{2,ii}(t)}\right), \quad \text{for } i = 1, \ldots n, \tag{4.12}$$

The update dynamics for $\mathbf{y}(t)$ can be derived using a similar approach as for $\mathbf{h}(t)$. By substituting the descent direction expression from (A.10) and the substitution in (A.11), we obtain the update dynamics for $\mathbf{y}(t)$. In these dynamics, $\mathbf{\Gamma}_Y(t)$ represents a diagonal matrix containing the diagonal elements of $\mathbf{M}_Y(t)$, and $\bar{\mathbf{M}}Y(t) = \mathbf{M}_Y(t)$ minus $\mathbf{\Gamma}_Y(t)$. Also, note that the nonlinear mapping $\sigma_1(\cdot)$ corresponds to the projection onto the presumed domain of sources, in this case, $\mathcal{P} = \mathcal{B}_\infty$, which is achieved through an elementwise clipping operation.

The state space representations given in (4.9)-(4.10) and (4.11)-(4.12) can be interpreted as a two-layer recurrent neural network. In this network, the input is represented by $\mathbf{x}(t)$, the hidden layer activation is represented by $\mathbf{h}(t)$, and the output layer activation is represented by $\mathbf{y}(t)$. The synaptic weight matrices $\mathbf{W}_{HX}$ and $\mathbf{W}_{YH}$ (and their transposes) are the feedforward (feedback) synaptic weight matrices for the first and second layers, respectively. Additionally, $\bar{\mathbf{M}}_H$ and $\bar{\mathbf{M}}_Y$ represent the recurrent synaptic weight matrices for the first and second layers, respectively. A schematic of this neural network is provided in Figure 4.1b. The gain and synaptic weight dynamics described below provide the learning mechanism for this network:

Update dynamics for gains $D_{l,ii}$: By taking the derivative of the cost function with respect to $D_{1,ii}$, as shown in equation (A.12), we can derive the dynamics for the gain variable $D_{1,ii}$ as follows:

$$\mu_{D_1}\frac{dD_{1,ii}(t)}{dt} = -(\lambda_{SM}\beta)(\|\mathbf{M}_{Hi,:}\|^2_{\mathbf{D}_1(t)} - \|\mathbf{W}_{HXi,:}\|_2^2) - (1-\lambda_{SM})\frac{1}{D_{1,ii}(t)}, \quad (4.13)$$

where μ_{D_1} corresponds to the learning time-constant. Similarly, for the gain variable $D_{2,ii}$, the dynamics expression for the corresponding coefficient is given by the following equation, derived from equation (A.13):

$$\mu_{D_2}\frac{dD_{2,ii}(t)}{dt} = -(\lambda_{SM}\beta)(\|\mathbf{M}_{Yi,:}\|^2_{\mathbf{D}_2(t)} - \|\mathbf{W}_{YHi,:}\|_2^2) - (1-\lambda_{SM})\frac{1}{D_{2,ii}(t)}, \quad (4.14)$$

where μ_{D_2} corresponds to the learning time-constant.

The inverses of the inner product weights $D_{l,ii}$ can be interpreted as homeostatic gain parameters. An interesting observation can be made by inspecting the gain updates in (4.13) and (4.14): The homeostatic gain of the neuron is regulated by the balance between the norms of the recurrent and feedforward synaptic strengths, which represent statistical indicators of recent output and input activations, respectively. As a result, the gain will increase if the recent output activations fall behind the level of recent input activations, aiming to balance the input/output energy levels. Conversely, the gain will decrease if the recent output activations surpass the level of recent input activations, again aiming for balance. This mechanism aligns with the experimental observations of homeostatic balance observed in biological neurons [Turrigiano, 1999].

We can write the update dynamics of the synaptic weight matrices in (4.8) as follows based on their recursive definitions:

$$\mathbf{M}_H(t+1) = \gamma^2\mathbf{M}_H(t) + (1-\gamma^2)\mathbf{h}(t)\mathbf{h}(t)^T, \quad (4.15)$$

$$\mathbf{M}_Y(t+1) = \gamma^2\mathbf{M}_Y(t) + (1-\gamma^2)\mathbf{y}(t)\mathbf{y}(t)^T, \quad (4.16)$$

$$\mathbf{W}_{HX}(t+1) = \gamma^2\mathbf{W}_{HX}(t) + (1-\gamma^2)\mathbf{h}(t)\mathbf{x}(t)^T, \quad (4.17)$$

$$\mathbf{W}_{YH}(t+1) = \gamma^2\mathbf{W}_{YH}(t) + (1-\gamma^2)\mathbf{y}(t)\mathbf{h}(t)^T. \quad (4.18)$$

Indeed, these update rules are local since they only depend on variables that are available to the synapse, which makes them biologically plausible.

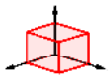
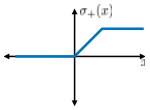
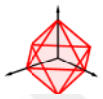
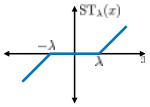
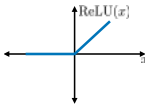
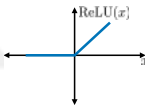
The Det-Max Neural Network obtained for the source domain $\mathcal{P} = \mathcal{B}_\infty$ in this section can indeed be extended to more general identifiable source domains by making changes to the output dynamics. Table 4.1 provides a summary of the output dynamics obtained for specific identifiable source domain examples shown in Figure 2.1. We perform these derivations in the following sections. Based on Table 4.1, we can observe that: (1) In the case of sparse and unit simplex settings, an additional inhibitory neuron is introduced in the neural network architecture. This neuron receives input from all the output neurons and its activation is the inhibitory signal $\lambda_1(\tau)$. (2) The activation functions at the output layer of the neural network are determined by the source attributes, which are defined globally across all sources. Furthermore, the proposed framework can be applied to any polytope described by the general formulation in (4.1). The corresponding Det-Max neural network for such a polytope will contain combinations of the activation functions listed in Table 4.1, as illustrated in Figure 4.1a. This flexibility allows for the modeling of various identifiable source domains with different constraints and characteristics, expanding the applicability of the Det-Max Neural Network. In Section 4.7, we demonstrate one particular example of this case.

4.5.3 Description of Network Dynamics for Bounded Nonnegative Anti-sparse Sources

For nonnegative anti-sparse sources, the network architecture is similar to the antisparse case discussed in Section 4.5.2. The main distinction is that the clipping activation functions at the output layer are replaced with a nonnegative clipping function $\sigma_+(x)$, as shown in Figure 4.2. By using the nonnegative clipping function, the network ensures that the output activations remain nonnegative, aligning with the nonnegative constraint imposed on the sources.

Consequently, the network dynamics corresponding to the nonnegative antisparse case can be written as follows:

Table 4.1: Illustrative source domains depicted in Figure 2.1 and their respective output dynamics.

	Source Domain	**Output Dynamics**	**Output Activation**
	$\mathcal{P} = \mathcal{B}_{\infty,+}$ Nonnegative Anti-sparse	$\boldsymbol{y}_{t,i}(\tau) = \sigma_+\left(\frac{\boldsymbol{u}_i(\tau)}{\Gamma_{Y\,ii}(t)D_{2,ii}(t)}\right)$	
	$\mathcal{P} = \mathcal{B}_1$ Sparse	$\boldsymbol{y}_{t,i}(\tau) = \mathrm{ST}_{\lambda_1(\tau)}\left(\frac{u_i(\tau)}{\lambda_{SM}(1-\beta)\Gamma_{Y\,ii}(t)D_{2,ii}(t)}\right)$ $\frac{da(\tau)}{d\tau} = -a(\tau) + \sum_{k=0}^{n}\lvert\boldsymbol{y}_{t,k}(\tau)\rvert - 1 + \lambda_1(\tau),$ $\lambda_1(\tau) = \mathrm{ReLU}(a(\tau))$	
	$\mathcal{P} = \mathcal{B}_{1,+}$ Nonnegative Sparse	$\boldsymbol{y}_{t,i}(\tau) = \mathrm{ReLU}\left(\frac{u_i(\tau)}{\lambda_{SM}(1-\beta)\Gamma_{Y\,ii}(t)D_{2,ii}(t)} - \lambda_1(\tau)\right)$ $\frac{da(\tau)}{d\tau} = -a(\tau) + \sum_{k=0}^{n}\boldsymbol{y}_{t,k}(\tau) - 1 + \lambda_1(\tau),$ $\lambda_1(\tau) = \mathrm{ReLU}(a(\tau))$	
	$\mathcal{P} = \Delta$ Unit Simplex	$\boldsymbol{y}_{t,i}(\tau) = \mathrm{ReLU}\left(\frac{u_i(\tau)}{\lambda_{SM}(1-\beta)\Gamma_{Y\,ii}(t)D_{2,ii}(t)} - \lambda_1(\tau)\right)$ $\frac{d\lambda_1(\tau)}{d\tau} = -\lambda_1(\tau) + \sum_{k=0}^{n}\boldsymbol{y}_{t,k}(\tau) - 1 + \lambda_1(\tau)$	

Update dynamics for the hidden layer $\mathbf{h}(t)$:

$$\begin{aligned}\frac{d\mathbf{v}(\tau)}{d\tau} &= -\mathbf{v}(\tau) - \lambda_{SM}[((1-\beta)\bar{\mathbf{M}}_H(t) + \beta\mathbf{D}_1(t)\bar{\mathbf{M}}_H(t)\mathbf{D}_1(t))\mathbf{h}(\tau) \\ &\quad +\beta\mathbf{D}_1(t)\mathbf{W}_{HX}(t)\mathbf{x}(\tau) + (1-\beta)\mathbf{W}_{YH}(t)^T\mathbf{D}_2(t)\mathbf{y}(\tau)] \\ \mathbf{h}(t;\tau)_i &= \sigma_A\left(\frac{\mathbf{v}(\tau)_i}{\lambda_{SM}\Gamma_{Hii}(t)((1-\beta) + \beta D_{1,ii}(t)^2)}\right) \quad \text{for } i = 1,\dots n,\end{aligned}$$

Update dynamics for the output $\mathbf{y}(t)$:

$$\begin{aligned}\frac{d\mathbf{u}(\tau)}{d\tau} &= -\mathbf{u}(\tau) + \mathbf{W}_{YH}(t)\mathbf{h}(\tau) - \bar{\mathbf{M}}_Y(t)\mathbf{D}_2(t)\mathbf{y}(\tau) \\ \mathbf{y}(t;\tau)_i &= \sigma_+\left(\frac{\mathbf{u}(\tau)_i}{\Gamma_{Y\,ii}(t)D_{2,ii}(t)}\right), \quad \text{for } i = 1,\dots n.\end{aligned}$$

Figure 4.3 illustrates the neural network that corresponds to nonnegative anti-sparse sources.

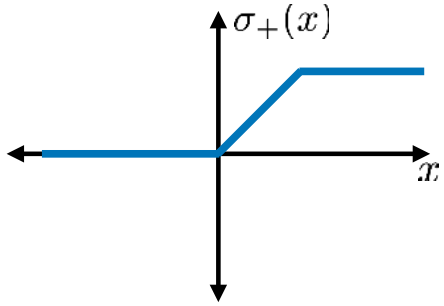


Figure 4.2: Nonnegative clipping function utilized for elementwise projection onto the set $\mathcal{B}_{\infty,+}$.

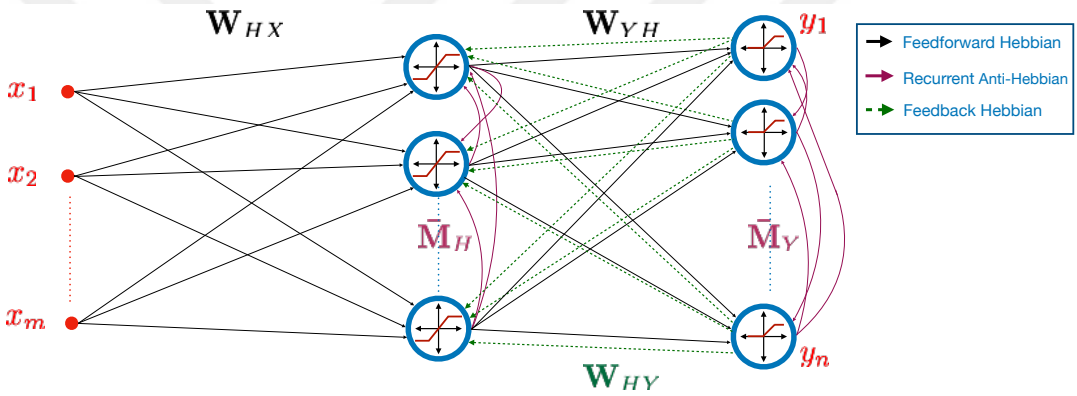


Figure 4.3: Det-Max WSM neural network for the extraction of nonnegative anti-sparse sources ($\mathcal{P} = \mathcal{B}_{\infty,+}$).

4.5.4 Description of Network Dynamics for Nonnegative Sparse Sources

We consider the following optimization setting for the nonnegative sparse sources, where $\mathcal{P} = \mathcal{B}_{1,+}$:

$$\begin{aligned} \underset{\mathbf{h}(t),\mathbf{y}(t)}{\text{minimize}} \quad & \beta c_1(\mathbf{h}(t)) + (1-\beta)c_2(\mathbf{h}(t),\mathbf{y}(t)) \\ \text{subject to} \quad & \|\mathbf{y}(t)\|_1 \leq 1, \quad \mathbf{y}(t) \geq 0 \end{aligned} \tag{4.19}$$

for which we can write the Lagrangian based reformulation as follows

$$\underset{\lambda_1 \geq 0}{\text{maximize}} \; \underset{\mathbf{h}(t),\mathbf{y}(t)}{\text{minimize}} \quad \beta c_1(\mathbf{h}(t)) + (1-\beta)c_2(\mathbf{h}(t),\mathbf{y}(t)) + \lambda_1(\|\mathbf{y}(t)\|_1 - 1)$$

We note that the updates for $\mathbf{h}(t)$, the synaptic weights matrices, and gain variables $D_{1,ii}, D_{2,ii}$ follow the equations provided in Section 4.5.2.

Note that the cost function is an ℓ_1 regularized quadratic cost function for the output component $\mathbf{y}_t$. We following the primal-dual approach presented in [Pehlevan et al., 2017a] to obtain the dynamic equations for output dynamics as

$$\begin{aligned}\frac{d\mathbf{u}(\tau)}{d\tau} &= -\mathbf{u}(\tau) + \lambda_{SM}(1-\beta)[\mathbf{W}_{YH}(t)\mathbf{h}(\tau) - \bar{\mathbf{M}}_Y(t)\mathbf{D}_2(t)\mathbf{y}(\tau)],\\ \mathbf{y}(t;\tau)_i &= \text{ReLU}\left(\frac{\mathbf{u}(\tau)_i}{\lambda_{SM}(1-\beta)\Gamma_{Yii}(t)D_{2,ii}(t)} - \lambda_1(\tau)\right), \quad \text{for } i = 1,\dots n,\end{aligned}$$

where $\text{ReLU}(x)$ is the rectified-linear unit mapping which is defined by

$$\text{ReLU}(x) = \begin{cases} x & x > 0,\\ 0 & \text{otherwise.}\end{cases}$$

The update dynamic of the Lagrangian variable $\lambda_1(\tau)$ can obtained based on the dual maximization:

$$\frac{da(\tau)}{d\tau} = -a(\tau) + \sum_{k=0}^{n}\mathbf{y}(t;\tau)_k - 1 + \lambda_1(\tau), \qquad \lambda_1(\tau) = \text{ReLU}(a(\tau)). \tag{4.20}$$

Based on the derived expressions, apart from the hidden layer and output layer neurons, an extra neuron is present which represents the Lagrangian variable λ_1. The dynamics of this neuron are governed by equation (4.20). This particular neuron generates an inhibitory signal for the output neurons, determined by the cumulative output activation. Figure 4.4 illustrates the network structure corresponding to this configuration.

4.5.5 Description of Network Dynamics for Sparse Sources

In the scenario of sparse sources, specifically when $\mathcal{P} = \mathcal{B}_1$, the only modification compared to the nonnegative sparse case involves substituting the ReLU output activation function with the soft thresholding function $ST_\lambda(.)$ which is defined as

$$ST_\lambda(x) = \begin{cases} 0 & |x| \le \lambda\\ x - sign(x)\lambda & \text{otherwise.}\end{cases}$$

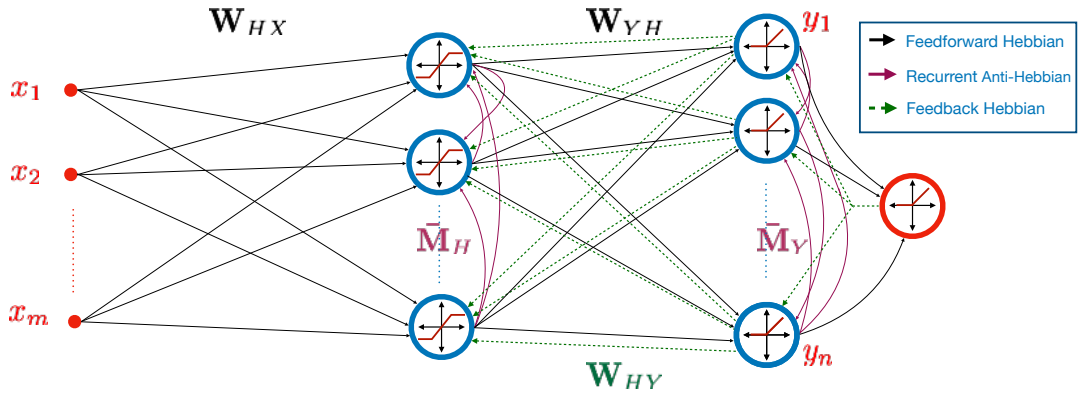


Figure 4.4: Det-Max WSM neural network for the extraction of nonnegative sparse sources ($\mathcal{P} = \mathcal{B}_{1,+}$).

Hence, we can express the output dynamics for the case of $\mathcal{P} = \mathcal{B}_1$ as follows:

$$\begin{aligned}
\frac{d\mathbf{u}(\tau)}{d\tau} &= -\mathbf{u}(\tau) + \lambda_{SM}(1-\beta)[\mathbf{W}_{YH}(t)\mathbf{h}(\tau) - \bar{\mathbf{M}}_Y(t)\mathbf{D}_2(t)\mathbf{y}(\tau)], \\
\mathbf{y}(t;\tau)_i &= \mathrm{ST}_{\lambda_1(\tau)}\left(\frac{\mathbf{u}(\tau)_i}{\lambda_{SM}(1-\beta)\Gamma_{Yii}(t)D_{2,ii}(t)}\right), \quad \text{for } i = 1,\dots n, \\
\frac{da(\tau)}{d\tau} &= -a(\tau) + \sum_{k=0}^{n} |\mathbf{y}(t;\tau)_k| - 1 + \lambda_1(\tau), \qquad \lambda_1(\tau) = \mathrm{ReLU}(a(\tau)).
\end{aligned}$$

Figure 4.5 depicts the WSM-based Det-Max neural network designed for sparse BSS.

4.6 Description of Network Dynamics for Unit Simplex Sources

The unit simplex set Δ forms a facet of the polytope $\mathcal{P} = \mathcal{B}_{1,+}$, which represents the domain for nonnegative sparse sources. Consequently, we replace the inequality constraint involving the ℓ_1-norm in (4.19) with an equality constraint, resulting in the Det-Max WSM optimization problem formulated for the unit simplex domain:

$$\begin{aligned}
\underset{\mathbf{h}(t),\mathbf{y}(t)}{\text{minimize}} \quad & \beta c_1(\mathbf{h}(t)) + (1-\beta)c_2(\mathbf{h}(t),\mathbf{y}(t)) \\
\text{subject to} \quad & \|\mathbf{y}(t)\|_1 = 1, \quad \mathbf{y}(t) \geq 0
\end{aligned}$$

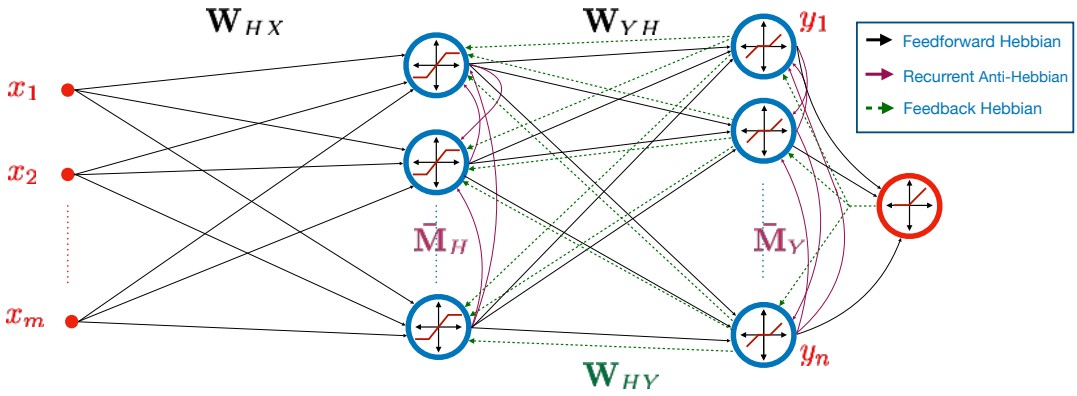


Figure 4.5: Det-Max WSM neural network for the extraction of sparse sources ($\mathcal{P} = \mathcal{B}_1$).

Therefore, for the Lagrangian-based optimization can be written as

$$\underset{\lambda_1}{\text{maximize}} \; \underset{\mathbf{h}(t),\mathbf{y}(t)}{\text{minimize}} \quad \beta c_1(\mathbf{h}(t)) + (1-\beta)c_2(\mathbf{h}(t),\mathbf{y}(t)) + \lambda_1(\|\mathbf{y}(t)\|_1 - 1),$$

where λ is no longer required to be nonnegative. Thus, for $\mathcal{P} = \Delta$, the only necessary modification compared to $\mathcal{B}_{1,+}$ is to replace the ReLU activation function of the rightmost inhibition neuron in Figure 4.4 with a linear activation. Consequently, the output dynamics for unit simplex sources can be expressed as follows:

$$\begin{aligned}
\frac{d\mathbf{u}(\tau)}{d\tau} &= -\mathbf{u}(\tau) + \lambda_{SM}(1-\beta)[\mathbf{W}_{YH}(t)\mathbf{h}(\tau) - \bar{\mathbf{M}}_Y(t)\mathbf{D}_2(t)\mathbf{y}(\tau)], \\
\mathbf{y}(t;\tau)_i &= \text{ReLU}\left(\frac{\mathbf{u}(\tau)_i}{\lambda_{SM}(1-\beta)\Gamma_{Yii}(t)D_{2,ii}(t)} - \lambda_1(\tau)\right), \quad \text{for } i = 1,\dots n, \\
\frac{d\lambda_1(\tau)}{d\tau} &= -\lambda_1(\tau) + \sum_{k=0}^{n} \mathbf{y}(\tau)_k - 1 + \lambda_1(\tau).
\end{aligned}$$

Figure 4.6 depicts the WSM-based Det-Max neural network for the unit-simplex sources.

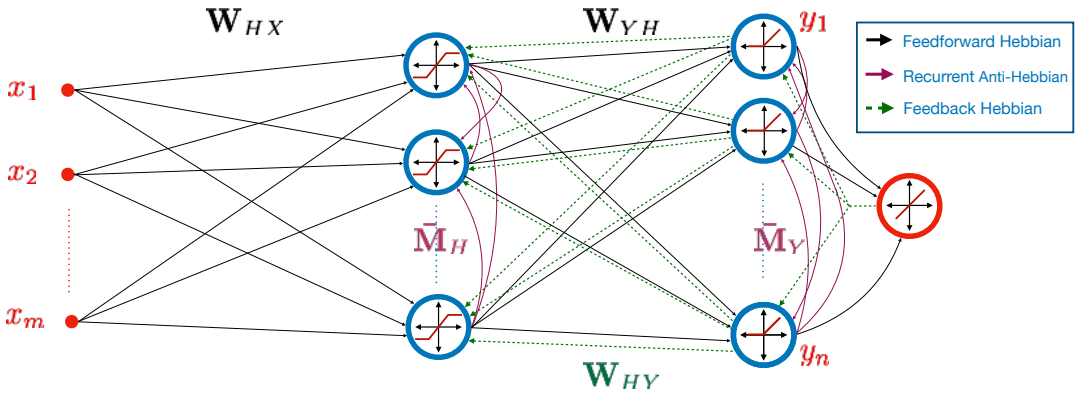


Figure 4.6: Det-Max WSM neural network for the extraction of unit-simplex sources ($\mathcal{P} = \Delta$).

4.7 Description of Network Dynamics for Sources with Mixed Attributes

We consider the following polytope example provided in [Tatli and Erdogan, 2021b]

$$\mathcal{P}_{ex} = \left\{ \mathbf{s} \in \mathbb{R}^3 \;\middle|\; \begin{array}{l} s_1, s_2 \in [-1,1], s_3 \in [0,1], \\ \left\| \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \right\|_1 \leq 1, \left\| \begin{bmatrix} s_2 \\ s_3 \end{bmatrix} \right\|_1 \leq 1 \end{array} \right\}, \tag{4.21}$$

which is an instance of domains where source attributes, such as nonnegativity and sparsity, are defined solely at the subvector level.

The Det-Max WSM optimization formulation for this scenario can be given as:

$$\begin{aligned} \underset{\mathbf{h}(t),\mathbf{y}(t)}{\text{minimize}} \quad & C(\mathbf{h}(t),\mathbf{y}(t)) \\ \text{subject to} \quad & \left\| \begin{bmatrix} \mathbf{y}(t)_1 \\ \mathbf{y}(t)_2 \end{bmatrix} \right\|_1 \leq 1, \quad \left\| \begin{bmatrix} \mathbf{y}(t)_2 \\ \mathbf{y}(t)_3 \end{bmatrix} \right\|_1 \leq 1, \quad \mathbf{y}(t)_3 \geq 0 \end{aligned}$$

for which the Lagrangian based reformulation can be written as

$$\begin{aligned} \underset{\lambda_1,\lambda_2 \geq 0}{\text{maximize}} \; \underset{\substack{\mathbf{h}(t),\mathbf{y}(t)_3 \geq 0, \\ \mathbf{y}(t)_1,\mathbf{y}(t)_2}}{\text{minimize}} \quad & C(\mathbf{h}(t),\mathbf{y}(t)) \\ & + \lambda_1 \left(\left\| \begin{bmatrix} \mathbf{y}(t)_1 \\ \mathbf{y}(t)_2 \end{bmatrix} \right\|_1 - 1 \right) + \lambda_2 \left(\left\| \begin{bmatrix} \mathbf{y}(t)_2 \\ \mathbf{y}(t)_3 \end{bmatrix} \right\|_1 - 1 \right). \end{aligned}$$

The proximal operator corresponding to the Lagrangian terms can be defined as

$$\text{prox}_{\lambda_1,\lambda_2}(\mathbf{v}) = \underset{q_3\geq 0, q_1, q_2}{\text{argmin}} \left(\frac{1}{2}\|\mathbf{v}-\mathbf{q}\|_2^2 + \lambda_1 \left\| \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \right\|_1 + \lambda_2 \left\| \begin{bmatrix} q_2 \\ q_3 \end{bmatrix} \right\|_1 \right). \quad (4.22)$$

Let us denote $\mathbf{q}^*$ the output of the proximal operator. Based on the subdifferential set-based optimality condition, we have the following:

- if $q_1^* \neq 0$ then $q_1^* - v_1 + \lambda_1 \text{sign}(v_1) = 0$ which implies $q_1^* = v_1 - \lambda_1 \text{sign}(v_1)$,
- if $q_2^* \neq 0$ then $q_2^* = v_2 - (\lambda_1 + \lambda_2)\text{sign}(v_2)$,
- if $q_3^* \neq 0$ then $q_3^* = v_3 - \lambda_2$.

Hence, we can express $q_1 = \text{ST}_{\lambda_1}(v_1)$, $q_2 = \text{ST}_{\lambda_1+\lambda_2}(v_2)$ and $q_3 = \text{ReLU}(v_3 - \lambda_2)$. Consequently, we can represent the corresponding output dynamics expressions in the following form:

$$\begin{aligned}
\frac{du(\tau)}{d\tau} &= -u(\tau) + \lambda_{SM}(1-\beta)[\mathbf{W}_{YH}(t)\mathbf{h}(\tau) - \bar{\mathbf{M}}_Y(t)\mathbf{D}_2(t)\mathbf{y}(\tau)], \\
\mathbf{y}(t;\tau)_1 &= \text{ST}_{\lambda_1(\tau)}\left(\frac{\mathbf{u}(\tau)_1}{\lambda_{SM}(1-\beta)\Gamma_{Y11}(t)D_{2,11}(t)}\right), \\
\mathbf{y}(t;\tau)_2 &= \text{ST}_{\lambda_1(\tau)+\lambda_2(\tau)}\left(\frac{\mathbf{u}(\tau)_2}{\lambda_{SM}(1-\beta)\Gamma_{Y22}(t)D_{2,22}(t)}\right), \\
\mathbf{y}(t;\tau)_3 &= \text{ReLU}\left(\frac{\mathbf{u}(\tau)_3}{\lambda_{SM}(1-\beta)\Gamma_{Y33}(t)D_{2,33}(t)} - \lambda_2(\tau)\right), \\
\frac{da_1(\tau)}{d\tau} &= -a_1(\tau) + |\mathbf{y}(t;\tau)_1| + |\boldsymbol{y}(t;\tau)_2| - 1 + \lambda_1(\tau), \\
\lambda_1(\tau) &= \text{ReLU}(a_1(\tau)), \\
\frac{da_2(\tau)}{d\tau} &= -a_2(\tau) + |\mathbf{y}(t;\tau)_2| + \boldsymbol{y}(t;\tau)_3 - 1 + \lambda_2(\tau), \\
\lambda_2(\tau) &= \text{ReLU}(a_2(\tau)).
\end{aligned}$$

Figure 4.7 illustrates the Det-Max WSM neural network for the source domain in (4.21).

4.8 Numerical Experiments

In this section, we present the applications of the proposed WSM-based BSS framework to synthetic and natural sources. To facilitate the replication of our results,

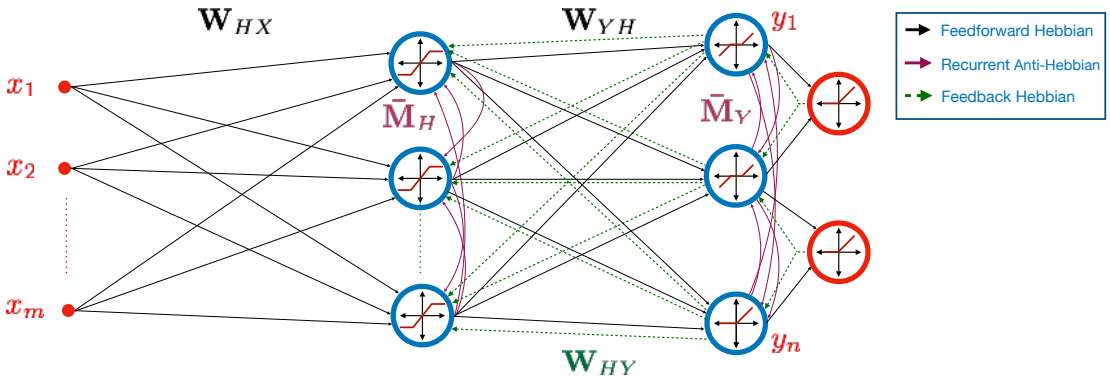


Figure 4.7: Det-Max WSM neural network corresponding to the polytope in (4.21).

we have made our implementation code publicly available[1].

4.8.1 On the Computation of Neural Dynamics

Update dynamics for the hidden states $\mathbf{h}(t)$ and the output vector $\mathbf{y}(t)$ are defined by differential equations, which depend on the choice of the source domain and result in recursive neural dynamic iterations. We implemented these neural dynamics on a digital hardware (utilizing *Python 3.7*) and Algorithm 2 summarizes the neural dynamic iterations specifically for anti-sparse sources for our simulatins based on the discussion in Section 4.5.2. Similar calculations for the output dynamics can be obtained for each source assumption based on the derivations presented earlier in this thesis. We iterate the neural dynamics until a convergence check is satisfied or a predetermined maximum number of iterations $\tau_{\max}$ is reached. In Algorithm 2, ϵ represents the tolerance used in the relative error check for the stopping condition, and $\eta(\tau)$ denotes the learning rate at iteration count τ. In the following subsections, we provide additional experimental details and source separation examples for various assumptions on the sources.

[1]https://github.com/BariscanBozkurt/Biologically-Plausible-DetMaxNNs-for-Blind-Source-Separation

Algorithm 2 Neural dynamic iterations for anti-sparse sources

1: Initialize $\tau_{\max}$, ϵ, and $\tau = 1$
2: **while** $(||\mathbf{v}(\tau) - \mathbf{v}(\tau-1)||/||\mathbf{v}(\tau)|| > \epsilon$ or $||\mathbf{u}(\tau) - \mathbf{u}(\tau-1)||/||\mathbf{u}(\tau)|| > \epsilon)$ and $\tau < \tau_{\max}$ **do**
3: $\quad \mathbf{v}(\tau) = \mathbf{v}(\tau-1) + \eta(\tau)\frac{d\mathbf{v}(\tau-1)}{d(\tau-1)}$
4: $\quad$ Apply Equation 4.10 for $\mathbf{h}(t;\tau)_i$
5: $\quad \mathbf{u}(\tau) = \mathbf{u}(\tau-1) + \eta(\tau)\frac{d\mathbf{u}(\tau-1)}{d(\tau-1)}$
6: $\quad$ Apply Equation 4.12 for $\mathbf{y}(t;\tau)_i$
7: $\quad \tau = \tau + 1$
8: **end while**

4.8.2 Synthetically Correlated Source Separation with Nonnegative Anti-sparse Sources

To demonstrate the correlated source separation capability of the proposed WSM neural networks, we conduct a numerical experiment involving five copula-T distributed sources that are both uniform and correlated. To generate the correlation calibration matrix for these sources, we utilize a Toeplitz matrix with its first row set as $\begin{bmatrix}1 & \rho & \rho & \rho & \rho\end{bmatrix}$. The correlation level is determined by the parameter ρ, which we vary within the range of $[0, 0.6]$. These sources are mixed using a 10×5 random matrix with independent and identically distributed (i.i.d.) standard normal random variables. Additionally, the mixtures are corrupted by i.i.d. normal noise corresponding to a signal-to-noise ratio (SNR) level of 30dB. In this experiment, we employ the nonnegative-antisparse-WSM neural network (Figure 4.3 in Section 4.5.3) since the sources are nonnegative uniform random variables.

We conducted a comparison of the signal-to-interference-plus-noise-power-ratio (SINR) performance between our algorithm and several existing algorithms, including the NSM algorithm [Pehlevan et al., 2017b], the Infomax ICA algorithm [Bell and Sejnowski, 1995], as implemented in Python's MNE Toolbox [Gramfort et al., 2013], the LD-InfoMax algorithm [Erdogan, 2022], and the PMF algorithm [Tatli and Erdogan, 2021b]. In Figure 4.8, we present the SINR performances of these algorithms, averaged over 300 realizations, plotted against the correlation parameter

ρ. Our WSM-based network exhibits favorable performance even in the presence of source correlations. Conversely, the NSM and ICA algorithms, which assume uncorrelated or independent sources, show noticeable degradation as the correlation levels increase. Additionally, the batch Det-Max algorithms, namely LD-InfoMax and PMF, demonstrate robustness against source correlations. Moreover, as expected due to their batch nature, these algorithms typically achieve better performance compared to our neural network with online-restriction.

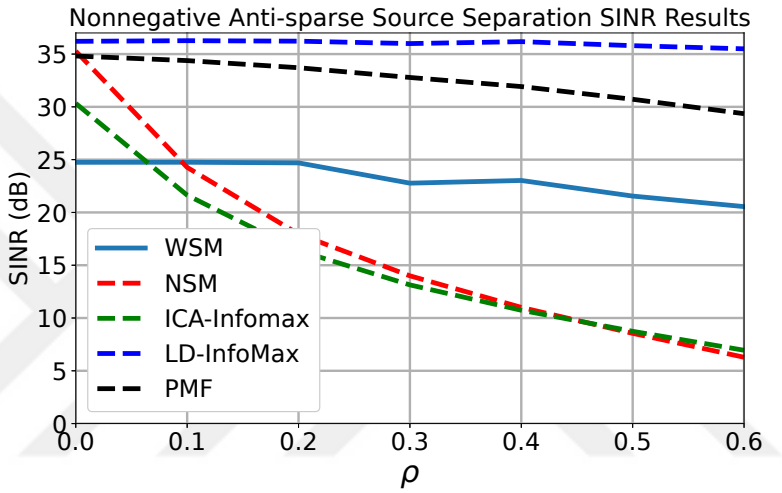


Figure 4.8: The SINR performances of the WSM, NSM, ICA, LD-InfoMax, and PMF algorithms, evaluated with respect to the correlation parameter ρ

4.8.3 Synthetically Correlated Source Separation with Anti-sparse Sources

To demonstrate the correlated source separation capabilities of WSM neural networks with antisparse sources, we conducted a similar numerical experiment with four copula-T distributed sources in the range of $[-1, 1]$. The correlation calibration matrix utilized in this experiment is a Toeplitz matrix with its first row set as $\begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 \end{bmatrix}$, where ρ represents the correlation level within the range of $\rho \in [0, 0.6]$. The sources were mixed using an 8×4 random matrix with independent and identically distributed (i.i.d.) standard normal entries. The mixtures were corrupted by i.i.d. standard normal noise, resulting in a signal-to-noise ratio (SNR) level of 30dB. For this experiment, we employed the antisparse-WSM neural

network illustrated in Figure 4.1b. The performance of the WSM algorithm was compared against the BSM algorithm [Erdogan and Pehlevan, 2020], the Infomax ICA algorithm [Gramfort et al., 2013], the PMF algorithm [Tatli and Erdogan, 2021b], and the LD-InfoMax algorithm [Erdogan, 2022]. We evaluated the SINR performance of each algorithm to assess their respective separation capabilities. In Figure 4.9, we present the SINR performances of these algorithms, averaged over 100 realizations, as a function of the correlation parameter ρ. Similar to the results obtained in the experiments for nonnegative antisparse source separation (discussed in Section 4.8.2), the WSM approach demonstrates its robustness against source correlations. Conversely, the BSM and ICA algorithms, which assume uncorrelated or independent sources, exhibit degradation in performance as the source correlation increases. The LD-InfoMax and PMF algorithms show relatively similar SINR behaviors, with their performance remaining relatively stable even with increasing source correlation. We note that both the PMF and LD-InfoMax algorithms typically achieve better performances compared to our proposed online algorithm, as these approaches employ batch algorithms.

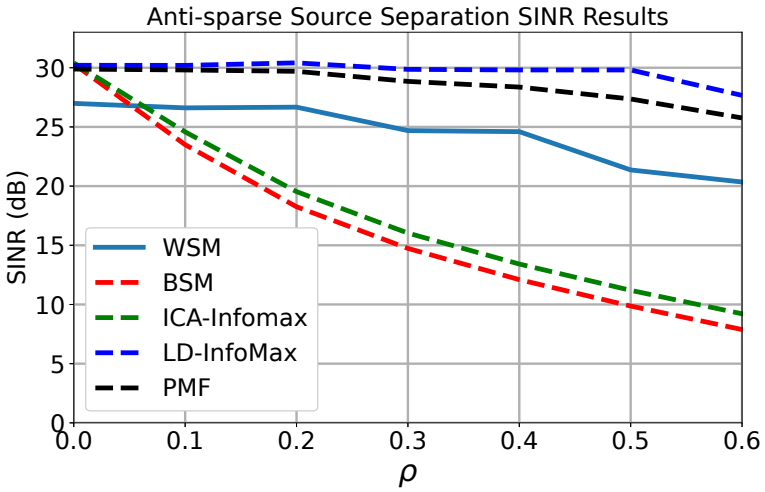


Figure 4.9: The SINR performances of the WSM, BSM, ICA, LD-InfoMax, and PMF algorithms, evaluated with respect to the correlation parameter ρ

4.8.4 Image Separation

To further highlight the advantage of our approach in correlated source separation, we consider a natural image separation scenario. In this example, we have three RGB images of size $324 \times 432 \times 3$ as sources (see Figure 4.10a). The sample Pearson correlation coefficients between the images are $\rho_{12} = 0.263$, $\rho_{13} = 0.066$, and $\rho_{23} = 0.333$. We employ a random 5×3 mixing matrix with entries drawn from an i.i.d. standard normal distribution. The resulting mixtures are shown in Figure 4.10b.

ICA-InfoMax, NSM, WSM, LD-InfoMax and PMF algorithms are applied to the mixtures. The corresponding output images are illustrated in Figure 4.10c, 4.10d, 4.10e, 4.10f, and 4.10g for ICA-InfoMax, NSM, WSM, LD-InfoMax, and PMF, respectively. The Infomax ICA algorithm achieved an SINR level of 13.92dB, resulting in residual interference effects in the output images. The NSM algorithm achieved a significantly higher SINR level of 17.45dB, leading to visually improved output images. Our algorithm achieved a higher SINR level of 27.49dB, resulting in output images closely resembling the original source images. The LD-InfoMax algorithm achieved an SINR level of 28.65dB, while the PMF algorithm achieved an SINR level of 31.92dB. As expected, both the PMF and LD-InfoMax algorithms achieved better performance due to their batch nature. However, our proposed approach's output is compatible with these frameworks.

(a)

Figure 4.10: (a) Original RGB images.

(b)

(c)

(d)

(e)

(f)

Figure 4.10: (b) mixture RGB images, (c) ICA outputs, (d) NSM outputs, (e) WSM outputs, (f) LD-InfoMax Outputs.

(g)

Figure 4.10: (g) PMF Outputs.

4.8.5 Sparse Source Separation

To demonstrate the applicability of the proposed framework for a different source domain, we consider sparse sources where $\mathcal{P} = \mathcal{B}_1$. We generate source vectors in $n = 5$ dimensions by projecting i.i.d. uniform vectors from $\mathcal{B}_\infty$ to $\mathcal{B}_1$. The mixing matrix used is a 10×5 matrix with i.i.d. standard normal entries. These mixed signals are then employed to train the sparse-WSM Det-Max network depicted in Figure 4.5, which was introduced in Section 4.5.5.

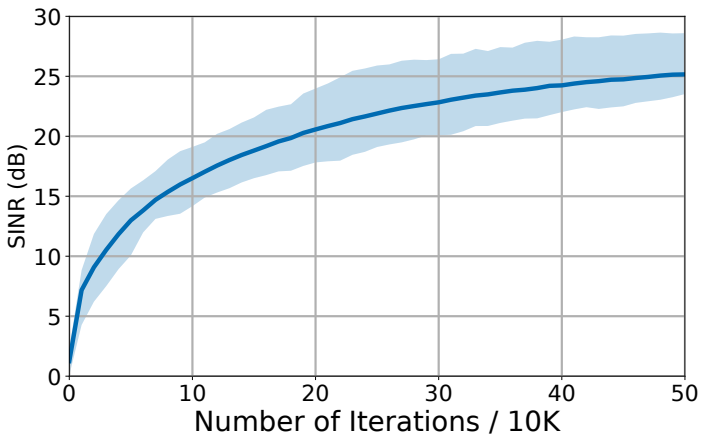


Figure 4.11: The convergence curve of the sparse-WSM algorithm for the SINR, considering an input SNR level of 30dB. The curve is represented by the mean (solid line) along with the 25th and 75th percentile envelopes.

Figure 4.11 showcases the convergence behavior of the sparse-WSM network in terms of SINR as a function of update iterations, considering an input SNR level of 30dB. The results are averaged over 200 realizations. Furthermore, Figure 4.12 showcases the performance of the sparse-WSM network in terms of signal separation for various noise levels.

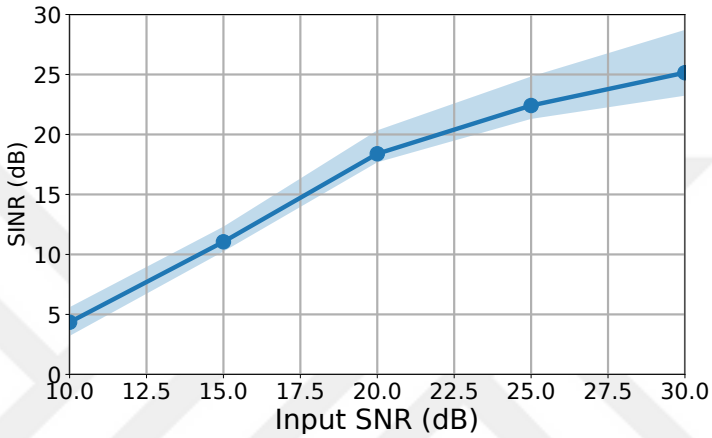


Figure 4.12: The output SINR as a function of the input SNR level for the sparse-WSM algorithm, displayed using the mean (solid line) along with the 25th and 75th percentile envelopes.

To conduct a comparative analysis between our online approach and the batch algorithms LD-InfoMax and PMF, we conducted experiments with these algorithms using an input SNR level of 30 dB. The averaged SINR results of each algorithm over 200 realizations for this input SNR level are summarized in Table 4.2. Our observations from these experiments indicate that both PMF and LD-InfoMax achieve higher SINR performances on average when compared to our WSM Det-Max network. This outcome can be attributed to the batch nature of both PMF and LD-InfoMax, as discussed previously.

4.8.6 Sparse Dictionary Learning

In this study, we investigate the application of sparse coding to the task of dictionary learning for natural image patches [Olshausen and Field, 1997]. Specif-

Table 4.2: Averaged SINR results for sparse source separation using the WSM, PMF, and LD-InfoMax algorithms.

Algorithm	**WSM**	**PMF**	**LD-InfoMax**
SINR	25.14	30.17	30.0

ically, we utilize 12×12 prewhitened image patches obtained from the website http://www.rctn.org/bruno/sparsenet. These patches are vectorized and employed to train the sparse Det-Max WSM neural network depicted in Figure 4.5.

Figure 4.13 displays the receptive field images obtained from the columns of the inverse of the sparse-WSM separator. These images correspond to localized Gabor-like edge features. This example validates that the sparse WSM neural network, equipped with a local update rule, effectively captures the behavior observed in the primary visual cortical neurons of primates.

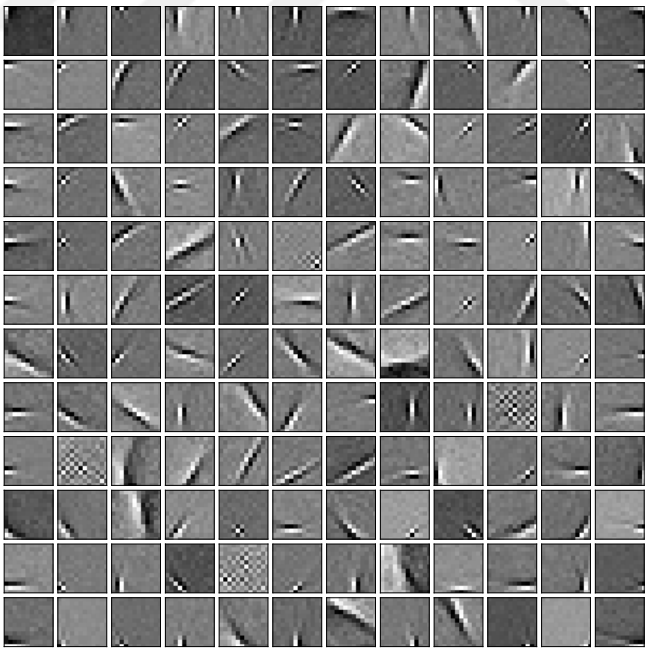


Figure 4.13: Dictionary obtained from the natural image patches through the sparse-WSM Network.

4.8.7 Source Separation with Mixed Latent Attributes

In this section, we present a series of experiments that showcase the source separation capabilities of the WSM Neural Network in various identifiable-enabling polytopes, similar to the example discussed in Section 4.7. These experiments aim to demonstrate the effectiveness of the proposed network in handling general identifiable polytopes. To ensure the identifiability of the polytopes used in this section, we employ the graph automorphism-based identifiability characterization algorithm presented in [Bozkurt and Erdogan, 2022] (as well as in Section 3 of this thesis) for verification.

Special Polytope Example in Section 4.7

We present the results of numerical experiments conducted using the WSM Det-Max network depicted in Figure 4.7, corresponding to the polytope defined in (4.21). To utilize this WSM Det-Max Neural Network, we synthetically generate $n = 3$ dimensional uniform vectors within this polytope. These vectors are then mixed using a randomly generated 6×3 matrix with i.i.d. standard normal entries. Furthermore, the resulting mixtures are corrupted by i.i.d. standard normal noise, maintaining a SNR level of 30dB.

Figure 4.14 illustrates the behavior of the overall SINR, individual source SNRs, as well as the diagonal weight matrices ($\mathbf{D}_1$ and $\mathbf{D}_2$), with respect to the number of update iterations for a single experiment. To assess the average behavior of this neural network, we conduct experiments for a total of 100 different source and mixing matrix generations. Figure 4.15 showcases the averaged SINR convergence behavior, along with the 25/75-percentile envelope, as a function of update iterations.

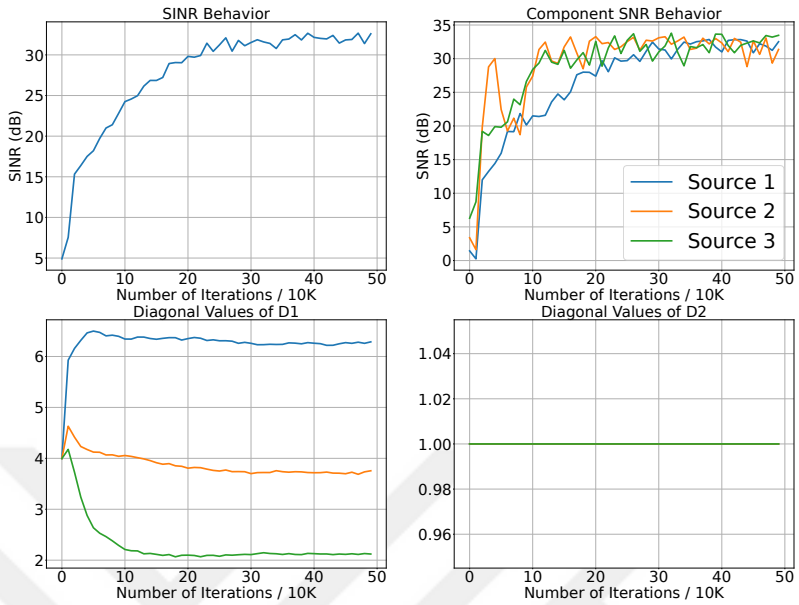


Figure 4.14: Example behaviors of SINR, component SNR values, and diagonal weights of $\mathbf{D}_1$ and $\mathbf{D}_2$ for a single experiment corresponding to the polytope in (4.21)

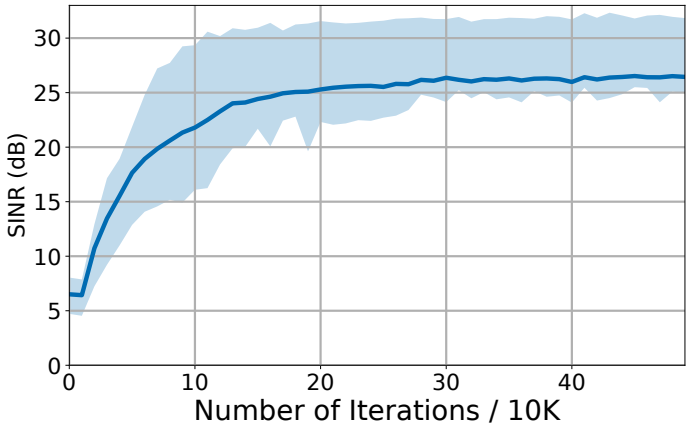


Figure 4.15: The convergence curve of SINR for the experiments corresponding to the polytope defined in (4.21), represented by the mean (solid line) along with the 25th and 75th percentile envelopes.

Mixed Anti-sparse and Nonnegative Anti-sparse Sources

As an additional example of an identifiable polytope, we consider a set that assigns mixed antisparse attributes to the source components, specifically signed or nonnegative. In this experiment, we randomly select two components to be nonnegative, while the remaining three components are antisparse as specified in (4.23). The mixing matrix utilized is a 10×5 matrix with i.i.d. standard normal entries. The mixtures obtained from this setup are employed to train the WSM Det-Max network, similar to the architecture illustrated in Figure 4.1b. However, in this case, the clippings at the output layer corresponding to nonnegative sources are replaced with nonnegative clippings to accommodate the mixed antisparse attributes of the sources. Figure 4.16 illustrates the SINR convergence for this experiment averaged over 100 trials.

$$\mathcal{P} = \left\{ \mathbf{s} \in \mathbb{R}^3 \;\middle|\; s_{j_1}, s_{j_2}, s_{j_3} \in [-1, 1], s_{j_4}, s_{j_5} \in [0, 1], j_i \in \{1, 2, 3, 4, 5\} \right\}, \quad (4.23)$$

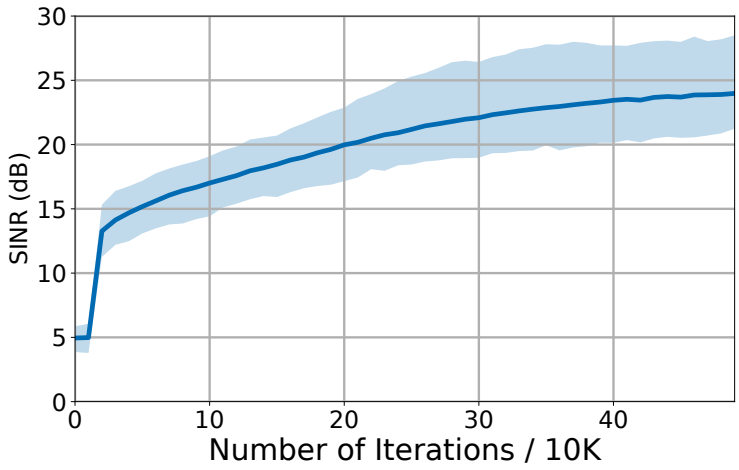


Figure 4.16: The convergence curve of SINR for the experiments corresponding to the polytope defined in (4.23), represented by the mean (solid line) along with the 25th and 75th percentile envelopes.

Mixed Sparse and Nonnegative Anti-sparse Sources

In this final illustration of source separation on identifiable domains, we consider the polytope,

$$\mathcal{P} = \left\{ \mathbf{s} \in \mathbb{R}^3 \;\middle|\; s_{j_1} \in [0,1], \left\| \begin{bmatrix} s_{j_2} \\ s_{j_3} \\ s_{j_4} \\ s_{j_5} \end{bmatrix} \right\|_1 \leq 1, j_i \in \{1,2,3,4,5\} \right\}, \tag{4.24}$$

where only one component is nonnegative and the subvector containing the remaining components is sparse. To demonstrate the source separation capability of the WSM Det-Max Neural Network for this specific domain, we generate $n = 5$ dimensional uniform vectors within this polytope. The generated sources are mixed with a random 10×5 matrix with standard normal entries. Figure 4.17 showcases the SINR convergence behavior of the WSM Det-Max network for this scenario, averaged over 100 realizations, as a function of update iterations.

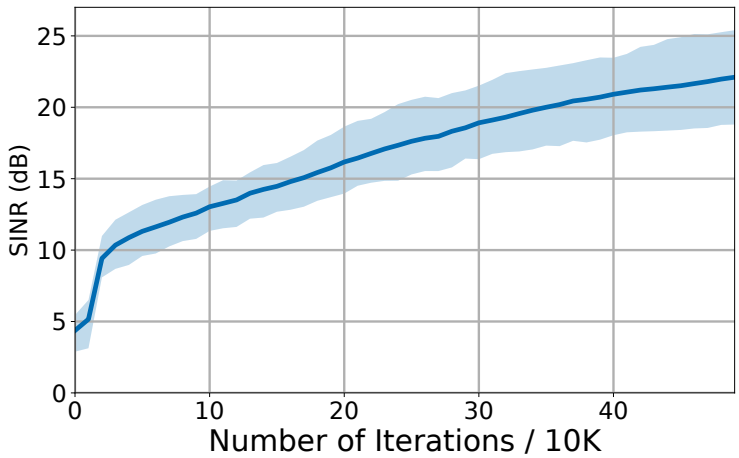


Figure 4.17: The convergence curve of SINR for the experiments corresponding to the polytope defined in Equation (4.24), represented by the mean (solid line) along with the 25th and 75th percentile envelopes.

4.8.8 Digital Communication Example: 4-PAM Modulation Scheme

We extend the application of blind source separation to the 4 Pulse-amplitude modulation (4-PAM) scheme, which represents a realistic scenario for digital communication signals. In this scheme, the transmitted symbols are $\pm 3, \pm 1$, and we assume a uniform symbol distribution, where $P(s = i) = \frac{1}{4}$ for $i = \pm 3, \pm 1$, with s denoting the transmitted symbol.

For this experiment, we consider the transmission of 5 sources, each consisting of 400,000 samples. These sources are mixed using a random 10×5 matrix with standard normal entries. Without loss of generality, we assume the source domain assumption to be $\mathcal{B}_{\ell_\infty}$, and feed the mixed signals into the WSM Det-Max neural network designed for antisparse sources.

Figure 4.18 illustrates the SINR convergence behavior of the WSM Det-Max network for this scenario, averaged over 20 realizations, as a function of update iterations. The results demonstrate that our proposed approach successfully separates the source symbols from their mixtures, indicating its effectiveness in recovering the original transmitted information.

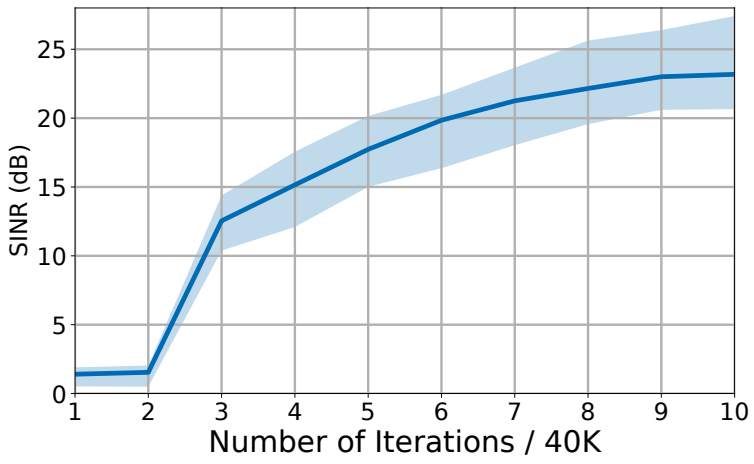


Figure 4.18: The SINR convergence curve for 4-PAM digital communication signals, depicted by the mean (solid line) along with the 25th and 75th percentile envelopes.

4.9 Conclusion

Combining recent advances in Det-Max-based unsupervised matrix factorization methods and a two-layer WSM architecture enabling arbitrary linear transformations adaptable by local learning rules, we proposed a general framework for constructing biologically plausible neural networks capable of separating both independent and correlated sources. Based on the domain choices for sources, this new framework offers a principled way to obtain infinitely many networks. Furthermore, we demonstrated the successful correlated source separation capability for both natural and synthetic sources, filling an important gap in the literature on biologically plausible networks.

Our networks share similar limitations with other recurrent, biologically-plausible BSS networks. Firstly, the performance of the algorithms can be significantly influenced by certain hyperparameters, as discussed in Appendix A.4. Specifically, the inner product gains (D_{ii}) are sensitive to the combined choices of algorithm parameters, necessitating careful tuning. Secondly, the numerical experiments with our neural networks are relatively slow due to the recursive computations involved in equations (4.9)-(4.10) and (4.11)-(4.12) for hidden layer and output vectors. This computational burden is a common characteristic of biologically-plausible recurrent source separation networks, as outlined in Appendix A.5. Potential approaches to address this issue include early-stopping the recursive computation, as proposed by [Minden et al., 2018].

Chapter 5

CORRELATIVE INFORMATION MAXIMIZATION BASED BIOLOGICALLY PLAUSIBLE NEURAL NETWORKS FOR CORRELATED SOURCE SEPARATION

5.1 Abstract

The brain possesses a remarkable ability to extract underlying causes of stimuli, yet the mechanisms responsible for this at the network level remain unknown. Previous attempts to tackle this problem have proposed neural networks based on independent component analysis, which assumes that the latent causes are mutually independent. However, we relax this assumption and propose a biologically plausible neural network that can extract correlated latent sources by leveraging information about their domains. To construct this network, we define the separation objective as maximizing the correlative information from the inputs to the outputs, while restricting the output vectors to a set where the source vectors are assumed to reside. By formulating this optimization problem in an online manner, we naturally arrive at neural networks with local learning rules. Our framework allows for an infinite number of choices for the source domain, enabling flexible modeling of complex latent structures. Specifically, selecting simplex or polytopic source domains leads to networks with piecewise-linear activation functions. We provide numerical examples to illustrate the superior capability of our approach in separating correlated sources, both for synthetic and natural sources.

The contributions presented in this chapter has been published in [Bozkurt et al., 2023a] as conference paper at the eleventh International Conference on Learning Representations (ICLR 2023).

5.2 Introduction

Blind source separation (BSS) algorithms often rely on normative principles, with the information maximization principle being one of the most important ones. This principle aims to maximize the information transferred from input mixtures to separator outputs while satisfying a specific generative assumption about the sources. However, quantifying information transfer using Shannon mutual information can be challenging, especially in data-driven adaptive applications, due to its dependence on the joint and conditional densities of the input and output components. To address this challenge, the independent component analysis (ICA) framework introduces joint densities into separable forms by assuming source independence [Bell and Sejnowski, 1995]. Recently, [Bahroun et al., 2021] presented a biologically plausible neural network algorithm for the fourth-order blind identification (FOBI) [Cardoso, 1989], which can separate independent sources from their linear mixtures. However, in certain scenarios, assuming the mutual independence of latent causes in real observations may not be plausible [Träuble et al., 2021]. To accommodate potential dependence among latent components, [Erdogan, 2022] recently proposed using the second-order statistics-based *"correlative (log-determinant) mutual information"* maximization for BSS (see Section 2.2). This approach eliminates the need for the independence assumption, enabling correlated source separation.

In this chapter, we introduce a novel framework called CorInfoMax, which is an online correlative information maximization-based biologically plausible neural network designed to address the blind source separation (BSS) problem. Our motivations for proposing this framework are as follows:

- The correlative mutual information objective function used in our framework relies solely on the second-order statistics of the inputs and outputs. This characteristic eliminates the requirement for computationally expensive higher-order statistics or joint probability density function (PDF) estimates.

- The optimization problem associated with the correlative mutual information maximization in our framework is equivalent to maximizing the correlation or

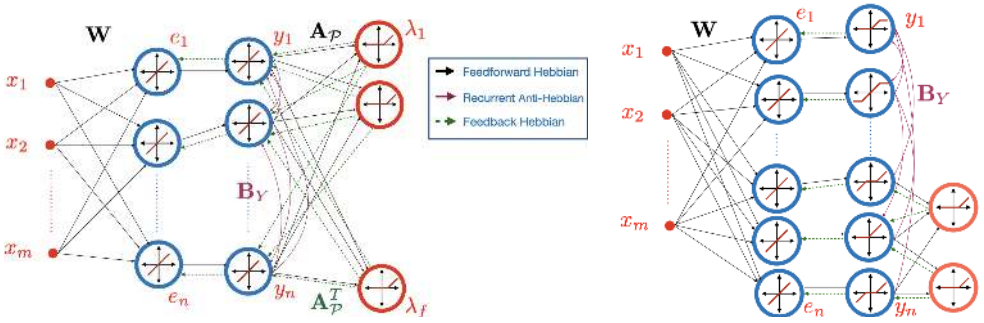
linear dependence between the input mixtures and separator outputs, which aligns well with the concept of a linear inverse problem

- One key advantage of the CorInfoMax framework is that it relies on the information about the source domain while eliminating the need for the source independence assumption. This characteristic allows the neural networks constructed within this framework to effectively separate correlated sources. Additionally, the CorInfoMax framework enables the construction of networks corresponding to infinitely many source domains, encompassing the combinations of various attributes such as sparsity, nonnegativity, and other relevant characteristics.

- A notable aspect of the proposed objective in the CorInfoMax framework is that its optimization naturally gives rise to learning with local update rules.

- The CorInfoMax framework serves as a unifying framework that can generate biologically plausible neural networks for diverse unsupervised data decomposition methods aimed at obtaining structured latent representations, such as nonnegative matrix factorization (NMF) [Fu et al., 2019], sparse component analysis (SCA) [Babatas and Erdogan, 2018], bounded component analysis (BCA) [Erdogan, 2013, Inan and Erdogan, 2014] and polytopic matrix factorization (PMF) [Tatli and Erdogan, 2021b].

Figure 5.1 presents the visualization of CorInfoMax neural networks for two distinct source domain representation choices, which are three-layer neural networks employing piecewise linear activation functions. It is important to highlight that the CorInfoMax framework can be extended beyond the task of blind source separation. It can also be leveraged to learn structured representations from data by maximizing the correlative information transfer between the input data and the chosen structured domain at the network's output.



(a) (b)

Figure 5.1: CorInfoMax BSS neural networks designed for two distinct canonical source domain representations. In both representations, x_i's and y_i's denote the input (mixtures) and output (separator) signals, respectively. The feedforward weights are denoted by $\mathbf{W}$, while e_i's represent the errors between the transformed inputs and outputs. The lateral weights at the output, $\mathbf{B_y}$, correspond to the inverse of the output autocorrelation matrix. For canonical form (a), the Lagrangian interneurons, λ_i, impose source domain constraints. The matrix $\mathbf{A}_{\mathcal{P}}$ ($\mathbf{A}_{\mathcal{P}}^T$) represents the feedforward (feedback) connections between the outputs and interneurons. In canonical form (b), the interneurons on the right enforce sparsity constraints on specific subsets of outputs.

5.3 Blind Source Separation Setting

In this section, we provide a review of the BSS problem, which is also discussed in Section 4.3. This review serves two purposes: 1) it provides a refresher for readers who may have skipped Chapter 4, 2) it allows us to introduce the notation used in this section, which may have slight differences compared to Chapter 4.

SOURCES: In our BSS setting, we consider a scenario where there are a finite number of n-dimensional source vectors, represented by the set $\mathbb{S} = \{\mathbf{s}(1), \mathbf{s}(2), \ldots, \mathbf{s}(N)\} \subset \mathcal{P}$, where $\mathcal{P}$ is a specific subset of $\mathbb{R}^n$. The choice of the source domain $\mathcal{P}$ plays a crucial role in determining the identifiability of the sources from their mixtures, as well as the properties and relationships among the individual sources. Structured unsupervised matrix factorization methods are typically defined based

on the presumed source domain. Some of these methods include:

i. *Normalized nonnegative sources in the NMF(SSMF) framework*: $\Delta = \{\mathbf{s} \mid \mathbf{s} \geq 0, \mathbf{1}^T\mathbf{s} = 1\}$. Its applications include hyperspectral unmixing and text mining in signal processing and machine learning domain [Abdolali and Gillis, 2021], [Fu et al., 2016].

ii. *Bounded antisparse sources in the BCA framework*: $\mathcal{B}_{\ell_\infty} = \{\mathbf{s} \mid \|\mathbf{s}\|_\infty \leq 1\}$. One of its applications include digital communication signals [Erdogan, 2013].

iii. *Bounded sparse sources in the SCA framework*: $\mathcal{B}_{\ell_1} = \{\mathbf{s} \mid \|\mathbf{s}\|_1 \leq 1\}$. This presumed source domain can be used to model efficient representations of stimulus such as sound [Smith and Lewicki, 2006] and vision [Olshausen and Field, 1997].

iv *Nonnegative bounded antiparse sources in the nonnegative-BCA framework*: $\mathcal{B}_{\ell_\infty,+} = \mathcal{B}_{\ell_\infty} \cap \mathbb{R}^n_+$. It can be potentially used in applications including natural images [Erdogan, 2013].

v. *Nonnegative bounded sparse sources in the nonnegative-SCA framework*: $\mathcal{B}_{\ell_1,+} = \mathcal{B}_{\ell_1} \cap \mathbb{R}^n_+$. Its potential applications are similar to Δ in (i).

In the study conducted by [Tatli and Erdogan, 2021b], it was demonstrated that there exists an infinite number of polytopes that satisfy certain symmetry restrictions and enable the identifiability of sources in the BSS problem. Several notable ones of these polytopes are included in items (ii) to (v) of the preceding enumeration. Furthermore, the H-representation is a commonly used canonical form for describing polytopes [Grünbaum et al., 1967]

$$\mathcal{P} = \{\mathbf{y} \in \mathbb{R}^n | \mathbf{A}_\mathcal{P}\mathbf{y} \preccurlyeq \mathbf{b}_\mathcal{P}\}, \tag{5.1}$$

which corresponds to the intersection of half-spaces. Alternatively, as presented in [Bozkurt et al., 2022] (see (4.1) in Section 4.3 for details), we can also consider *feature-based* polytopes defined in terms of attributes:

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

In this chapter, we consider both representations of the polytopes above.

MIXING: In our analysis, we consider a linear generative model, where the source vectors are mixed using an unknown matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x}(i) = \mathbf{A}\mathbf{s}(i), \quad \forall i = 1, \ldots, N$, where we assume the overdetermined case, i.e., $m \geq n$, and $rank(\mathbf{A}) = n$. We can define the mixture matrix as $\mathbf{X} = \begin{bmatrix} \mathbf{x}(1) & \ldots & \mathbf{x}(N) \end{bmatrix}$.

SEPARATION: The objective of the source separation scenario is to retrieve the original source matrix $\mathbf{S}$ from the mixture matrix $\mathbf{X}$ up to possible scaling and/or permutation ambiguities. In other words, the output vectors $\{\mathbf{y}(i)\}$ of the separator satisfy $\mathbf{y}(i) = \mathbf{W}\mathbf{x}(i) = \mathbf{\Pi}\mathbf{\Lambda}\mathbf{s}(i)$, for all $i = 1, \ldots, N$. Here, the matrix $\mathbf{W} \in \mathbb{R}^{n \times m}$ represents the learned weights of the separator. $\mathbf{\Pi}$ denotes a permutation matrix, $\mathbf{\Lambda}$ is a full-rank diagonal matrix, and $\mathbf{y}(i)$ represents the estimated source corresponding to sample index i. Therefore, if we define the source estimate matrix as $\mathbf{Y} = \begin{bmatrix} \mathbf{y}(1) & \mathbf{y}(2) & \ldots & \mathbf{y}(N) \end{bmatrix} = \mathbf{W}\mathbf{X} \in \mathbb{R}^{n \times N}$, the "ideal separation" condition refers to $\mathbf{Y} = \mathbf{\Pi}\mathbf{\Lambda}\mathbf{S}$.

5.4 Method: Biologically Plausible Neural Networks for Correlative Information Maximization

5.4.1 Online Optimization Setting for Correlative Mutual Information Maximization

To initiate the online optimization formulation for CorInfoMax (whose batched formulation is reviewed in Section 2.2), we begin by substituting the output and error sample autocorrelation matrices in (2.16) with their respective weighted counterparts,

$$\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k) = \frac{1-\zeta_{\mathbf{y}}}{1-\zeta_{\mathbf{y}}^{k}} \sum_{i=1}^{k} \zeta_{\mathbf{y}}^{k-i} \mathbf{y}(i)\mathbf{y}(i)^{T} \quad \hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k) = \frac{1-\zeta_{\mathbf{e}}}{1-\zeta_{\mathbf{e}}^{k}} \sum_{i=1}^{k} \zeta_{\mathbf{e}}^{k-i} \mathbf{e}(i)\mathbf{e}(i)^{T}, \quad (5.2)$$

Here, $\zeta_{\mathbf{y}}$ is the forgetting factor which is a small positive value close to but less than 1. The matrix $\mathbf{W}(i)$ represents the best linear minimum mean square error (MMSE) estimator matrix used to estimate $\mathbf{y}$ from $\mathbf{x}$, and the vector $\mathbf{e}(i) = \mathbf{y}(i) - \mathbf{W}(i)\mathbf{x}(i)$ is the corresponding error. Consequently, we define the corresponding

online CorInfoMax optimization problem as

$$\underset{\mathbf{y}(k)\,\in\,\mathbb{R}^n}{\text{maximize}} \quad \mathcal{J}(\mathbf{y}(k)) = \frac{1}{2}\big(\log\det(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I}) - \log\det(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})\big) \tag{5.3a}$$

$$\text{subject to} \quad \mathbf{y}(k)\in\mathcal{P}. \tag{5.3b}$$

Note that the aforementioned formulation assumes prior knowledge of the best linear MMSE matrix $\mathbf{W}(i)$. The update of this matrix is determined by solving the following online regularized least squares problem,

$$\underset{\mathbf{W}(i)\,\in\,\mathbb{R}^{n\times m}}{\text{minimize}} \quad \mu_{\mathbf{W}}\|\mathbf{y}(i)-\mathbf{W}(i)\mathbf{x}(i)\|_2^2 + \|\mathbf{W}(i)-\mathbf{W}(i-1)\|_F^2. \tag{5.4a}$$

The optimal point of (5.4) corresponds to the famous least mean square update for the estimator $\mathbf{W}(i)$, which converges to the vicinity of the best MMSE estimator for sufficiently small $\mu_{\mathbf{W}}$ [Sayed, 2003].

5.4.2 Description of the Network Dynamics for Sparse Sources

Here, we demonstrate that maximizing the online CorInfoMax objective in Equation (5.3) through gradient ascent corresponds to the neural dynamics of a multilayer recurrent neural network using local learning rules. Moreover, the chosen source domain $\mathcal{P}$ influences the output activation functions and can introduce additional inhibitory neurons. To provide an illustrative example, this section focuses on the sparse special case discussed in Section 5.3, where $\mathcal{P}=\mathcal{B}_{\ell_1}$. The corresponding Lagrangian optimization setting can written as

$$\underset{\lambda\geq 0}{\text{minimize}}\ \underset{\mathbf{y}(k)\in\mathbb{R}^n}{\text{maximize}} \quad \mathcal{L}(\mathbf{y}(k),\lambda(k)) = \mathcal{J}(\mathbf{y}(k)) - \lambda(k)(\|\mathbf{y}(k)\|_1 - 1). \tag{5.5}$$

To derive network dynamics, we employ the proximal gradient update [Parikh et al., 2014] for $\mathbf{y}(k)$ using the expression (B.11) for $\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k))$, which is derived in Appendix B.1. Additionally, we utilize the projected gradient descent update for $\lambda(k)$ with $\nabla_\lambda\mathcal{L}(\mathbf{y}(k),\lambda(k)) = 1-\|\mathbf{y}(k;\nu+1)\|_1$. These lead to the following iterative

updates:

$$\mathbf{e}(k;\nu) = \mathbf{y}(k;\nu) - \mathbf{W}(k)\mathbf{x}(k) \tag{5.6}$$

$$\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) = \gamma_{\mathbf{y}}(k)\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)\mathbf{y}(k;\nu) - \gamma_{\mathbf{e}}(k)\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)\mathbf{e}(k;\nu), \tag{5.7}$$

$$\mathbf{y}(k;\nu+1) = ST_{\lambda(k;\nu)}\left(\mathbf{y}(k;\nu) + \eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu))\right) \tag{5.8}$$

$$\lambda(k;\nu+1) = \mathrm{ReLU}\Big(\lambda(k;\nu) - \eta_{\lambda}(\nu)(1 - \|\mathbf{y}(k;\nu+1)\|_1)\Big). \tag{5.9}$$

In the given iterative updates, $\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)$ and $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)$ represent the inverses of $\mathbf{R}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)$ and $\mathbf{R}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)$, respectively. The values of $\gamma_{\mathbf{y}}(k)$ and $\gamma_{\mathbf{e}}(k)$ are defined in equations (B.8) and (B.10), respectively. The iteration index is denoted by $\nu \in \mathbb{N}$, and $\eta_{\mathbf{y}}(\nu)$ and $\eta_{\lambda}(\nu)$ represent the learning rates for the output and λ at iteration ν, respectively. Furthermore, the function $\mathrm{ReLU}(\cdot)$ denotes the rectified linear unit, and $ST_{\lambda}(\cdot)$ is the soft-thresholding nonlinearity defined as follows:

$$ST_{\lambda}(\mathbf{y})_i = \begin{cases} 0 & |y_i| \leq \lambda, \\ y_i - sign(y_i)\lambda & \text{otherwise} \end{cases}$$

Also, it should be noted that in the final iteration ν_{final}, the values are represented as $\mathbf{y}(k) = \mathbf{y}(k;\nu_{final})$ and $\mathbf{e}(k) = \mathbf{e}(k;\nu_{final})$, with some abuse of notation.

The neural dynamic iterations presented in (5.6)- (5.9) define a recurrent neural network. In this network, the matrix $\mathbf{W}(k)$ is feedforward synaptic connections from the input $\mathbf{x}(k)$ to the error $\mathbf{e}(k)$, $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)$ corresponds to feedforward connections from the error $\mathbf{e}(k)$ to the output $\mathbf{y}(k)$, and $\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)$ denotes the lateral synaptic connections among the output components. In the following, we will examine the learning rules for this network.

Update of inverse correlation matrices $\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)$ and $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)$: We apply matrix inversion lemma to $(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k) + \epsilon\boldsymbol{I})^{-1}$ and $(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k) + \epsilon\boldsymbol{I})^{-1}$ in order to obtain (B.7) and (B.9), as derived in Appendix B.1. As a result, we obtain the update expressions as follows,

$$\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k+1) = \frac{1-\zeta_{\mathbf{y}}^k}{\zeta_{\mathbf{y}} - \zeta_{\mathbf{y}}^k}(\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k) - \gamma_{\mathbf{y}}(k)\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k)\mathbf{y}(k)^T\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)), \tag{5.10}$$

$$\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k+1) = \frac{1-\zeta_{\mathbf{e}}^k}{\zeta_{\mathbf{e}} - \zeta_{\mathbf{e}}^k}(\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k) - \gamma_{\mathbf{e}}(k)\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)\mathbf{e}(k)\mathbf{e}(k)^T\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)). \tag{5.11}$$

However, it is important to note that the updates in (5.10) and (5.11) violate biological plausibility because the multipliers $\gamma_{\mathbf{y}}$ and $\gamma_{\mathbf{e}}$ depend on all output and error components, which goes against the principle of locality. Additionally, the update in (5.11) is not local as it only depends on the feedforward signal $\boldsymbol{z}_e(k) = \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)\mathbf{e}(k)$ entering into output neurons: to update $[\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}]_{ij}$, the synaptic connection between the output neuron i and the error neuron j, one requires $[\boldsymbol{z}_e]_j$ which is a signal input to the output neuron j. To address these issues and ensure biological plausibility, we make the following observations and assumptions:

- $\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k) + \epsilon\boldsymbol{I} \approx \epsilon\boldsymbol{I} \Rightarrow \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k+1) \approx \frac{1}{\epsilon}\boldsymbol{I}$, since we presume that the error $\mathbf{e}(k)$ to converge near zero in the noiseless linear observation model, which is a reasonable assumption.

- Assuming $\zeta_{\mathbf{y}}$ is close enough to 1, and the time step k is large enough, $\gamma_{\mathbf{y}}(k)$ can be approximated as $\dfrac{1-\zeta_{\mathbf{y}}}{\zeta_{\mathbf{y}}}$.

Hence, we can modify the update equation of $\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k+1)$ in (5.10) as

$$\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k+1) = \frac{1}{\zeta_{\mathbf{y}}}(\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k) - \frac{1-\zeta_{\mathbf{y}}}{\zeta_{\mathbf{y}}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k)\mathbf{y}(k)^T\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)), \tag{5.12}$$

and approximate $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k+1)$ with $\epsilon^{-1}\boldsymbol{I}$ instead of updating.

Feed-forward synaptic connections $\mathbf{W}(k)$: The solution to the online optimization in (5.4) can be expressed as

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \mu_{\mathbf{W}}(k)\mathbf{e}(k)\mathbf{x}(k)^T, \tag{5.13}$$

where $\mu_{\mathbf{W}}(k)$ is the adaptive step size corresponding to the adaptive least-mean-squares (LMS) update, which is based on the MMSE criterion [Sayed, 2003]. The learning dynamics and output of the Sparse CorInfoMax are summarized in Algorithm 3.

Neural Network Realizations: Figure 5.2a depicts the three-layer implementation of the sparse CorInfoMax neural network. This architecture is derived from the network dynamics expressions given in (5.6)- (5.9) and the approximation $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k) \approx \frac{1}{\epsilon}\boldsymbol{I}$. The first layer of neurons in the network corresponds to the error ($\mathbf{e}(k)$), while

the second layer corresponds to the output ($\mathbf{y}(k)$) neurons, which employ soft thresholding activation functions. By substituting (5.6) and $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k+1) = \frac{1}{\epsilon}\boldsymbol{I}$ into (5.7), we obtain $\nabla_{\mathbf{y}(k)} J(\mathbf{y}(k;\nu)) = \mathbf{M}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k;\nu) + \frac{\gamma_{\mathbf{e}}}{\epsilon}\mathbf{W}(k)\mathbf{x}(k)$ where $\mathbf{M}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k) = \gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k) - \frac{\gamma_{\mathbf{e}}}{\epsilon}\boldsymbol{I}$. Hence, the gradient expression and (5.8)- (5.9) correspond to the two-layer network depicted in Figure 5.2b.

Algorithm 3 Sparse CorInfoMax Algorithm

Input: Streaming data (mixtures) $\{\mathbf{x}(k) \in \mathbb{R}^m\}_{k=1}^N$

Output: Estimated Sources $\{\mathbf{y}(k) \in \mathbb{R}^n\}_{k=1}^N$.

1: Initialize $\zeta_{\mathbf{y}}, \zeta_{\mathbf{e}}, \mu(1)_{\mathbf{W}}, \mathbf{W}(1), \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)$, $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)$.

2: **for** k = 1, 2, . . . , N **do**

3: **run neural output dynamics until convergence:**

$$\begin{aligned}
\mathbf{e}(k;\nu) &= \mathbf{y}(k;\nu) - \mathbf{W}(k)\mathbf{x}(k) \\
\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) &= \gamma_{\mathbf{y}}(k)\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)\mathbf{y}(k;\nu) - \gamma_{\mathbf{e}}(k)\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)\mathbf{e}(k;\nu), \\
\mathbf{y}(k;\nu+1) &= ST_{\lambda(k;\nu)}\left(\mathbf{y}(k;\nu) + \eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu))\right) \\
\lambda(k;\nu+1) &= \mathrm{ReLU}(\lambda(k;\nu) - \eta_{\lambda}(\nu)(1 - \|\mathbf{y}(k;\nu+1)\|_1)),
\end{aligned}$$

4: Update feedforward synapses: $\mathbf{W}(k+1) = \mathbf{W}(k) + \mu_{\mathbf{W}}(k)\mathbf{e}(k)\mathbf{x}(k)^T$

5: Update lateral synapses: $\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k+1) = \frac{1}{\zeta_{\mathbf{y}}}(\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k) - \frac{1-\zeta_{\mathbf{y}}}{\zeta_{\mathbf{y}}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k)\mathbf{y}(k)^T\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k))$

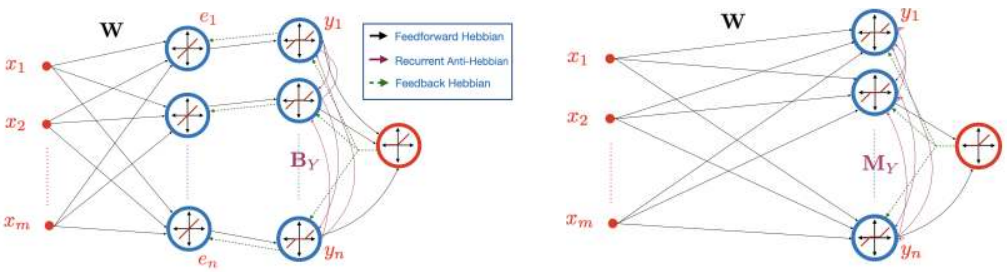
6: **end for**

5.4.3 Description of the Network Dynamics for a Canonical Polytope Representation

In this section, we address the optimization problem defined in (5.3) for a generic polytope with H-representation given by (5.1). We can formulate the corresponding online optimization setting in Lagrangian Min-Max form as follows

$$\underset{\boldsymbol{\lambda}(k)\succcurlyeq 0}{\text{minimize}} \; \underset{\mathbf{y}(k)\in\mathbb{R}^n}{\text{maximize}} \qquad \mathcal{L}(\mathbf{y}(k), \boldsymbol{\lambda}(k)) = \mathcal{J}(\mathbf{y}(k)) - \boldsymbol{\lambda}(k)^T(\mathbf{A}_{\mathcal{P}}\mathbf{y}(k) - \mathbf{b}_{\mathcal{P}}) \tag{5.14}$$

To derive the recursive update dynamics for the network output and the Lagrangian variable, we differentiate the objective in (5.14) with respect to $\mathbf{y}(k)$ and



(a) (b)

Figure 5.2: Sparse CorInfoMax Network: (a) three-layer (b) two-layer. x_i's and y_i's denote the input (mixtures) and output (separator) signals, respectively. The feedforward weights are denoted by $\mathbf{W}$, while e_i's in the three-layer implementation (on the left) represent the errors between the transformed inputs and outputs. $\mathbf{B_Y}$ in (a) correspond to the inverse of the output autocorrelation matrix and represents lateral weights at the output. In representation (b), the output lateral weights, denoted by $\mathbf{M_Y}$, are a diagonally modified form of the inverse of the output correlation. In both representations, the rightmost interneuron enforces the sparsity constraint.

$\boldsymbol{\lambda}(k)$. The derivatives are given by:

$$\nabla_{\mathbf{y}(k)}\mathcal{L}(\mathbf{y}(k;\nu),\boldsymbol{\lambda}(k;\nu)) = \gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)\mathbf{y}(k;\nu) - \gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)\mathbf{e}(k;\nu) - \mathbf{A}_{\mathcal{P}}^{T}\boldsymbol{\lambda}(k;\nu), \tag{5.15}$$

$$\nabla_{\boldsymbol{\lambda}(k)}\mathcal{L}(\mathbf{y}(k;\nu)) = -\mathbf{A}_{\mathcal{P}}\mathbf{y}(k;\nu) + \mathbf{b}_{\mathcal{P}}. \tag{5.16}$$

Recursive update dynamics for $\mathbf{y}(k)$ and $\boldsymbol{\lambda}(k)$: In order to solve the optimization problem in (5.14), we utilize projected gradient updates for $\mathbf{y}(k)$ and $\boldsymbol{\lambda}(k)$. These updates result in the following neural dynamic iterations:

$$\mathbf{y}(k;\nu+1) = \mathbf{y}(k;\nu) + \eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}\mathcal{L}(\mathbf{y}(k;\nu),\boldsymbol{\lambda}(k;\nu)), \tag{5.17}$$

$$\boldsymbol{\lambda}(k,\nu+1) = \mathrm{ReLU}\left(\boldsymbol{\lambda}(k,\nu) - \eta_{\boldsymbol{\lambda}}(\nu)(\mathbf{b}_{\mathcal{P}} - \mathbf{A}_{\mathcal{P}}\mathbf{y}(k;\nu))\right). \tag{5.18}$$

In these iterations, $\eta_{\boldsymbol{\lambda}}(\nu)$ represents the learning rate for $\boldsymbol{\lambda}$ at iteration ν. Observing the recurrent neural network structure, we can make the following observations: i) The output neurons employ linear activation functions since $\mathbf{y}(k)$ is unconstrained in (5.14), ii) The network includes f interneurons corresponding to the Lagrangian

vector $\boldsymbol{\lambda}$, where f represents the number of rows of $\mathbf{A}_{\mathcal{P}}$ in (5.14), or equivalently, the number of $(n-1)$-faces of the corresponding polytope, iii) The non-negativity constraint on $\boldsymbol{\lambda}$ implies that the interneurons adopt ReLU activation functions. The neural network architecture corresponding to the neural dynamics in (5.17)-(5.18) is depicted in Figure 5.1a. It consists of a layer of f interneurons that enforce the polytopic constraint specified in (5.1). The updates of $\mathbf{W}(k)$ and $\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)$ follow the equations provided in Section 5.4.2. While the architecture shown in Figure 5.1a enables the implementation of arbitrary polytopic source domains, the value of f can be quite large. Alternatively, we can consider a subset of polytopes described by individual properties and the relationships between source components, as defined in (4.1). In this case, we can derive the network dynamics for the *feature-based* polytope representation in Section 5.4.8, and its specific realization is illustrated in Figure 5.1b. The number of interneurons in this case corresponds to the number of sparsity constraints in (4.1), which can be significantly fewer than the number of faces of the polytope.

5.4.4 Description of the Network Dynamics for Antisparse Sources

In the case of the source domain $\mathcal{P} = \mathcal{B}_{\ell_\infty}$, which corresponds to antisparse sources, we can derive the network for the antisparse CorInfoMax through the projected gradient ascent method. Unlike the sparse CorInfoMax case, we do not require any interneurons related to the projection operation since projection onto the $\mathcal{B}_{\ell_\infty}$ is simply an elementwise clipping operation

Recursive update dynamics for $\mathbf{y}(k)$: Using the gradient of (5.3a) with respect to $\mathbf{y}(k)$ in (B.11), which we derived in Appendix B.1, we can express the projected gradient ascent iterations for (5.3) as

$$\begin{aligned}
\mathbf{e}(k;\nu) &= \mathbf{y}(k;\nu) - \mathbf{W}(k)\mathbf{x}(k), \\
\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) &= \gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k;\nu) - \gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)\mathbf{e}(k;\nu), \\
\mathbf{y}(k;\nu+1) &= \sigma_1\left(\mathbf{y}(k;\nu) + \eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu))\right),
\end{aligned}$$

where $\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)$ and $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)$ represent the inverses of $\mathbf{R}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)$ and $\mathbf{R}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)$ respectively, $\nu \in \mathbb{N}$ is the index of neural dynamic iterations, $\sigma_1(.)$ is the projection onto

the presumed domain $\mathcal{B}_{\ell_\infty}$, which is defined as

$$\sigma_1(\mathbf{y})_i = \begin{cases} y_i & -1 \leq y_i \leq 1, \\ \text{sign}(y_i) & \text{otherwise.} \end{cases}$$

.

Figure 5.3 demonstrates the corresponding realization of the neural network.

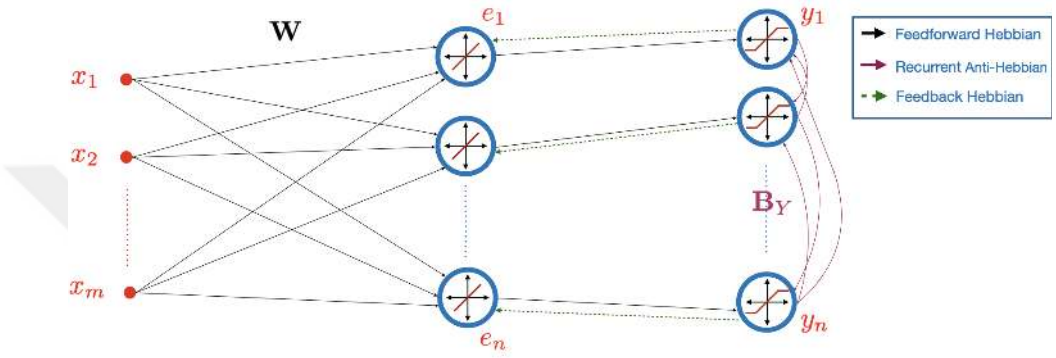


Figure 5.3: Two-Layer antisparse CorInfoMax Network. x_i's represent the inputs (mixtures) and y_i's represent the output (separator) signals. $\mathbf{W}$ denotes the feedforward weights. e_i's represent the errors between the transformed inputs and outputs. $\mathbf{B_Y}$ represents the lateral weights at the output, which is the inverse of the output autocorrelation matrix. The output nonlinearities are clipping functions.

5.4.5 Description of the Network Dynamics for Nonnegative Antisparse Sources

The treatment for the presumed source domain $\mathcal{P} = \mathcal{B}_{\ell_\infty,+}$ is almost the same as the signed antisparse case in Section 5.4.4. The only difference is that, in the nonnegative antisparse case, the projection to the source domain $\mathcal{P} = \mathcal{B}_{\ell_\infty}$ is achieved by applying an elementwise nonnegative clipping function to individual outputs. Hence, we can express the neural network dynamics for the nonnegative antisparse case as follows:

$$\begin{aligned} \mathbf{e}(k;\nu) &= \mathbf{y}(k;\nu) - \mathbf{W}(k)\mathbf{x}(k), \\ \nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) &= \gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k;\nu) - \gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)\mathbf{e}(k;\nu), \\ \mathbf{y}(k;\nu+1) &= \sigma_+\left(\mathbf{y}(k;\nu) + \eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu))\right), \end{aligned}$$

where the nonnegative clipping function is defined as

$$\sigma_+(\mathbf{y})_i = \begin{cases} 0 & y_i \leq 0, \\ y_i & 0 \leq y_i \leq 1, \\ 1 & y_i \geq 1. \end{cases}$$

Figure 5.4 depicts the corresponding neural network realization.

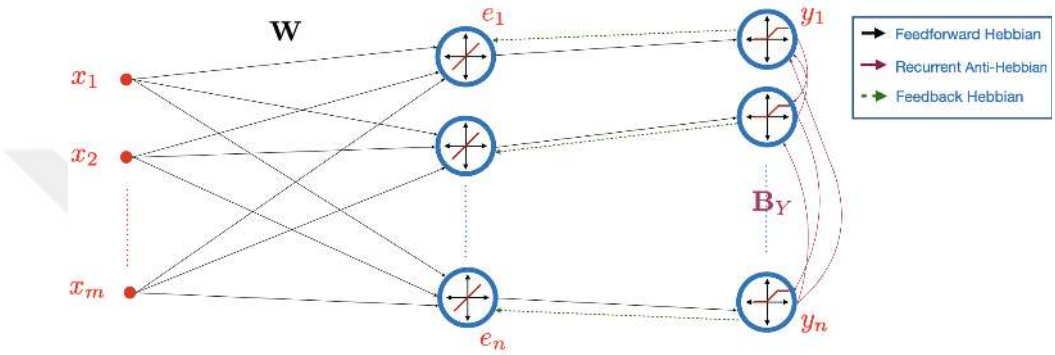


Figure 5.4: Two-Layer nonnegative antisparse CorInfoMax Network. x_i's represent the inputs (mixtures) and y_i's represent the output (separator) signals. $\mathbf{W}$ denotes the feedforward weights. e_i's represent the errors between the transformed inputs and outputs. $\mathbf{B_Y}$ represents the lateral weights at the output, which is the inverse of the output autocorrelation matrix. The output nonlinearities are nonnegative clipping functions.

5.4.6 Description of the Network Dynamics for Nonnegative Sparse Sources

For the nonnegative sparse CorInfoMax network, the dynamics of the output and inhibitory neurons can be stated as follows, with the only change compared to its sparse counterpart in Section 5.4.2 being the replacement of the soft-thresholding

activation functions at the output layer with rectified linear units:

$$\begin{aligned}
\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) &= \gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)\mathbf{y}(k;\nu) - \gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)\mathbf{e}(k;\nu),\\
\mathbf{y}(k;\nu+1) &= \mathrm{ReLU}\left(\mathbf{y}(k;\nu) + \eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) - \lambda(k;\nu)\right),\\
\nabla_{\lambda(k)}\mathcal{L}(\mathbf{y}(k;\nu)) &= 1 - \left(\sum_{i=1}^{n} y_i(k;\nu+1)\right),\\
\lambda(k;\nu+1) &= \mathrm{ReLU}\left(\lambda(k;\nu) - \eta_{\lambda}(\nu)\nabla_{\lambda(k)}\mathcal{L}(\mathbf{y}(k;\nu))\right).
\end{aligned}$$

The corresponding network structure is demonstrated in Figure 5.5.

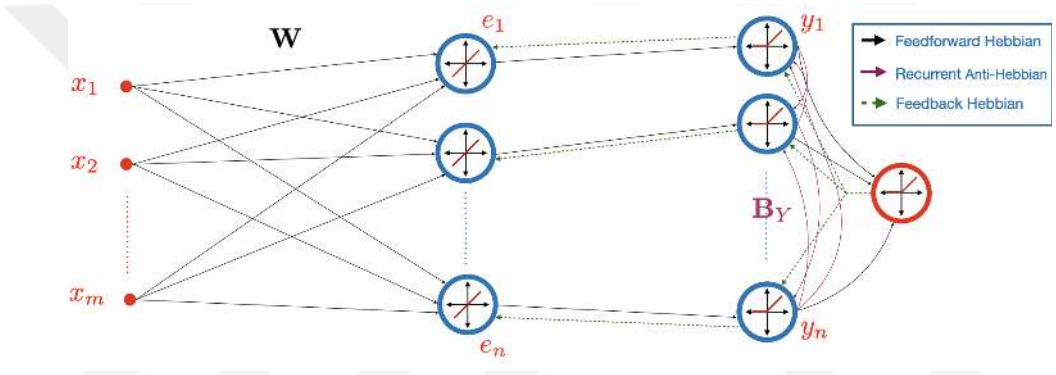


Figure 5.5: Three-Layer nonnegative sparse CorInfoMax Network. x_i's represent the inputs (mixtures) and y_i's represent the output (separator) signals. $\mathbf{W}$ denotes the feedforward weights. e_i's represent the errors between the transformed inputs and outputs. $\mathbf{B_Y}$ represents the lateral weights at the output, which is the inverse of the output autocorrelation matrix. The output nonlinearities are ReLU functions. The leftmost interneuron enforces sparsity constraints on the outputs.

5.4.7 Description of the Network Dynamics for Unit Simplex Sources

In the BSS setting for the unit simplex set Δ, which is commonly employed in nonnegative matrix factorization, we consider the following optimization problem:

$$\underset{\mathbf{y}(k)\in\mathbb{R}^n}{\text{maximize}} \quad \mathcal{J}(\mathbf{y}(k)) \tag{5.19a}$$

$$\text{subject to} \quad ||\mathbf{y}(k)||_1 = 1, \quad \mathbf{y}(k) \succcurlyeq 0. \tag{5.19b}$$

The Lagrangian-based Min-Max problem corresponding to this optimization problem can be stated as follows:

$$\underset{\lambda}{\text{minimize}} \; \underset{\mathbf{y}(k)\in\mathbb{R}^n}{\text{maximize}} \quad \overbrace{\mathcal{J}(\mathbf{y}(k)) - \lambda(k)(\|\mathbf{y}(k)\|_1 - 1)}^{\mathcal{L}(\mathbf{y}(k),\lambda(k))}.$$

In this context, unlike the optimization problem defined in (5.5), we do not impose the nonnegativity constraint on the Lagrangian variable λ due to the equality constraint in (5.19b). Consequently, we can express the network dynamics for a simplex source as follows:

$$\begin{aligned}
\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) &= \gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)\mathbf{y}(k;\nu) - \gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)\mathbf{e}(k;\nu),\\
\mathbf{y}(k;\nu+1) &= \text{ReLU}\left(\mathbf{y}(k;\nu) + \eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) - \lambda(k;\nu)\right),\\
\nabla_{\lambda(k)}\mathcal{L}(\mathbf{y}(k;\nu)) &= 1 - \left(\sum_{i=1}^{n} y_i(k;\nu+1)\right),\\
\lambda(k;\nu+1) &= \lambda(k;\nu) - \eta_{\lambda}(\nu)\nabla_{\lambda(k)}\mathcal{L}(\mathbf{y}(k;\nu)).
\end{aligned}$$

Figure 5.6 illustrates the network structure for simplex sources, which is identical to the nonnegative sparse CorInfoMax network except for the substitution of the ReLU activation function for the inhibitory neuron with a linear activation function.

5.4.8 Description of the Network Dynamics for Feature Based Specified Polytopes

In this section, we focus on the source separation setting where the source samples originate from a polytope represented in the form of (4.1). We extend the derivation presented in [Bozkurt et al., 2022] (refer to Section 4.7 in this thesis) to obtain a neural network solution for the BSS problem applicable to any identifiable polytope that can be expressed in the form of (4.1). Accordingly, we address the following optimization problem:

$$\underset{\mathbf{y}(k)\,\in\,\mathbb{R}^n}{\text{maximize}} \qquad \mathcal{J}(\mathbf{y}(k)) \tag{5.20a}$$

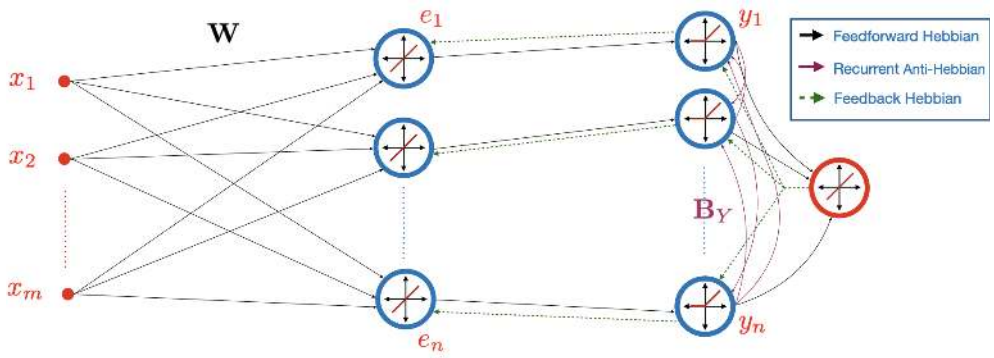


Figure 5.6: Three-Layer CorInfoMax Network for unit simplex sources. x_i's represent the inputs (mixtures) and y_i's represent the output (separator) signals. $\mathbf{W}$ denotes the feedforward weights. e_i's represent the errors between the transformed inputs and outputs. $\mathbf{B_Y}$ represents the lateral weights at the output, which is the inverse of the output autocorrelation matrix. The output nonlinearities are ReLU functions. The leftmost interneuron enforces sparsity constraints on the outputs through inhibition.

$$\text{subject to} \qquad -\mathbf{1} \preccurlyeq \mathbf{y}(k)_{\mathcal{I}_s} \preccurlyeq \mathbf{1}, \tag{5.20b}$$

$$\mathbf{0} \preccurlyeq \mathbf{y}(k)_{\mathcal{I}_+} \preccurlyeq \mathbf{1}, \tag{5.20c}$$

$$||\mathbf{y}(k)_{\mathcal{J}_l}||_1 \leq 1 \quad \forall l = 1, \ldots, L. \tag{5.20d}$$

The corresponding online Lagrangian Min-Max optimization setting can be written as follows:

$$\underset{\lambda_l(k)\geq 0}{\text{minimize}} \quad \underset{\substack{\mathbf{y}(k)\in\mathbb{R}^n \\ -\mathbf{1}\preccurlyeq \mathbf{y}(k)_{\mathcal{I}_s}\preccurlyeq \mathbf{1} \\ \mathbf{0}\preccurlyeq \mathbf{y}(k)_{\mathcal{I}_+}\preccurlyeq \mathbf{1}}}{\text{maximize}} \overbrace{\mathcal{J}(\mathbf{y}(k)) - \sum_{l=1}^{L}\lambda_l(k)(\|\mathbf{y}(k)_{\mathcal{J}_l}\|_1 - 1)}^{\mathcal{L}(\mathbf{y}(k),\lambda_1(k),\ldots,\lambda_L(k))}.$$

We can write the proximal operator corresponding to the Lagrangian term as

$$\text{prox}_{\lambda}(\mathbf{y}) = \underset{\boldsymbol{q}\text{ s.t. }\boldsymbol{q}_{\mathcal{I}_+}\succcurlyeq\mathbf{0}}{\text{argmin}} \left(\frac{1}{2}\|\mathbf{y}-\boldsymbol{q}\|_2^2 + \sum_{l=1}^{L}\lambda_l\|\boldsymbol{q}_{\mathcal{J}_l}\|_1\right). \tag{5.21}$$

Let $\boldsymbol{q}^*$ denote the output of the proximal operator defined in (5.21). By considering the first-order optimality condition, we can obtain the following expression:

- If $j \notin \mathcal{I}_+$, then $q_j^* - y_j + \sum_{\substack{l \in \mathcal{J}_l \\ \text{s.t. } j \in \mathcal{J}_l}} \lambda_l sign(y_j) = 0.$
  Therefore, $q_j^* = y_j - \sum_{\substack{l \in \mathcal{J}_l \\ \text{s.t. } j \in \mathcal{J}_l}} \lambda_l sign(y_j).$
- If $j \in \mathcal{I}_+$, then $q_j^* = y_j - \sum_{\substack{l \in \mathcal{J}_l \\ \text{s.t. } j \in \mathcal{J}_l}} \lambda_l.$

Consequently, by defining $\mathcal{I}_a = (\cap_l \mathcal{J}_l)^{\complement}$ as the set of dimension indices that do not appear in the sparsity constraints, we can express the corresponding output dynamics as follows:

$$\begin{aligned}
\nabla_{\mathbf{y}(k)} \mathcal{J}(\mathbf{y}(k;\nu)) &= \gamma_{\mathbf{y}} \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)\mathbf{y}(k;\nu) - \gamma_{\mathbf{e}} \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)\mathbf{e}(k;\nu), \\
\bar{\mathbf{y}}(k;\nu+1) &= \mathbf{y}(k;\nu) + \eta_{\mathbf{y}}(\nu) \nabla_{\mathbf{y}(k)} \mathcal{J}(\mathbf{y}(k;\nu)) \qquad (5.22) \\
y_j(k;\nu+1) &= ST_{\alpha_j(k,\nu)}\left(\bar{y}_j(k;\nu+1)\right) \qquad (5.23) \\
\text{where } \alpha_j(k;\nu) &= \sum_{\substack{l \in \mathcal{J}_l \\ \text{s.t. } j \in \mathcal{J}_l}} \lambda_l(k;\nu) \quad \forall j \in \mathcal{I}_s \cap \mathcal{I}_a^{\complement} \\
y_j(k;\nu+1) &= \text{ReLU}\left(\bar{y}_j(k;\nu+1) - \sum_{\substack{l \in \mathcal{J}_l \\ \text{s.t. } j \in \mathcal{J}_l}} \lambda_l(k;\nu)\right) \quad \forall j \in \mathcal{I}_+ \cap \mathcal{I}_a^{\complement} \\
y_j(k;\nu+1) &= \sigma_1(\bar{y}_j(k;\nu)) \quad \forall j \in \mathcal{I}_s \cap \mathcal{I}_a, \\
y_j(k;\nu+1) &= \sigma_+(\bar{y}_j(k;\nu)) \quad \forall j \in \mathcal{I}_+ \cap \mathcal{I}_a
\end{aligned}$$

The update dynamics for inhibitory neurons corresponding to Lagrangian variables $\lambda_1, \ldots, \lambda_L$ can be obtained using the derivative of $\mathcal{L}(\mathbf{y}(k;\nu), \lambda_1(k;\nu), \ldots, \lambda_L(k;\nu))$ as

$$\begin{aligned}
\frac{d\mathcal{L}(\mathbf{y}(k), \lambda_1(k), \ldots, \lambda_L(k))}{d\lambda_l(k)}\Big|_{\lambda_l(k;\nu)} &= 1 - \|[\mathbf{y}(k;\nu+1)]_{\mathcal{J}_l}\|_1 \quad \forall l, \\
\bar{\lambda}_l(k;\nu+1) &= \lambda_l(k;\nu) \\
&\quad - \eta_{\lambda_l}(\nu) \frac{d\mathcal{L}(\mathbf{y}(k), \lambda_1(k), \ldots, \lambda_L(k))}{d\lambda_l(k)}\Big|_{\lambda_l(k;\nu)}, \\
\lambda_l(k,\nu+1) &= \text{ReLU}\left(\bar{\lambda}_l(k;\nu+1)\right).
\end{aligned}$$

In Section 5.5.7 for a numerical experiment, we present an example setting where the underlying domain is defined as

$$\mathcal{P}_{ex} = \left\{ \mathbf{s} \in \mathbb{R}^5 \;\middle|\; \begin{array}{c} s_1, s_2, s_4 \in [-1,1], s_3, s_5 \in [0,1], \\ \left\| \begin{bmatrix} s_1 \\ s_2 \\ s_5 \end{bmatrix} \right\|_1 \leq 1, \left\| \begin{bmatrix} s_2 \\ s_3 \\ s_4 \end{bmatrix} \right\|_1 \leq 1 \end{array} \right\}.$$

Based on our derivations, we can summarize the neural dynamics for this particular example:

$$\begin{aligned}
y_1(k;\nu+1) &= ST_{\lambda_1(k,\nu)}\left(\bar{y}_1(k;\nu+1)\right), \\
y_2(k;\nu+1) &= ST_{\lambda_1(k,\nu)+\lambda_2(k,\nu)}\left(\bar{y}_2(k;\nu+1)\right), \\
y_3(k;\nu+1) &= \mathrm{ReLU}\big(\bar{y}_3(k;\nu+1) - \lambda_2(k;\nu)\big), \\
y_4(k;\nu+1) &= ST_{\lambda_2(k,\nu)}\left(\bar{y}_4(k;\nu+1)\right), \\
y_5(k;\nu+1) &= \mathrm{ReLU}\big(\bar{y}_5(k;\nu+1) - \lambda_1(k;\nu)\big), \\
\lambda_1(k;\nu+1) &= \lambda_1(k;\nu) - \eta_{\lambda_1}(\nu)\big(1 - |y_1(k;\nu+1)| - |y_2(k;\nu+1)| - y_5(k;\nu+1)\big), \\
\lambda_2(k;\nu+1) &= \lambda_2(k;\nu) - \eta_{\lambda_1}(\nu)\big(1 - |y_2(k;\nu+1)| - y_3(k;\nu+1) - |y_4(k;\nu+1)|\big),
\end{aligned}$$

where $\bar{\mathbf{y}}(k;\nu)$ is defined as in (5.22).

5.5 Numerical Experiments

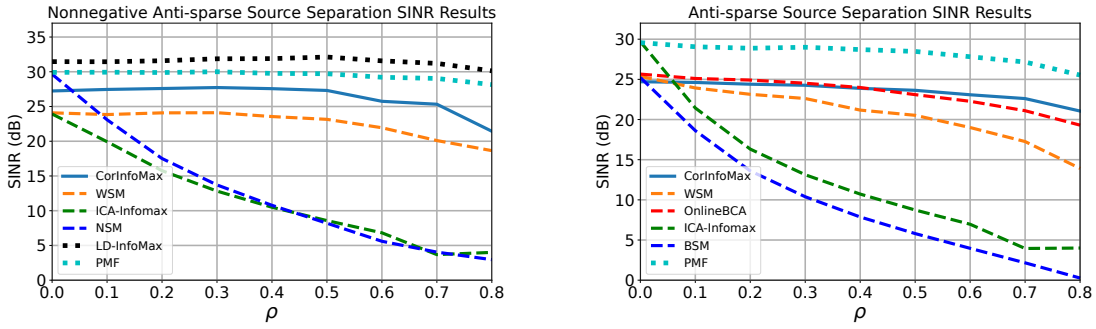
In this section, we illustrate various domain selections for sources and compare the proposed CorInfoMax framework with existing batch algorithms and online biologically plausible neural network approaches. We showcase the capability of the proposed framework in correlated source separation using both synthetic and natural sources. Python scripts and notebooks for each experiment, allowing replication of the reported results, are provided on Github [1].

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

5.5.1 Synthetically Correlated Source Separation with Antisparse Sources

In order to demonstrate the correlated source separation capabilities of the online CorInfoMax framework for both nonnegative and signed antisparse sources, namely $\mathbf{s}(i) \in \mathcal{B}_{\ell_{\infty,+}}\ \forall i$ and $\mathbf{s}(i) \in \mathcal{B}_{\ell_\infty}\ \forall i$ respectively, we conducted the BSS experiments involving $n = 5$ sources and $m = 10$ mixtures. The 5-dimensional sources were generated using the Copula-T distribution with 4 degrees of freedom. To control the correlation level among the sources, we adjusted a Toeplitz distribution parameter matrix, with its first row being $\begin{bmatrix} 1 & \rho & \rho & \rho & \rho \end{bmatrix}$ where $\rho \in [0, 0.8]$. Each source was realized by generating $N = 5 \times 10^5$ samples, and the mixing process was performed using a random matrix $\mathbf{A} \in \mathbb{R}^{10\times 5}$ with i.i.d. standard normal entries. Additionally, white Gaussian noise (WGN) with an SNR level of 30dB was added to corrupt the mixture signals. For these experiments, we utilized the antisparse CorInfoMax network described in Section 5.4.4 and the nonnegative CorInfoMax network presented in Appendix 5.4.5. In order to provide a comparative analysis, we also included biologically plausible algorithms, namely online BCA [Simsek and Erdogan, 2019], WSM [Bozkurt et al., 2022], NSM [Pehlevan et al., 2017b], BSM [Erdogan and Pehlevan, 2020], as well as batch algorithms, including ICA-Infomax [Bell and Sejnowski, 1995], LD-InfoMax [Erdogan, 2022], and PMF [Tatli and Erdogan, 2021b].

In Figure 5.7, we present the curves depicting the signal-to-interference-plus-noise ratio (SINR) as a function of the correlation level ρ for various algorithms used in the experiments on nonnegative antisparse and antisparse source separation. Our observations indicate that the proposed CorInfoMax approach demonstrates commendable performance with relatively high SINR results, even as the correlation level ρ increases, for both nonnegative antisparse and antisparse source separation experiments. While the WSM curve exhibits a similar trend, it lags behind the performance of CorInfoMax. As anticipated, the batch learning algorithms LD-InfoMax and PMF consistently deliver superior results, benefiting from their *batch* learning settings. Conversely, the performance of NSM, BSM, and ICA-Infomax deteriorates with increasing source correlation, as these methods assume uncorrelated or independent sources.



(a) (b)

Figure 5.7: The averaged SINR performances of CorInfoMax (ours), LD-InfoMax, PMF, ICA-InfoMax, NSM, and BSM, obtained from 100 realizations, are plotted on the y-axis against the correlation factor ρ on the x-axis. The SINR vs. ρ curves are depicted for two different source domain configurations: (a) nonnegative antisparse ($\mathcal{B}_{\ell_\infty,+}$), (b) antisparse ($\mathcal{B}_{\ell_\infty}$) source domains.

5.5.2 Video Separation

To provide a concrete example and showcase a real-world scenario with naturally correlated sources, we conducted a video separation experiment. In this setup, we mixed three videos, each of duration 10 seconds, to generate five mixture videos. The source videos used in this experiment were obtained from the website https://www.pexels.com/, which allows free download and usage of the videos. The three source videos exhibited different levels of correlation, as measured by average and maximum Pearson correlation coefficients. The average (across frames) correlation coefficients between sources were $\rho_{12}^{\text{average}} = -0.1597$, $\rho_{13}^{\text{average}} = -0.1549$, and $\rho_{23}^{\text{average}} = 0.3811$. The corresponding maximum (across frames) correlation coefficients were $\rho_{12}^{\text{maximum}} = 0.3139$, $\rho_{13}^{\text{maximum}} = 0.2587$, and $\rho_{23}^{\text{maximum}} = 0.5173$. For the mixing process, the videos were linearly combined using a randomly selected

nonnegative matrix of size 5×3 as follows:

$$\mathbf{A} = \begin{bmatrix} 1.198 & 1.020 & 1.273 \\ 0.367 & 1.364 & 0.901 \\ 0.100 & 0.869 & 0.627 \\ 1.257 & 0.859 & 0.015 \\ 0.957 & 0.789 & 0.592 \end{bmatrix},$$

to generate 5 mixture videos. This choice ensured nonnegative mixtures, allowing them to be displayed as proper images without any loss of generality. In Figures 5.8a and 5.8b, we present the final RGB frames of the original videos and the mixture videos, respectively.

(a)

(b)

(c)

Figure 5.8: Video separation example: the final frames of (a) sources, (b) mixtures, and (c) the outputs of the nonnegative antisparse CorInfoMax network.

Given that the image pixels are in the range of $[0, 1]$, we employed the nonnegative antisparse CorInfoMax network depicted in Figure 5.4 for separating the original videos. To train the network, we followed the procedure outlined below.

- During each iteration of the algorithm, we adopt a random pixel location and extract the corresponding pixels from one of the color channels of all mixture videos. These pixels are then combined to create a mixture vector of size 5×1.

- We perform 20 iterations of the algorithm per frame by sampling 20 mixture vectors from each frame. (e.g., first 20 random pixels from the first frame are fed consecutively for 20 algorithm iterations, later on 20 random pixels from the second frames are fed consecutively, and so on).

The demonstration video for this experiment comprises three rows of frames. The first row displays the source frames, representing the original videos. The second row consists of three of the five mixture frames, showcasing a subset of the mixed videos. Finally, the last row showcases the network outputs for these mixture frames, indicating the separated videos. You can access the demo video at the following link: (https://figshare.com/s/a3fb926f273235068053). By utilizing the separator matrix of the CorInfoMax network, which serves as an estimate for the left inverse of $\mathbf{A}$, to predict the original videos after training, we achieved peak signal-to-noise ratio (PSNR) values of 35.60dB, 48.07dB, and 44.58dB for the respective videos. These PSNR values are calculated as the average levels across the frames. Figure 5.8c showcases the final output frames of the CorInfoMax network, providing a visual representation of the separated videos.

5.5.3 Image Separation

In the context of blind separation of correlated images from their linear mixtures, we present an example involving three RGB natural scenes. These images have dimensions of 454×605 and are displayed in Figure 5.9a. The Pearson correlation coefficients between these sources are $\rho_{12} = 0.076$, $\rho_{13} = 0.262$, and $\rho_{23} = 0.240$, respectively. To generate the mixture images, the image sources are linearly combined

using a random matrix $\mathbf{A} \in \mathbb{R}^{5\times 3}$ with entries drawn from an i.i.d. standard normal distribution. Additionally, the mixture images are corrupted with white Gaussian noise (WGN) corresponding to a signal-to-noise ratio (SNR) of 40dB. The resulting mixture images are displayed in Figure 5.9b. The mixing matrix for this specific example is as follows:

$$\mathbf{A} = \begin{bmatrix} -0.363 & 0.650 & 1.757 \\ 1.100 & 1.568 & 1.487 \\ -1.266 & 0.032 & -0.417 \\ -0.822 & 0.643 & 1.260 \\ -0.023 & -0.752 & 0.661 \end{bmatrix}.$$

In our experiment, we evaluate the performance of the proposed nonnegative antisparse CorInfoMax network as well as the biologically plausible NSM and WSM networks, along with the batch ICA-InfoMax and LD-InfoMax frameworks, for the purpose of separating the original sources. The outputs obtained from these different methods are illustrated in Figures 5.9c to 5.9g. These figures showcase the separated images produced by each of the respective algorithms. We observe that the residual interference effects in the output images of the ICA-InfoMax algorithm are significantly noticeable, resulting in PSNR values of 18.56dB, 20.52dB, and 21.06dB, respectively. Although the outputs of the NSM algorithm exhibit improved visual quality, some interference effects remain perceptible, yielding PSNR values of 25.30dB, 26.49dB, and 26.45dB. In contrast, the WSM algorithm produces outputs with minimal visible interference, resulting in higher PSNR values of 27.99dB, 29.71dB, and 31.92dB. Among the algorithms considered, the batch LD-InfoMax algorithm demonstrates the best PSNR performance, achieving values of 33.60dB, 31.99dB, and 33.62dB. Notably, our proposed CorInfoMax method outperforms other biologically plausible neural networks and the batch ICA-InfoMax algorithm, while achieving comparable performance to the batch LD-InfoMax algorithm. Specifically, the output PSNR values of our proposed method are 32.45dB, 29.72dB, and 32.37dB.

It is worth mentioning that the performance of the ICA and NSM algorithms is

significantly influenced by the assumption of independent and uncorrelated sources, respectively. Consequently, the correlation level of the sources has a notable impact on their respective performances. Typically, the LD-InfoMax algorithm, benefiting from its batch nature and ability to separate dependent sources, achieves the best performance. Notably, the biologically plausible WSM network demonstrates the capability to separate correlated sources, while the CorInfoMax algorithm outperforms it both visually and in terms of the PSNR metric. For the specific hyperparameters used in this experiment for CorInfoMax, please refer to Appendix B.3.

(a)

(b)

(c)

Figure 5.9: Image separation example: (a) Original RGB images, (b) mixture RGB images, (c) ICA outputs.

(d)

(e)

(f)

(g)

Figure 5.9: (d) NSM outputs, (e) WSM outputs, (f) LD-InfoMax Outputs, (g) CorInfoMax (ours) Outputs.

5.5.4 Sparse Source Separation

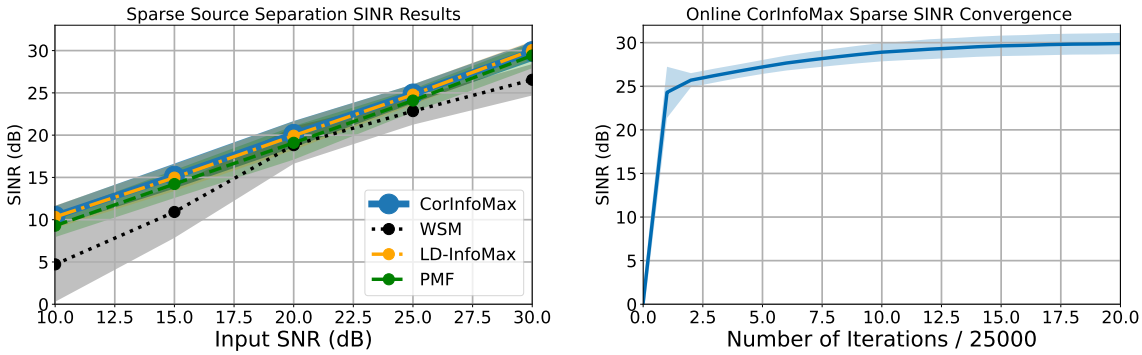
In this section, we present the blind separation of sparse sources, specifically where each source $\mathbf{s}(i)$ belongs to the $\mathcal{B}_{\ell_1}$ for all i. Our experiment involves five sources

($n = 5$) and ten mixtures ($m = 10$). In each realization of the experiment, we generate 5×10^5 samples for each source. We examine two different factors in our experiments. Firstly, we analyze the output SINR performance as a function of mixture SNR levels. Secondly, we investigate the output SINR performance for different distribution selections of the entries in the mixing matrix. This analysis helps us understand how different distributions affect the separation performance in terms of SINR.

In the first scenario involving different mixture SNR levels, we mix the sources using a random matrix $\mathbf{A} \in \mathbb{R}^{10\times5}$, where the entries are drawn from an i.i.d. standard normal distribution. The resulting mixtures are then corrupted by white Gaussian noise (WGN) with an SNR of 30 dB. In our evaluation, we compare our proposed approach, CorInfoMax, with the WSM algorithm (WSMDetMax), LD-InfoMax, and PMF algorithms. The results are visualized in Figure 5.10. To train the batch LD-InfoMax and PMF algorithms, we utilize a portion of the mixtures. Figure 5.10h presents the SINR performances of these algorithms for different input noise levels. It is worth noting that the SINR results of CorInfoMax, LD-InfoMax, and PMF are consistently close to each other, nearly eqaul to the input SNR. Figure 5.10i depicts the SINR convergence of the sparse CorInfoMax network for the mixture SNR level of 30 dB, plotted against the number of update iterations. From this figure, we can observe that the proposed CorInfoMax network exhibits robust and smooth convergence behavior.

In the second experimental setting, we investigate the impact of different distribution choices for generating the random mixing matrix. Figure 5.11 displays box plots of the SINR results for CorInfoMax, LD-InfoMax, and PMF algorithms, considering various distribution selections for the mixing matrix generation. The distributions considered are $\mathcal{N}(0, 1)$ (normal distribution), $\mathcal{U}[-1, 1]$ (uniform distribution between -1 and 1), $\mathcal{U}[-2, 2]$ (uniform distribution between -2 and 2), and $L(0, 1)$ (Laplace distribution). From the figure, it is evident that the performance of CorInfoMax remains robust across different distribution selections for the unknown mixing matrix. Additionally, its performance is comparable to that of the batch algorithms LD-InfoMax and PMF. This observation highlights the stability

and effectiveness of CorInfoMax in achieving reliable source separation outcomes.



(h) (i)

Figure 5.10: SINR performances of CorInfoMax (ours), LD-InfoMax, PMF, and WSM (averaged over 50 realizations) for sparse sources: (a) output SINR results (vertical axis) with respect to input SNR levels (horizontal axis), (b) SINR (vertical axis) convergence plot as a function of iterations (horizontal axis) of sparse CorInfoMax for the 30 dB SNR level (mean solid line and standard deviation envelopes).

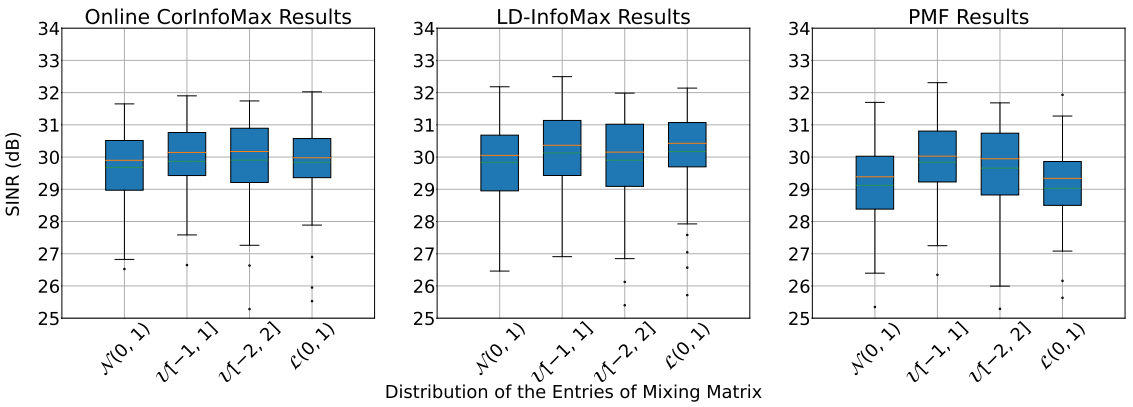
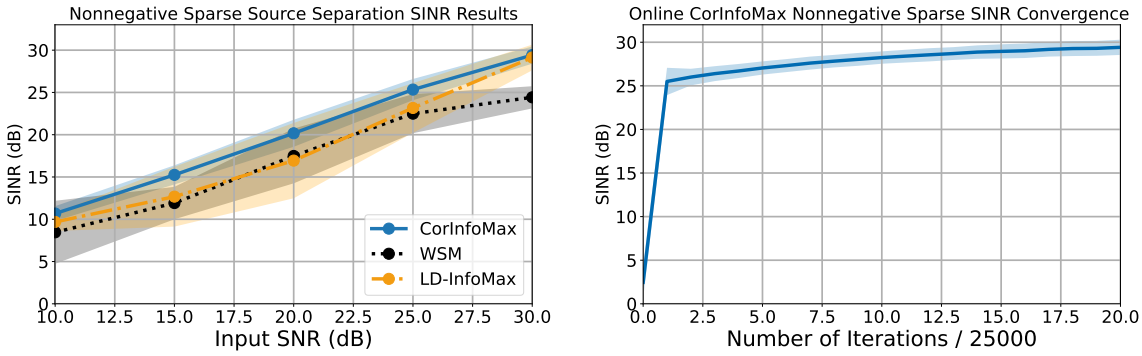


Figure 5.11: SINR performances of CorInfoMax (ours), LD-InfoMax, and PMF with different distribution selections for the mixing matrix entries and for sparse sources. The horizontal axis represents different distribution choices, and the vertical axis represents the SINR levels.

5.5.5 Nonnegative Sparse Source Separation

In order to investigate the nonnegative sparse source separation, we conducted experiments replicating the first experimental setup outlined in Section 5.5.4. The objective was to evaluate the SINR performance of the nonnegative sparse CorInfoMax network for varying levels of mixture SNR, in comparison with the batch LD-InfoMax algorithm and the biologically plausible WSM neural network. In these experiments, we considered a scenario with five uniform sources ($n = 5$) belonging to $\mathcal{B}_{\ell_1,+}$. These sources were randomly mixed to generate ten mixtures ($m = 10$). Figure 5.12a presents the averaged output SINR performances of each algorithm, along with a standard deviation envelope, for different input noise levels. Furthermore, Figure 5.12b showcases the SINR convergence behavior of the nonnegative sparse CorInfoMax network as a function of update iterations for 30dB mixture SNR level. Indeed, based on the presented figures, it is notable that CorInfoMax consistently outperforms the biologically plausible neural network WSM for all input SNR levels, and the convergence of CorInfoMax is noticeably stable

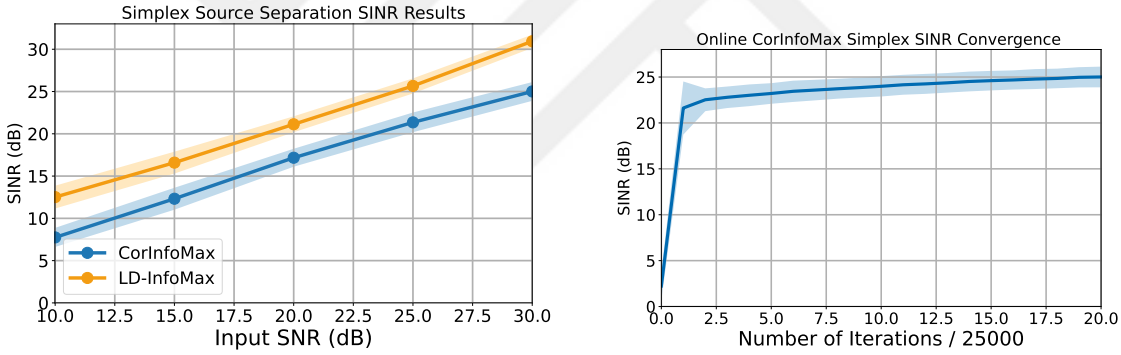


(a) (b)

Figure 5.12: SINR performances of CorInfoMax (ours), LD-InfoMax, and WSM for nonnegative sparse sources: (a) the output SINR (vertical axis) results with respect to the input SNR levels (horizontal axis), (b) the SINR (vertical axis) convergence plot as a function of iterations (horizontal axis) of nonnegative sparse CorInfoMax for the 30dB SNR level (mean solid line and standard deviation envelopes).

5.5.6 Simplex Source Separation

We replicate both experimental configurations discussed in Section 5.5.4 to perform blind separation of simplex sources utilizing the CorInfoMax network illustrated in Figure 5.6. The output SINR results for different mixture SNR levels are depicted in Figure 5.13a, showcasing the performance of both the online CorInfoMax and batch LD-InfoMax approaches. While the simplex CorInfoMax may not exhibit the same level of success as other examples (e.g., sparse CorInfoMax), it still demonstrates satisfactory source separation capability in terms of its proximity to the performance of batch algorithms. Moreover, similar to the sparse network examples, the SINR convergence of the simplex CorInfoMax is rapid and smooth, as demonstrated in Figure 5.13b.



(a) (b)

Figure 5.13: SINR performances of CorInfoMax (ours) and LD-InfoMax (averaged over 50 realizations) for unit simplex sources: (a) the output SINR (vertical axis) results with respect to the input SNR levels (horizontal axis), (b) SINR (vertical axis) convergence plot as a function of iterations (horizontal axis) of simplex-CorInfoMax for 30dB SNR level (mean solid line and standard deviation envelopes).

The box plots presented in Figure 5.14 display the SINR performances of the CorInfoMax and LD-InfoMax approaches, considering various distribution selections for generating the random mixing matrix. From this figure, it can be inferred that the simplex CorInfoMax network consistently maintains its performance across different

distributions used for the entries of the mixing matrix.

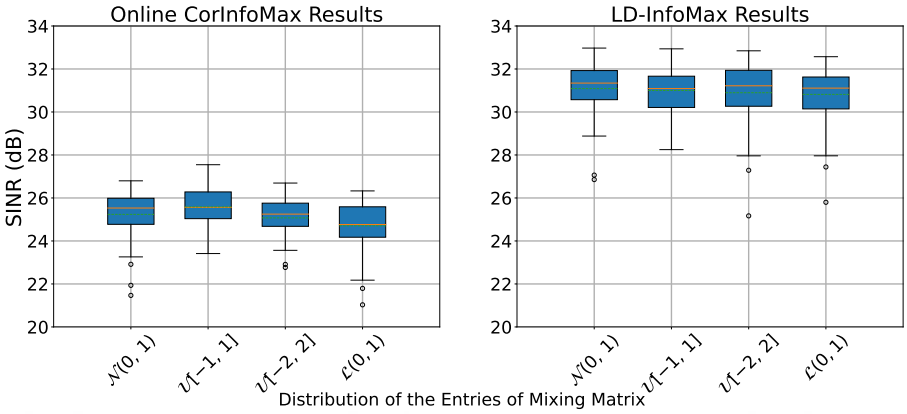


Figure 5.14: SINR performances of CorInfoMax (ours) and LD-InfoMax with different distribution selections for the mixing matrix entries (averaged over 50 realizations) and unit simplex sources. The horizontal axis represents different distribution choices, and the vertical axis represents the SINR levels.

5.5.7 Source Separation for Polytopes with Mixed Latent Attributes

In this section, we showcase the source separation capability of CorInfoMax on an identifiable polytope with mixed features. This specific polytope belongs to the category of "feature-based" polytopes, as described in Equation (4.1). Our attention is specifically directed towards the following polytope

$$\mathcal{P}_{ex} = \left\{ \mathbf{s} \in \mathbb{R}^5 \;\middle|\; \begin{array}{c} s_1, s_2, s_4 \in [-1,1], s_3, s_5 \in [0,1], \\ \left\| \begin{bmatrix} s_1 \\ s_2 \\ s_5 \end{bmatrix} \right\|_1 \leq 1, \left\| \begin{bmatrix} s_2 \\ s_3 \\ s_4 \end{bmatrix} \right\|_1 \leq 1 \end{array} \right\}. \tag{5.24}$$

The identifiability property of the mentioned polytope is confirmed by employing the identifiable polytope characterization algorithm introduced in [Bozkurt and Erdogan, 2022] (Algorithm 1 in Section 3.4 of this thesis). We conduct experiments using both approaches discussed in Section 5.4.3 and Section 5.4.8. In the case of the feature-based polytope setting presented in Section 5.4.8, we pro-

vide an example by summarizing the output dynamics corresponding to $\mathcal{P}_{ex}$. This polytope can also be represented as the intersection of 10 half-spaces, denoted as $\mathcal{P}_{ex} = \{\mathbf{s} \in \mathbb{R}^n | \mathbf{A}_{\mathcal{P}}\mathbf{s} \preccurlyeq \mathbf{b}_{\mathcal{P}}\}$, where $\mathbf{A}_{\mathcal{P}} \in \mathbb{R}^{10\times 5}$ and $\mathbf{b}_{\mathcal{P}} \in \mathbb{R}^{10}$. Consequently, by utilizing $\mathbf{A}_{\mathcal{P}}$ and $\mathbf{b}_{\mathcal{P}}$, we can employ the neural network depicted in Figure 5.1a with 10 inhibitory neurons.

In this BSS setting, we uniformly generated the sources within the 5-dimensional polytope $\mathcal{P}_{ex}$, with a sample size of 5×10^5. The source vectors were mixed using a random matrix $\mathbf{A} \in \mathbb{R}^{10\times 5}$ with standard normal entries. Figure 5.15a and 5.15b display the SINR convergence of the CorInfoMax networks based on the feature-based polytope representation in (4.1) and the H-representation in (5.1), respectively, for a mixture SNR level of 30 dB. Similarly, Figure 5.15c and 5.15d illustrate their SINR convergence curves for a mixture SNR level of 40 dB. In Table 5.1, we provide a comparison between both approaches and the batch algorithms LD-InfoMax and PMF.

Table 5.1: Source separation averaged SINR results on $\mathcal{P}_{\text{ex}}$ for the CorInfoMax (ours), CorInfoMax Canonical (ours) LD-InfoMax, and PMF algorithms (averaged for 50 realizations).

Algorithm	**CorInfoMax**	**CorInfoMax Canonical**	**LD-InfoMax**	**PMF**
SINR (\w 30dB SNR)	26.55	24.85	30.28	27.68
SINR (\w 40dB SNR)	30.93	26.19	38.50	31.48

5.5.8 Sparse Dictionary Learning

We explore sparse dictionary learning for natural images, aiming to model receptive fields in the early stages of visual processing [Olshausen and Field, 1997]. This experiment is also connected to the numerical experiment presented in Section 4.8.6. In this experiment, we utilize 12×12 pre-whitened image patches obtained from the website http://www.rctn.org/bruno/sparsenet. as input to the sparse CorInfoMax

network. Prior to feeding them into the neural network illustrated in Figure 5.2, the image patches are vectorized into a shape of 144×1. Figure 5.16 depicts the dictionary learned by the sparse CorInfoMax network.

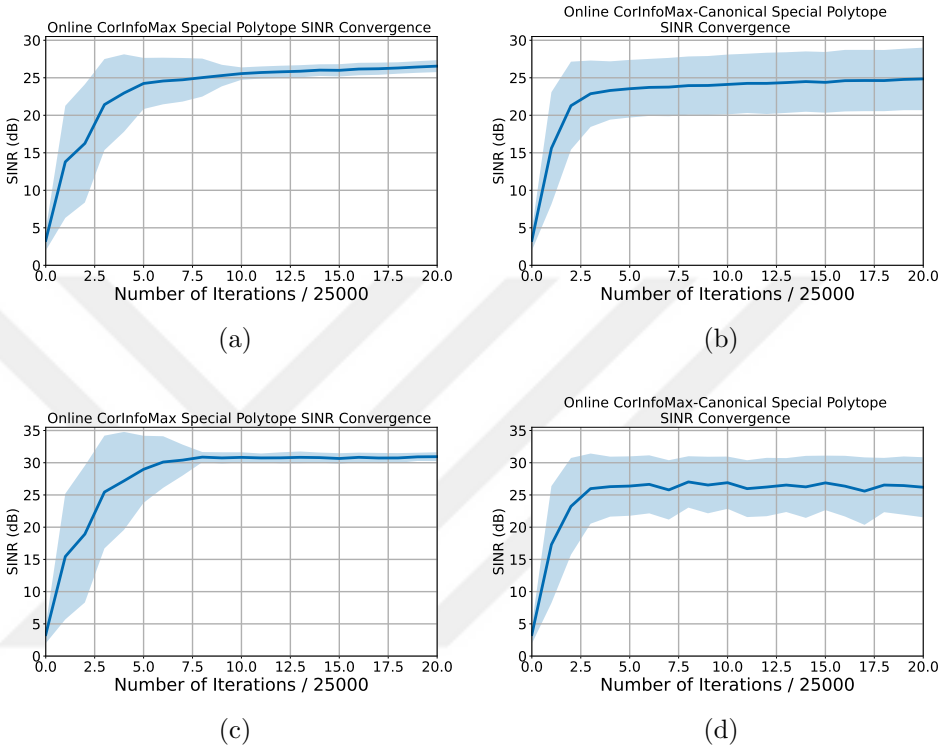


Figure 5.15: SINR performances of CorInfoMax networks on $\mathcal{P}_{\text{ex}}$ with mean-solid line with standard deviation envelope (averaged over 50 realization) for the polytope with mixed latent attributes: (a) SINR convergence curve for CorInfoMax feature-based polytope formulation with 30dB mixture SNR, (b) SINR convergence curve for CorInfoMax canonical formulation with 30dB mixture SNR, (c) SINR convergence curve for CorInfoMax feature-based polytope formulation with 40dB mixture SNR, (d) SINR convergence curve for CorInfoMax canonical formulation with 40dB SNR.

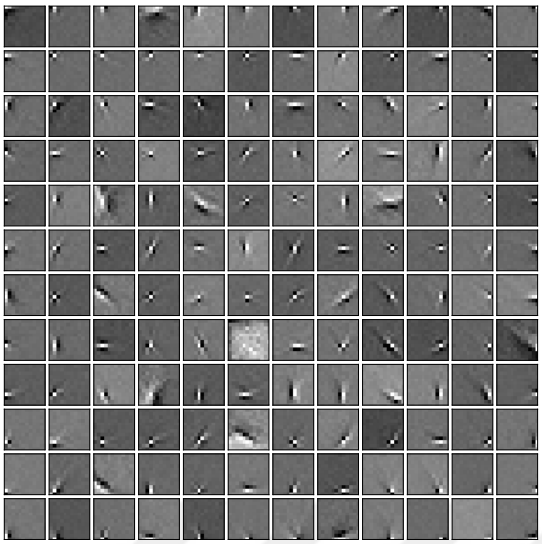


Figure 5.16: Sparse dictionary learned by sparse CorInfoMax from natural image patches.

5.5.9 Digital Communication Example: 4-PAM Modulation Scheme

One successful application of antisparse source modeling in solving BSS problems is in digital communication systems [Cruces, 2010, Erdogan, 2013]. In this section, we demonstrate the effectiveness of the proposed antisparse CorInfoMax network, depicted in Figure 5.3, for separating 4-pulse-amplitude-modulation (4-PAM) signals. These signals belong to the domain $\{-3, -1, 1, 3\}$ and have a uniform probability distribution. We consider a scenario where 5 digital communication sources, each employing 4-PAM modulation, transmit signals with a sample size of 10^5. These transmitted signals are then mixed through a Gaussian channel, simulating the received signals at base station antennas in a multipath propagation environment. Additionally, the mixtures are corrupted by WGN with an SNR level of 30 dB. The mixtures are provided as input to the antisparse CorInfoMax network. For 100 different realizations of the experimental setup, Figure 5.5.9 presents the SINR convergence as a function of update iterations. It is noteworthy that the proposed approach demonstrates rapid convergence, and in each realization of the experiments, a zero symbol error rate is achieved.

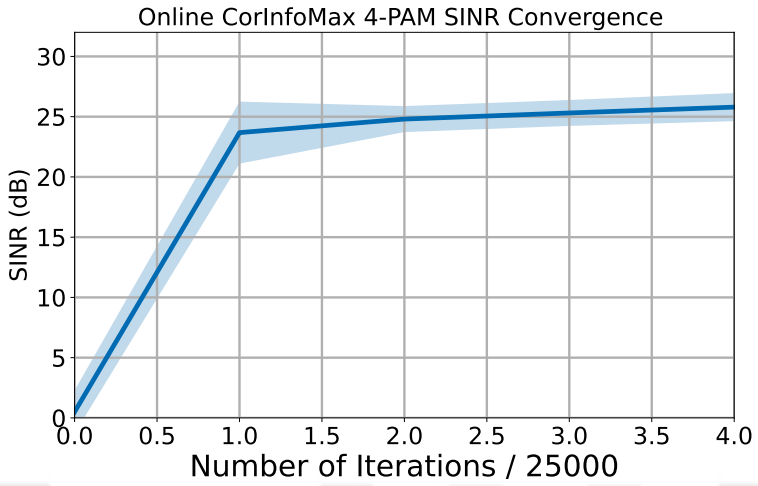


Figure 5.17: The SINR (vertical axis) convergence curve for the 4-PAM digital communication example as a function of iterations (horizontal axis) which is averaged over 100 realizations: mean-solid line with standard deviation envelope.

5.6 Conclusion

In this chapter, we present an information-theoretic framework, called CorInfoMax, for the generation of biologically plausible neural networks capable of separating both independent and correlated sources. The CorInfoMax framework offers the flexibility to be applied to a wide range of source domains, allowing for diverse source characterizations. Beyond its application in solving unsupervised linear inverse problems, CorInfoMax networks have the potential to generate structured embeddings from observations, depending on the chosen source domains. As a future extension, we explore representation frameworks that learn desirable source domain representations by adapting the output-interneuron connections illustrated in Figure 5.1a. Furthermore, the proposed unsupervised framework, along with its potential supervised extensions, holds promise for neuromorphic systems that rely on local learning rules. This framework aligns with the goal of developing neural networks that emulate the principles and constraints observed in biological systems.

One limitation of the proposed framework is the computational complexity asso-

ciated with simulating such networks on conventional computers. This is primarily due to the loop-based recurrent output computation. However, as discussed in Appendix B.6, the computational load of the neural networks generated by the proposed framework is comparable to that of existing biologically plausible BSS neural networks (see also Appendix A.5).

Chapter 6

CORRELATIVE INFORMATION MAXIMIZATION: A BIOLOGICALLY PLAUSIBLE APPROACH TO SUPERVISED DEEP NEURAL NETWORKS WITHOUT WEIGHT SYMMETRY

6.1 Abstract

The backpropagation algorithm has achieved impressive success in training large-scale artificial neural networks. The biological-plausibility of backpropagation is disputed, and it remains uncertain whether the brain employs similar supervised learning mechanisms. In this chapter, we propose correlative information maximization between layer activations as an alternative normative approach to explain signal propagation in biological neural networks, both in the forward and backward directions. This novel framework addresses several concerns regarding the biological plausibility of conventional artificial neural networks and the backpropagation algorithm. By optimizing the corresponding objective using coordinate descent, along with the mean square error loss function for fitting labeled supervision data, we establish a neural network structure that mimics a biologically realistic network of multi-compartment pyramidal neurons with dendritic processing and lateral inhibitory neurons. Moreover, our approach naturally resolves the weight symmetry problem that arises between the forward and backward signal propagation paths, a notable criticism of the conventional backpropagation algorithm. To achieve this, we leverage two alternative yet equivalent forms of the correlative mutual information objective. As a result, these alternatives inherently give rise to forward and backward prediction networks that do not suffer from weight symmetry issues, presenting a compelling solution to this long-standing challenge.

The contributions presented in this chapter has been published as a preprint arXiv paper [Bozkurt et al., 2023b]. At the time of submitting this thesis, the paper was undergoing the review process in an artificial intelligence and neuroscience venue.

6.2 Introduction

The question of how biological neural networks achieve supervised learning has remained an open problem for a considerable time. The backpropagation algorithm [Rumelhart et al., 1986] has demonstrated remarkable success in training large-scale artificial neural networks and possesses an intuitive structure. As a result, it has served as a basis for proposals on how biologically plausible neural networks could accomplish efficient credit assignment for supervised learning in deep neural architectures [Whittington and Bogacz, 2019]. Nevertheless, certain aspects of the backpropagation algorithm, in conjunction with the oversimplified nature of artificial neurons, have been perceived as obstacles to proposals that draw inspiration from it [Crick, 1989].

A fundamental criticism regarding the biological plausibility of the backpropagation algorithm pertains to the presence of a parallel backward path that propagates error signals from the output to the input layer, utilizing the same synaptic weights as the forward path [Rumelhart et al., 1986, Whittington and Bogacz, 2019, Grossberg, 1987a]. Experimental evidence suggests that such weight transport, also known as weight symmetry, is highly unlikely in biological systems [Crick, 1989, Grossberg, 1987a]. Nonetheless, certain biologically plausible frameworks exhibit this feature, as it is justified by the symmetric structure of the Hebbian updates employed in these frameworks [Whittington and Bogacz, 2019, Xie and Seung, 2003, Scellier and Bengio, 2017].

The limitations arising from the simplicity of artificial neurons have been alleviated by incorporating multi-compartment neuron models into networked architectures. These models attribute significant functions to dendritic processing in the credit assignment process [Larkum, 2013, Urbanczik and Senn, 2014, Sacramento et al., 2018, Golkar et al., 2022]. This fresh perspective has facilitated the creation

of neural networks that exhibit enhanced biological plausibility.

In this chapter, we propose the use of correlative information maximization (CorInfoMax) as an objective for biologically plausible neural network models. This approach aims to maximize the correlative information among consecutive layers, providing a promising avenue for enhancing the biological plausibility of such models. Our proposed framework offers

- a principled solution to the weight symmetry problem: our proposed information theoretic criterion aims to maximize the linear dependence between signals in adjacent layers, which naturally leads to the utilization of linear or affine transformations between them. An important characteristic of this approach is the utilization of two different expressions for the correlative mutual information (CMI), resulting in the potential development of forward and backward prediction networks with asymmetry. This asymmetry offers a natural solution to the weight transport problem. As a consequence, predictive coding emerges as an inherent solution to the maximization of correlative information, facilitating signal transmission in both forward and top-down directions through asymmetric connections. By incorporating set membership constraints on layer activations, such as non-negativity through activation nonlinearities and lateral inhibitions, the CorInfoMax principle promotes compression of information and sparse representations, enhancing information flow in both directions [Bozkurt et al., 2023a].

- a normative approach for deriving networks with multi-compartment neurons: The optimization of the CorInfoMax objective through gradient search naturally results in network models that utilize multi-compartment pyramidal neuron models, along with the inclusion of interneurons, as depicted in Figure 6.1.

As derived and extensively elaborated in Section 6.3, the resulting networks include lateral connections and autapses (auto-synapses) to increase the entropy of a layer, encouraging the utilization of all dimensions within the representation space of that layer. Concurrently, the asymmetric feedforward and feedback connections

function as forward and backward predictors, respectively, to minimize the conditional entropies between layers, aiming to eliminate redundancy.

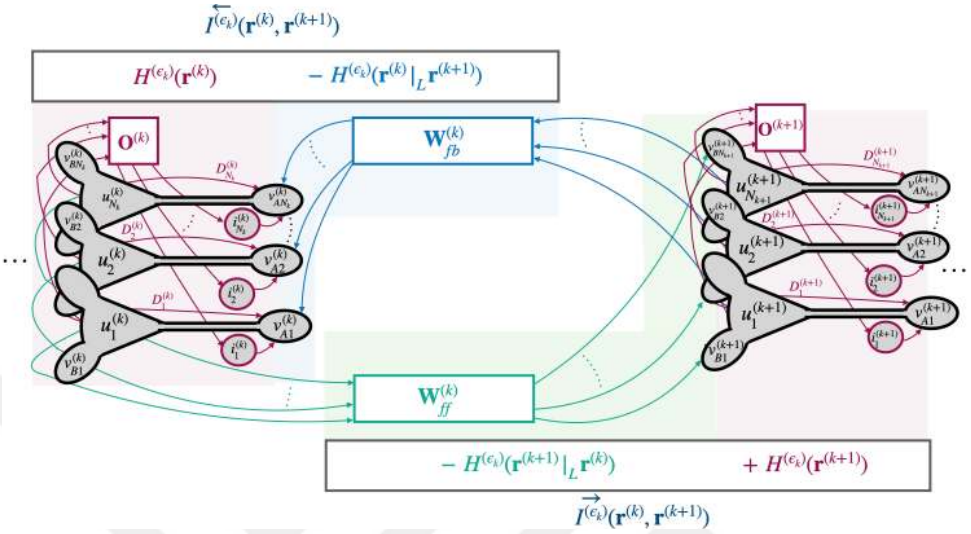


Figure 6.1: A segment of a correlative information maximization based neural network. Each layer consists of three-compartment pyramidal neurons with outputs $\mathbf{r}^{(k)}$ and membrane voltages ($\mathbf{u}^{(k)}$-soma, $\mathbf{v}_B^{(k)}$-basal dendrites, $\mathbf{v}_A^{(k)}$-distal apical dendrites) 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 $H^{(\epsilon_k)}(\mathbf{r}^{(k+1)})$ ($H^{(\epsilon_k)}(\mathbf{r}^{(k)})$).

6.3 Deep Correlative Information Maximization

6.3.1 Network Data Model

We make the assumption of having a dataset with L input data points, denoted as $\mathbf{x}[t] \in \mathbb{R}^m, t = 1, \ldots, L$ with the corresponding labels $\mathbf{y}_T[t] \in \mathbb{R}^n$. Our neural network consists of $P-1$ hidden layers, with the activities of these layers denoted as $\mathbf{r}^{(k)} \in \mathbb{R}^{N_k}, k = 1, \ldots, P-1$. For simplicity, we also refer to the input and

output of the network as $\mathbf{r}^{(0)}$ and $\mathbf{r}^{(P)}$, i.e., $\mathbf{r}^{(0)}[t] = \mathbf{x}[t]$ and $\mathbf{r}^{(P)}[t] = \hat{\mathbf{y}}[t]$. We impose polytopic constraints on the hidden and output layer activities, denoted as $\mathbf{r}^{(k)} \in \mathcal{P}^{(k)}$, where $\mathcal{P}^{(k)}$ represents the assumed polytopic domain for the k-th layer's neuronal activities [Bozkurt et al., 2023a, Tatli and Erdogan, 2021b]. It is worth noting that polytopic assumptions are reasonable since neuron activations in practice are typically bounded. Specifically, we make the specific assumption that $\mathcal{P}^{(k)} = \mathcal{B}_{\infty,+} = \{\mathbf{r} : \mathbf{0} \leq \mathbf{r} \leq \mathbf{1}\}$, indicating that the (normalized) activations lie within a non-negative unit-hypercube. Non-negativity constraints have been linked to disentangling behavior in previous studies [Plumbley, 2003, Pehlevan et al., 2017b, Whittington et al., 2023], however, we also explore extensions by considering alternative polytopic sets associated with different feature priors [Bozkurt et al., 2023a]. In a broader context, the corresponding label $\mathbf{y}_T$ can represent one-hot encoded label vectors for a classification problem or discrete/continuous valued vectors for a regression problem.

6.3.2 Correlative Information Maximization Based Signal Propagation

Stochastic CorInfoMax-based Supervised Criterion

We introduce a stochastic objective that combines the total correlative mutual information between consecutive layers with the mean-square-error (MSE) training loss.

$$J(\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(P)}) = \sum_{k=0}^{P-1} I^{(\epsilon_k)}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)}) - \frac{\beta}{2} E(\|\mathbf{y}_T - \mathbf{r}^{(P)}\|_2^2), \tag{6.1}$$

where, as defined in [Erdogan, 2022, Ozsoy et al., 2022],

$$\overrightarrow{I^{(\epsilon_k)}}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)}) = \frac{1}{2} \log\det\left(\mathbf{R}_{\mathbf{r}^{(k+1)}} + \epsilon_k \boldsymbol{I}\right) - \frac{1}{2} \log\det\left(\boldsymbol{R}_{\overrightarrow{\mathbf{e}}_*^{(k+1)}} + \epsilon_k \boldsymbol{I}\right), \tag{6.2}$$

and $\mathbf{R}_{\mathbf{r}^{(k+1)}} = \mathbb{E}[\mathbf{r}^{(k+1)}\mathbf{r}^{(k+1)^T}]$, $\mathbf{R}_{\mathbf{r}^{(k)}\mathbf{r}^{(l)}} = \mathbb{E}[\mathbf{r}^{(k)}\mathbf{r}^{(l)^T}]$ are the autocorrelation and the cross-correlation matrices corresponding to the layer activations, respectively. Additionally, $\boldsymbol{R}_{\overrightarrow{\mathbf{e}}_*^{(k+1)}} = \mathbf{R}_{\mathbf{r}^{(k+1)}} - \mathbf{R}_{\mathbf{r}^{(k+1)}\mathbf{r}^{(k)}}{}^T (\mathbf{R}_{\mathbf{r}^{(k)}} + \epsilon_k \boldsymbol{I})^{-1} \mathbf{R}_{\mathbf{r}^{(k)}\mathbf{r}^{(k+1)}}$ corresponds to the error autocorrelation matrix for the best linear regularized minimum MSE predictor of $\mathbf{r}^{(k+1)}$ from $\mathbf{r}^{(k)}$. We refer to the problem described above as the *regularized*

forward prediction problem, which is addressed through the following optimization problem

$$\underset{\boldsymbol{W}_{ff}^{(k)}}{\text{minimize}}\ E(\|\overrightarrow{\mathbf{e}}^{(k+1)}\|_2^2) + \epsilon_k \|\boldsymbol{W}_{ff}^{(k)}\|_F^2 \quad \text{s.t.} \quad \overrightarrow{\mathbf{e}}^{(k+1)} = \mathbf{r}^{(k+1)} - \boldsymbol{W}_{ff}^{(k)}\mathbf{r}^{(k)}. \tag{6.3}$$

In this context, $\mathbf{e}_*^{(k+1)}$ represents the forward prediction error associated with the optimal forward predictor $\boldsymbol{W}_{ff,*}^{(k)}$.

An alternative and equivalent expression for the correlative mutual information (CMI) can be expressed as follows (Section 2.2):

$$\overleftarrow{I}^{(\epsilon_k)}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)}) = \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{r}^{(k)}} + \epsilon_k \boldsymbol{I}) - \frac{1}{2}\log\det\left(\boldsymbol{R}_{\overleftarrow{\mathbf{e}}_*^{(k)}} + \epsilon_k \boldsymbol{I}\right), \tag{6.4}$$

where $\boldsymbol{R}_{\overleftarrow{\mathbf{e}}_*^{(k)}} = \mathbf{R}_{\mathbf{r}^{(k)}} - \mathbf{R}_{\mathbf{r}^{(k+1)}\mathbf{r}^{(k)}}{}^T(\mathbf{R}_{\mathbf{r}^{(k+1)}} + \epsilon_k \boldsymbol{I})^{-1}\mathbf{R}_{\mathbf{r}^{(k+1)}\mathbf{r}^{(k)}}$ corresponds to the error autocorrelation matrix for the optimal linear regularized minimum MSE predictor of $\mathbf{r}^{(k)}$ from $\mathbf{r}^{(k+1)}$. We define the corresponding problem as the *regularized backward prediction problem* and address it through the following optimization process:

$$\underset{\boldsymbol{W}_{fb}^{(k)}}{\text{minimize}}\ E(\|\overleftarrow{\mathbf{e}}^{(k)}\|_2^2) + \epsilon_k \|\boldsymbol{W}_{fb}^{(k)}\|_F^2 \quad \text{s.t.} \quad \overleftarrow{\mathbf{e}}^{(k)} = \mathbf{r}^{(k)} - \boldsymbol{W}_{fb}^{(k)}\mathbf{r}^{(k+1)}. \tag{6.5}$$

We observe a connection between the forward and backward prediction problems, represented by the optimizations in (6.3) and (6.5), respectively, and the two distinct yet equivalent representations of the correlative mutual information (CMI) between layers $\mathbf{r}^{(k)}$ and $\mathbf{r}^{(k+1)}$ in (6.2) and (6.4). This intrinsic link between the CMI representations and the prediction problems is of great significance, as it allows us to address the weight symmetry issue by designing a neural network architecture, as we will demonstrate in subsequent sections.

Sample-based Supervised CorInfoMax Criterion

Our objective is to develop a biologically plausible neural network that adaptively optimizes the total correlative mutual information (CMI) as defined in (6.1). In pursuit of this objective, we derive a sample-based version of (6.1).

To begin, we introduce the weighted sample auto and cross-correlation matrices

as follows:

$$\begin{aligned}\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] &= \frac{1-\lambda_{\mathbf{r}}}{1-\lambda_{\mathbf{r}}^{t}}\sum_{i=1}^{t}\lambda_{\mathbf{r}}^{t-i}\mathbf{r}^{(k)}[i]\mathbf{r}^{(k)}[i]^{T},\\ \hat{\mathbf{R}}_{\mathbf{r}^{(k)}\mathbf{r}^{(k+1)}}[t] &= \frac{1-\lambda_{\mathbf{r}}}{1-\lambda_{\mathbf{r}}^{t}}\sum_{i=1}^{t}\lambda_{\mathbf{r}}^{t-i}\mathbf{r}^{(k)}[i]\mathbf{r}^{(k+1)}[i]^{T},\end{aligned} \tag{6.6}$$

for $k = 0, \ldots, P$, respectively, and $0 \ll \lambda_{\mathbf{r}} < 1$ represents the forgetting factor. Subsequently, we define two equivalent formulations of the sample-based correlative mutual information ($\hat{I}^{(\epsilon)}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})[t]$):

$$\hat{I}^{(\epsilon_k)}_{\rightarrow}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})[t] = \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k+1)}}[t] + \epsilon_k \boldsymbol{I}) - \frac{1}{2}\log\det(\hat{\boldsymbol{R}}_{\overrightarrow{\mathbf{e}}^{(k+1)}}[t] + \epsilon_k \boldsymbol{I}), \tag{6.7}$$

$$\hat{I}^{(\epsilon_k)}_{\leftarrow}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})[t] = \frac{1}{2}\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_k \boldsymbol{I}) - \frac{1}{2}\log\det(\hat{\boldsymbol{R}}_{\overleftarrow{\mathbf{e}}^{(k)}}[t] + \epsilon_k \boldsymbol{I}), \tag{6.8}$$

where $\hat{\boldsymbol{R}}_{\overrightarrow{\mathbf{e}}^{(k+1)}}[t] = \hat{\mathbf{R}}_{\mathbf{r}^{(k+1)}}[t] - \hat{\mathbf{R}}_{\mathbf{r}^{(k)}\mathbf{r}^{(k+1)}}[t]^{T}(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_k \boldsymbol{I})^{-1}\hat{\mathbf{R}}_{\mathbf{r}^{(k)}\mathbf{r}^{(k+1)}}[t]$ represents the autocorrelation matrix for the forward prediction error at level-$(k+1)$, $\overrightarrow{\mathbf{e}}^{(k+1)}[t]$, that corresponds to the optimal linear weighted regularized least squares predictor of $\mathbf{r}^{(k+1)}[t]$ from the lower level activations $\mathbf{r}^{(k)}[t]$. Similarly, $\hat{\boldsymbol{R}}_{\overleftarrow{\mathbf{e}}^{(k)}}[t] = \hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] - \hat{\mathbf{R}}_{\mathbf{r}^{(k)}\mathbf{r}^{(k+1)}}[t](\hat{\mathbf{R}}_{\mathbf{r}^{(k+1)}}[t]+\epsilon_k \boldsymbol{I})^{-1}\hat{\mathbf{R}}_{\mathbf{r}^{(k)}\mathbf{r}^{(k+1)}}[t]^{T}$ represents the autocorrelation matrix for the backward prediction error at level-(k), $\overleftarrow{\mathbf{e}}^{(k)}[t]$, that corresponds to the optimal linear weighted regularized least squares predictor of $\mathbf{r}^{(k)}[t]$ from the higher level activations $\mathbf{r}^{(k+1)}[t]$.

As a result, we can express the sample-based CorInfoMax optimization as follows:

$$\underset{\mathbf{r}^{(k)}[t], k=0,\ldots,P}{\text{maximize}} \quad \sum_{k=0}^{P-1}\hat{I}^{(\epsilon_k)}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})[t] - \frac{\beta}{2}\|\mathbf{y}_T[t] - \mathbf{r}^{(P)}[t]\|_2^2 \tag{6.9a}$$

$$\text{subject to} \quad \mathbf{r}^{(k)}[t] \in \mathcal{P}^{(k)}, k = 1, \ldots, P, \tag{6.9b}$$

$$\mathbf{r}^{(0)}[t] = \mathbf{x}[t], \tag{6.9c}$$

The first-order Taylor series approximation of the log det expressions on the right-

hand side of (6.7) and (6.8) are:

$$\log\det\left(\hat{\boldsymbol{R}}_{\overrightarrow{\mathbf{e}}^{(k+1)}}[t]+\epsilon_k\boldsymbol{I}\right)\approx\frac{1}{\epsilon_k}\operatorname{Tr}\left(\hat{\boldsymbol{R}}_{\overrightarrow{\mathbf{e}}^{(k+1)}}[t]\right)+N_{k+1}\log(\epsilon_k)$$
$$=\frac{1}{\epsilon_k}\sum_{i=1}^{t}\lambda_{\mathbf{r}}^{t-i}\|\mathbf{r}^{(k+1)}[i]-\mathbf{W}_{ff,*}^{(k)}[t]\mathbf{r}^{(k)}[i]\|_2^2+\epsilon_k\|\boldsymbol{W}_{ff,*}^{(k)}[t]\|_F^2+N_{k+1}\log(\epsilon_k), \tag{6.10}$$

$$\log\det\left(\hat{\boldsymbol{R}}_{\overleftarrow{\mathbf{e}}^{(k)}}[t]+\epsilon_k\boldsymbol{I}\right)\approx\frac{1}{\epsilon_k}\operatorname{Tr}\left(\hat{\boldsymbol{R}}_{\overleftarrow{\mathbf{e}}^{(k)}}[t]\right)+N_k\log(\epsilon_k)$$
$$=\frac{1}{\epsilon_k}\sum_{i=1}^{t}\lambda_{\mathbf{r}}^{t-i}\|\mathbf{r}^{(k)}[i]-\mathbf{W}_{fb,*}^{(k)}[t]\mathbf{r}^{(k+1)}[i]\|_2^2+\epsilon_k\|\boldsymbol{W}_{fb,*}^{(k)}[t]\|_F^2+N_k\log(\epsilon_k). \tag{6.11}$$

Note that in (6.10), $\mathbf{W}_{ff,*}^{(k)}[t]$ refers to the optimal coefficients of the linear regularized weighted least squares forward predictor used to predict $\mathbf{r}^{(k+1)}[i]$ from $\mathbf{r}^{(k)}[i]$ for $i=1,\ldots,t$. Similarly, $\mathbf{W}_{fb,*}^{(k)}[t]$ in (6.11) represents the optimal coefficients of the linear regularized weighted least squares backward predictor used to predict $\mathbf{r}^{(k)}[i]$ from $\mathbf{r}^{(k+1)}[i]$ for $i=1,\ldots,t$. As a result, the optimal choices of forward and backward predictor coefficients are coupled to the optimal selections of layer activations.

In the online optimization process, we begin by relaxing this requirement and initializing the predictor coefficients with random selections. Throughout the learning process, we employ a coordinate ascent-based approach to optimize the activation signals and predictor coefficients. Specifically, at each time step t, we first optimize the activations $\{\mathbf{r}^{(k)}[t], k=1,\ldots,P\}$, where we assume that the predictor coefficients are fixed. We then proceed to update the forward and backward predictor coefficients, denoted as $\boldsymbol{W}_{ff}^{(k)}$ and $\boldsymbol{W}_{fb}^{(k)}$, for $k=1,\ldots,P$, respectively. The goal is to minimize the forward and backward prediction errors by adjusting these coefficients. As the algorithm iterates, the predictor coefficients gradually approach their optimal values.

For the first phase of the online optimization, we utilize a projected gradient ascent-based approach for activations. Specifically, for $k=1,\ldots,P-1$, the layer activation vector $\mathbf{r}^{(k)}[t]$ is incorporated into 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 compute the gradient with respect to $\mathbf{r}^{(k)}[t]$, we can utilize the modified versions of the expressions in (6.7) and (6.8), where we

employ the approximations provided in (6.10)-(6.11), and the optimal predictors are substituted with their current estimates:

$$\begin{aligned}\nabla_{\mathbf{r}^{(k)}}\hat{J}_k(\mathbf{r}^{(k)})[t] &= \nabla_{\mathbf{r}^{(k)}}\hat{\overrightarrow{I}}^{(\epsilon_{k-1})}(\mathbf{r}^{(k-1)},\mathbf{r}^{(k)})[t] + \nabla_{\mathbf{r}^{(k)}}\hat{\overleftarrow{I}}^{(\epsilon_k)}(\mathbf{r}^{(k)},\mathbf{r}^{(k+1)})[t] \\ &= \frac{1}{2}\nabla_{\mathbf{r}^{(k)}}(\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t]+\epsilon_{k-1}\boldsymbol{I}) + \log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t]+\epsilon_k\boldsymbol{I})) \\ &\quad - \frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}^{(k)}[t] - \frac{1}{\epsilon_k}\overleftarrow{\mathbf{e}}^{(k)}[t], \end{aligned} \tag{6.12}$$

where

$$\overrightarrow{\mathbf{e}}^{(k)}[t] = \mathbf{r}^{(k)}[t] - \boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t], \quad \overleftarrow{\mathbf{e}}^{(k)}[t] = \mathbf{r}^{(k)}[t] - \boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)}[t] \tag{6.13}$$

correspond to the forward and backward prediction errors at level-k, respectively. Based on the procedure outlined in [Bozkurt et al., 2023a], we can express the gradient term in (6.12) as follows:

$$\frac{1}{2}\nabla_{\mathbf{r}^{(k)}}(\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t]+\epsilon_{k-1}\boldsymbol{I}) + \log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t]+\epsilon_k\boldsymbol{I})) = 2\gamma\boldsymbol{B}_{\mathbf{r}^{(k)}}[t]\mathbf{r}^{(k)}[t], \tag{6.14}$$

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 function for the final layer can be represented as follows:

$$\begin{aligned}&\nabla_{\mathbf{r}^{(P)}}(\hat{\overrightarrow{I}}^{(\epsilon_{P-1})}(\mathbf{r}^{(P-1)},\mathbf{r}^{(P)})[t] - \frac{\beta}{2}\|\mathbf{r}^{(P)}[t]-\boldsymbol{y}_T[t]\|_2^2) \\ &\qquad = \gamma\boldsymbol{B}_{\mathbf{r}^{(P)}}[t]\mathbf{r}^{(P)}[t] - \frac{1}{\epsilon_{P-1}}\overrightarrow{\mathbf{e}}^{(P)}[t] - \beta(\mathbf{r}^{(P)}[t]-\boldsymbol{y}_T[t]).\end{aligned}$$

6.3.3 Neural Network Formulation based on Information Maximization

In this section, we propose a biologically plausible neural network that is built upon the correlative information maximization-based network propagation model discussed in Section 6.3.2. To accomplish this, we utilize projected gradient ascent optimization to determine layer activations $\mathbf{r}^{(1)}[t], \mathbf{r}^{(2)}[t], \ldots, \mathbf{r}^{(P)}[t]$. These activations play a crucial role in shaping the network's structure and dynamics, as well as updating the corresponding synapses that govern the learning dynamics of the network.

Network structure and neural dynamics

This section demonstrates that the projected gradient ascent solution to the optimization in (6.9) defines a multilayer recurrent neural network. To achieve this,

we introduce the intermediate variable $\mathbf{u}^{(k)}$ to represent the updated layer-k activations before projecting them onto the domain set $\mathcal{P}^{(k)}$. By utilizing the gradient expressions provided in (6.12)-(6.14), we can write the network dynamics for layers $k = 1, \ldots, P-1$ as follows:

$$\tau_{\mathbf{u}} \frac{d\mathbf{u}^{(k)}[t;s]}{ds} = -g_{lk}\mathbf{u}^{(k)}[t;s] + \frac{1}{\epsilon_k}\boldsymbol{M}^{(k)}[t]\boldsymbol{r}^{(k)}[t;s] - \frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}_u^{(k)}[t;s] - \frac{1}{\epsilon_k}\overleftarrow{\mathbf{e}}_u^{(k)}[t;s], \tag{6.15}$$

$$\overrightarrow{\mathbf{e}}_u^{(k)}[t;s] = \mathbf{u}^{(k)}[t;s] - \boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t;s], \tag{6.16}$$

$$\overleftarrow{\mathbf{e}}_u^{(k)}[t;s] = \mathbf{u}^{(k)}[t;s] - \boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)}[t;s]. \tag{6.17}$$

$$\mathbf{r}^{(k)}[t;s] = \sigma_+(\mathbf{u}^{(k)}[t;s]), \tag{6.18}$$

In the above expressions, $\tau_{\mathbf{u}}$ denotes the update time constant, $\boldsymbol{M}^{(k)}[t] = 2\epsilon_k(\gamma\boldsymbol{B}^{(k)}[t] + g_{lk}\boldsymbol{I})$, and σ_+ corresponds the elementwise clipped-ReLU function, which performs a projection onto the nonnegative unit-hypercube $\mathcal{B}_{\infty,+}$, defined as

$$\sigma_+(u) = \min(1, \max(u, 0)).$$

To reinterpret the dynamics described in (6.15) to (6.18) as a multi-compartmental neural network, we introduce the following signals for $k = 1, \ldots, P-1$:

$$\mathbf{v}_A^{(k)}[t;s] = \boldsymbol{M}^{(k)}[t]\boldsymbol{r}^{(k)}[t;s] + \boldsymbol{W}_{fb}^{(k)}[t]\mathbf{r}^{(k+1)}[t;s], \tag{6.19}$$

$$\mathbf{v}_B^{(k)}[t;s] = \boldsymbol{W}_{ff}^{(k-1)}[t]\mathbf{r}^{(k-1)}[t;s]. \tag{6.20}$$

Using the defined signals, we can rewrite the network activation dynamics from (6.15) to (6.18) as follows:

$$\tau_{\mathbf{u}} \frac{d\mathbf{u}^{(k)}[t;s]}{ds} = -g_{lk}\mathbf{u}^{(k)}[t;s] + g_{A,k}(\mathbf{v}_A^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s]) + g_{B,k}(\mathbf{v}_B^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s]), \tag{6.21}$$

$$\mathbf{r}^{(k)}[t;s] = \sigma_+(\mathbf{u}^{(k)}[t;s]), \tag{6.22}$$

where $g_{A,k} = \frac{1}{\epsilon_{k-1}}$ and $g_{B,k} = \frac{1}{\epsilon_k}$. Similarly, for the output layer with $k = P$, we utilize the same expressions as (6.21) and (6.22), with the exception that in this case we have:

$$\mathbf{v}_A^{(P)}[t;s] = \boldsymbol{M}^{(P)}[t]\boldsymbol{r}^{(k)}[t;s] - (\mathbf{r}^{(P)}[t;s] - \boldsymbol{y}_T[t]) \tag{6.23}$$

$$\mathbf{v}_B^{(P)}[t;s] = \boldsymbol{W}_{ff}^{(P-1)}[t]\mathbf{r}^{(P-1)}[t;s], \tag{6.24}$$

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

Remarkably, equations (6.19) to (6.24) unveil a biologically plausible neural network that integrates three-compartment pyramidal neuron models, as described in [Sacramento et al., 2018, Golkar et al., 2022]. This intricate architecture, of which two-layer segment is depicted in Figure 6.1, naturally arises from the proposed correlative information maximization framework. Within this network structure:

- $\mathbf{u}^{(k)}$ represents the membrane potentials for neuronal somatic compartments of the neurons in layer-k, where $\tau_{\mathbf{u}}$ denotes the membrane leak time constant of soma.

- The variable $\mathbf{v}_B^{(k)}$ corresponds to the membrane potentials of the basal dendrite compartments, which receive feedforward input from the previous layer.

- The variable $\mathbf{v}_A^{(k)}$ represents the membrane potentials of the distal apical dendrite compartments, which receive top-down input from the subsequent layer and lateral inputs given by $\boldsymbol{M}^{(k)}[t]\boldsymbol{r}^{(k)}$ in (6.19) and (6.23). By decomposing $\boldsymbol{M}^{(k)}$ into $\boldsymbol{D}^{(k)} - \boldsymbol{O}^{(k)}$, we observe that $\boldsymbol{D}^{(k)}$ corresponds to autapses [Lübke et al., 1996], while the off-diagonal component $\boldsymbol{O}^{(k)}$ represents lateral inhibition synapses. We denote the activations of SST interneurons [Urban-Ciecko and Barth, 2016] that generate lateral inhibitions to the apical dendrites as $\mathbf{i}^{(k)} = -\boldsymbol{O}^{(k)}\mathbf{r}^{(k)}$.

- Forward (backward) prediction errors are reflected in the differences in membrane voltages between the soma and basal (distal) compartments of the pyramidal neurons.

- Forward (backward) prediction coefficients $\boldsymbol{W}_{ff}^{(k)}$ ($\boldsymbol{W}_{fb}^{(k)}$) are linked to feedforward (top-down) synapses that connect layers (k) and $(k+1)$.

- The inverse of the regularization coefficient ϵ_k is associated with the conductance between soma and dendritic compartments in the intermediate layers. On the other hand, at the output layer, the augmentation constant β represents

the conductance between soma and distal compartments. This relationship can be justified by modifying the objective in (6.9a) as follows:

$$\sum_{k=0}^{P-1} \hat{I}^{(\epsilon_k)}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})[t] + \frac{1}{2}\overleftarrow{\hat{I}^{(\beta^{-1})}}(\mathbf{r}^{(P)}, \mathbf{y}_T)[t], \tag{6.25}$$

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

6.3.4 Learning Dynamics

The network parameters include the feedforward $\boldsymbol{W}_{ff}^{(k)}$, feedback $\boldsymbol{W}_{fb}^{(k)}$ and lateral $\boldsymbol{B}^{(k)}$ coefficients.The learning dynamics of these coefficients can be described as follows:

- *Feedforward Coefficients* are associated with the forward prediction problem defined by the optimization in (6.3). The online optimization objective function for the regularized forward prediction error can be defined 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

$$\frac{\partial C_{ff}(\boldsymbol{W}_{ff}^{(k)}[t])}{\partial \boldsymbol{W}_{ff}^{(k)}} = 2\epsilon_k \boldsymbol{W}_{ff}^{(k)}[t] - 2\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T. \tag{6.26}$$

  In Section 6.4, we show that (6.26) can be written in terms of the membrane voltage difference between the distal apical and soma compartments of the neuron, using the equilibrium condition for the neuronal dynamics:

$$-\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T = g_{B,k}^{-1}(g_{A,k}\mathbf{v}_A^{(k)}[t] - (g_{lk} + g_{A_k})\mathbf{u}_*^{(k)}[t] + \mathbf{h}_*[t])\mathbf{r}^{(k)}[t]^T, \tag{6.27}$$

  where $\mathbf{h}_*[t]$ is nonzero only for neurons that are silent or firing at the maximum rate.

- Similarly, *Feedback Coefficients* are associated with the backward prediction problem defined by the optimization in (6.5). The online optimization objective function for the regularized backward prediction error can be defined 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

$$\frac{\partial C_{fb}(\boldsymbol{W}_{fb}^{(k)}[t])}{\partial \boldsymbol{W}_{fb}^{(k)}} = 2\epsilon_k \boldsymbol{W}_{fb}^{(k)}[t] - 2\overleftarrow{\mathbf{e}}^{(k)}[t]\mathbf{r}^{(k+1)}[t]^T. \tag{6.28}$$

For the computation of updates to both the feedforward and feedback coefficients, we employ the Equilibrium Propagation (EP) approach [Scellier and Bengio, 2017]. The update terms are obtained based on the contrastive expressions of the partial derivatives in (6.26) and (6.28) during the nudge phase, where $\beta = \beta' > 0$, and the free phase, where $\beta = 0$:

$$\delta \boldsymbol{W}_{ff}^{(k)}[t] \propto \frac{1}{\beta'} \left((\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T)\Big|_{\beta=\beta'} - (\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T)\Big|_{\beta=0} \right), \tag{6.29}$$

$$\delta \boldsymbol{W}_{fb}^{(k)}[t] \propto \frac{1}{\beta'} \left((\overleftarrow{\mathbf{e}}^{(k)}[t]\mathbf{r}^{(k+1)}[t]^T)\Big|_{\beta=\beta'} - (\overleftarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k+1)}[t]^T)\Big|_{\beta=0} \right). \tag{6.30}$$

- *Lateral Coefficients*, $\mathbf{B}^{(k)}$ are the inverses of the correlation matrices perturbed by $\epsilon\mathbf{I}$. The learning dynamics of these coefficients can be expressed using the rule presented in [Bozkurt et al., 2023a] after the nudge phase:

$$\boldsymbol{B}^{(k)}[t+1] = \lambda_{\mathbf{r}}^{-1}(\boldsymbol{B}^{(k)}[t] - \gamma \mathbf{z}^{(k)}[t]\mathbf{z}^{(k)}[t]^T), \text{ where } \mathbf{z}^{(k)} = \boldsymbol{B}^{(k)}[t]\mathbf{r}^{(k)}[t].$$

6.4 Alternative Expression for the Feedforward Weight Updates

In Section 6.3.4, we obtained the partial derivative of the forward prediction cost with respect to the feedforward weights, which is given by:

$$\frac{\partial C_{ff}(\boldsymbol{W}_{ff}^{(k)}[t])}{\partial \boldsymbol{W}_{ff}^{(k)}} = 2\epsilon_k \boldsymbol{W}_{ff}^{(k)}[t] - 2\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T. \tag{6.31}$$

In this equation, $\overrightarrow{\mathbf{e}}^{(k+1)}[t]$ represents the feedforward error value at the point when system dynamics have reached a steady state. To characterize this steady state value, we examine the optimal solution of the optimization problem used to derive the system dynamics:

$$\begin{aligned} \underset{\mathbf{r}^{(k)}[t]}{\text{maximize}} \quad & \left(\tfrac{1}{2}(\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_{k-1}\boldsymbol{I}) + \log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_k \boldsymbol{I})) \right. \\ & \left. - \tfrac{1}{2\epsilon_{k-1}} \left\| \overrightarrow{\mathbf{e}}^{(k)}[t] \right\|_2^2 - \tfrac{1}{2\epsilon_k} \left\| \overleftarrow{\mathbf{e}}^{(k)}[t] \right\|_2^2 \right) \\ \text{subject to} \quad & \mathbf{0} \leq \mathbf{r}^{(k)}[t] \leq \mathbf{1}. \end{aligned} \tag{6.32}$$

We can formulate the corresponding Lagrangian function for this optimization as follows:

$$L(\mathbf{r}^{(k)}[t], \mathbf{q}_1[t], \mathbf{q}_2[t]) = O(\mathbf{r}^{(k)}[t]) + \mathbf{q}_1[t]^T(\mathbf{r}^{(k)}[t]) + \mathbf{q}_2[t]^T(\mathbf{1} - \mathbf{r}^{(k)}[t]). \tag{6.33}$$

Here, $O(\mathbf{r}^{(k)}[t])$ represents the objective function from (6.32), and $\mathbf{q}_1[t], \mathbf{q}_2[t] \geq \mathbf{0}$ are the Lagrange multipliers. By applying the KKT optimality conditions [Boyd et al., 2004], we obtain the following equations:

$$\nabla_{\mathbf{r}^{(k)}} L(\mathbf{r}_*^{(k)}[t], \mathbf{q}_{1*}[t], \mathbf{q}_{2*}[t]) = \mathbf{0}, \tag{6.34}$$

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

$$2g_{B,k}\overrightarrow{\mathbf{e}}_*^{(k+1)}[t] + 2\left(g_{A,k}\mathbf{v}_A^{(k)}[t] - (g_{lk} + g_{A_k})\mathbf{r}_*^{(k)}[t]\right) + \mathbf{q}_{1*}[t] + \mathbf{q}_{2*}[t] = \mathbf{0}. \tag{6.35}$$

In the above equations, $\mathbf{q}_{1*}[t]$ and $\mathbf{q}_{2*}[t]$ are entirely zero except at the indices where $\mathbf{r}_*^{(k)}[t]$ equals 0 or 1. This is due to the nonnegative-clipping operation of $\sigma+(\mathbf{u}_*^{(k)}[t])$.

Therefore, by defining $\mathbf{q}_*[t] = \mathbf{q}_{1*}[t] + \mathbf{q}_{2*}[t]$, the outer product term in (6.31) used in the feedforward weight update can be equivalently written as

$$-2\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T = \frac{2}{g_{B,k}}\left(g_{A,k}\mathbf{v}_A^{(k)}[t] - (g_{lk} + g_{A_k})\mathbf{r}_*^{(k)}[t] + \frac{1}{2}\mathbf{q}_*[t]\right)\mathbf{r}^{(k)}[t]^T. \tag{6.36}$$

Based on the relationship $\mathbf{r}_*^{(k)}[t] = \sigma_+(\mathbf{u}_*^{(k)}[t])$, we can express the connection 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. Thus, outer product term in (6.36) can be rewritten as

$$-2\overrightarrow{\mathbf{e}}^{(k+1)}[t]\mathbf{r}^{(k)}[t]^T = \frac{2}{g_{B,k}}\left(g_{A,k}\mathbf{v}_A^{(k)}[t] - (g_{lk} + g_{A_k})\mathbf{u}_*^{(k)}[t] + \mathbf{h}_*[t]\right)\mathbf{r}^{(k)}[t]^T, \tag{6.37}$$

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

In line with a similar approach presented in [Golkar et al., 2022], we conclude that the update expression for the feedforward term can be derived from the difference between the membrane voltages of the soma and the distal apical compartments. This relation aligns with experimental observations that highlight the influence of basal synaptic plasticity on apical calcium plateau potentials [Larkum, 2013].

Figure 6.2 illustrates a CorInfoMax network with a single hidden layer. In this case, both the hidden and output layers share the same constraint set $\mathcal{P} = \mathcal{B}_{\infty,+}$, which corresponds to the nonnegative unit-hypercube constraint.

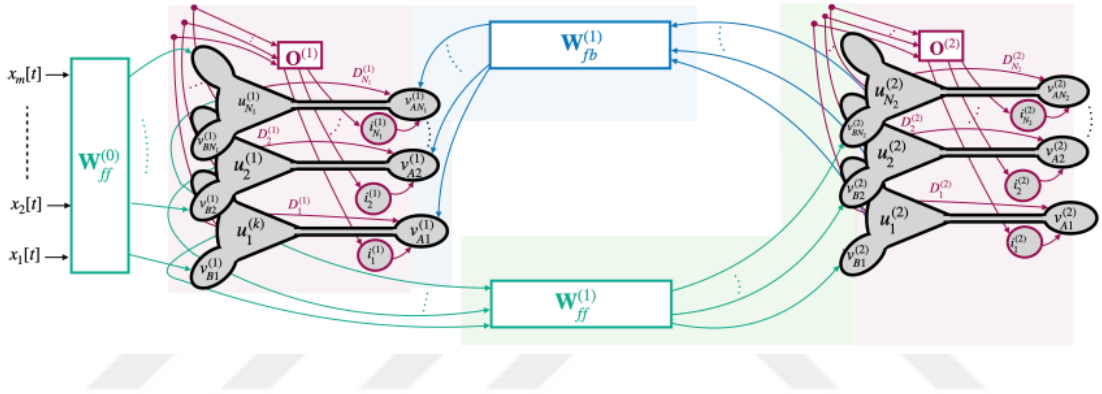


Figure 6.2: Two layer correlative information maximization based neural network with non-negative antisparsity constraint: $0 \leq \mathbf{r}^{(l)} \leq 1, l = 1, 2$.

6.5 Employing Sparsity Assumption on Neuronal Activities

In this section, we provide detailed information about the structure of the CorInfoMax network and the resulting neuronal dynamics that arise when the activation domain is selected as $\mathcal{P}^{(k)} = \mathcal{B}_{1,+} = \{\mathbf{r} : \|\mathbf{r}\|_1 \leq 1, \mathbf{0} \leq \mathbf{r}\}$. It is worth noting that this domain corresponds to the intersection of the the nonnegative orthant and ℓ_1-norm ball.

To derive the network dynamics for $\mathbf{r}^{(k)}[t]$, we consider the following constrained optimization problem, similar to (6.32):

$$\begin{aligned}
\underset{\mathbf{r}^{(k)}[t]}{\text{maximize}} \quad & \left(\tfrac{1}{2}(\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_{k-1}\boldsymbol{I}) + \log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_k \boldsymbol{I})) \right. \\
& \left. - \tfrac{1}{2\epsilon_{k-1}} \left\| \overrightarrow{\mathbf{e}}^{(k)}[t] \right\|_2^2 - \tfrac{1}{2\epsilon_k} \left\| \overleftarrow{\mathbf{e}}^{(k)}[t] \right\|_2^2 \right) \qquad (6.38) \\
\text{subject to} \quad & \|\mathbf{r}^{(k)}[t]\|_1 \leq 1, \\
& \mathbf{0} \leq \mathbf{r}^{(k)}[t].
\end{aligned}$$

We can express the Lagrangian min-max problem corresponding to this optimization as follows:

$$\underset{q^{(k)}[t]\geq 0}{\text{minimize}} \; \underset{\mathbf{r}^{(k)}[t]\geq \mathbf{0}}{\text{maximize}} \; L(\mathbf{r}^{(k)}[t], q^{(k)}[t]) = O(\mathbf{r}^{(k)}[t]) + q^{(k)}[t](1 - \|\mathbf{r}^{(k)}[t]\|_1). \quad (6.39)$$

In the above expression, $O(\mathbf{r}^{(k)}[t])$ represents the objective in (6.38). By utilizing the proximal gradient update [Parikh et al., 2014] for $\mathbf{r}^{(k)}[t]$ using the gradient expression in (6.12), we can describe the output dynamics for layer-k as:

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

In this formulation, we propose the introduction of the intermediate variable $\mathbf{u}^{(k)}$, with ReLU denoting the element-wise rectified linear unit. Subsequently, we establish the definitions of the apical and basal potentials as follows:

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

Having established these potentials, we can express the output dynamics in a manner

akin to (6.21) and (6.22).

$$\begin{aligned}\tau_{\mathbf{u}}\frac{d\mathbf{u}^{(k)}[t;s]}{ds} &= -g_{lk}\mathbf{u}^{(k)}[t;s] + g_{A,k}(\mathbf{v}_A^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s]) \\ &\quad + g_{B,k}(\mathbf{v}_B^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s]), \qquad (6.40)\\ \mathbf{r}^{(k)}[t;s] &= \mathrm{ReLU}(\mathbf{u}^{(k)}[t;s]). \qquad (6.41)\end{aligned}$$

To obtain the update for the Lagrangian variable $q^{(k)}[t;s]$, we can derive it through dual minimization, yielding the following expression:

$$\begin{aligned}\frac{da^{(k)}[t;s]}{ds} &= -a^{(k)}[t;s] + \sum_{j=1}^{N_k}\mathbf{r}_j^{(k)}[t;s] - 1 + q^{(k)}[t;s], \\ q^{(k)}[t;s] &= \mathrm{ReLU}(a^{(k)}[t;s]).\end{aligned} \qquad (6.42)$$

In the given formulation, the Lagrangian variable $q^{(k)}$ represents an inhibitory interneuron that receives input from all neurons in layer-k and generates an inhibitory signal to the apical compartments of these neurons. Figure 6.3 illustrates a CorInfoMax network with a single hidden layer, where both the hidden and output layers adhere to the same constraint set $\mathcal{P} = \mathcal{B}_{1,+}$. In comparison to the CorInfoMax-$\mathcal{B}_{\infty,+}$ network structure in Figure 6.2, this network incorporates additional interneurons, specifically $q^{(1)}$ and $q^{(2)}$, to enforce sparsity constraint on hidden and output layer networks.

6.6 On the Asymmetry of Feedforward and Feedback Weights

We highlight that a key contribution of the proposed supervised CorInfoMax framework is its ability to address the weight symmetry problem that arises in certain biologically plausible neural network frameworks. In this section, our objective is to provide further elaboration on this aspect.

6.6.1 The Importance of the Choice of Alternative CMI Expressions

In formulating the update formula for the layer activation $\mathbf{r}^{(k)}$, we utilized the gradient (6.12). This gradient consists of two alternative expressions of CMI: the first expression, $\hat{I}^{\overrightarrow{(\epsilon_{k-1})}}(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k)})[t]$ captures the CMI with the preceding layer,

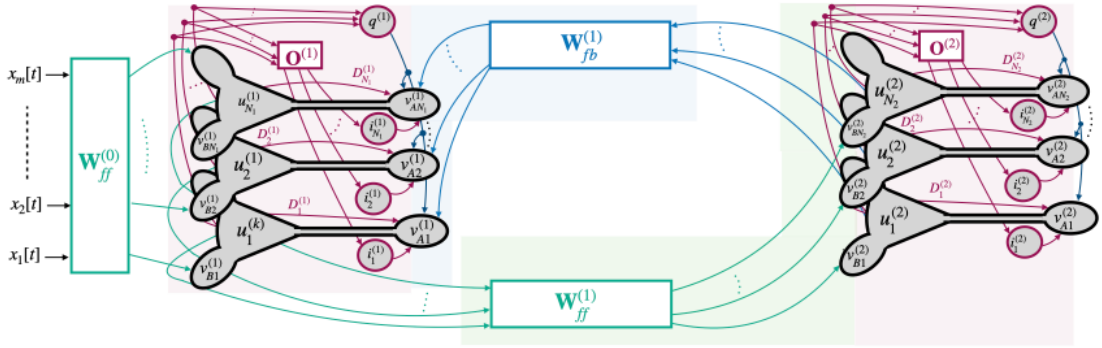


Figure 6.3: Two layer correlative information maximization based neural network with non-negative sparsity constraint: $\|\mathbf{r}^{(l)}\|_1 \leq 1$, and $\mathbf{r}^{(l)} \geq \mathbf{0}, l = 1, 2$. Interneurons with activations $q^{(1)}$ and $q^{(2)}$ are to impose ℓ_1-norm constraints for both layers.

incorporating the forward prediction error to estimate $\mathbf{r}^{(k)}$. The second expression, $\hat{I}^{\overleftarrow{(\epsilon_k)}}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})[t]$, represents the CMI with the subsequent layer, encompassing the backward prediction error to forecast $\mathbf{r}^{(k)}$. These carefully chosen expressions played a pivotal role in achieving a canonical network model that avoids the weight symmetry issue.

To emphasize the importance of our choice, let us consider an alternative representation of (6.12):

$$\begin{aligned}\nabla_{\mathbf{r}^{(k)}} \hat{J}_k(\mathbf{r}^{(k)})[t] &= \nabla_{\mathbf{r}^{(k)}} \hat{I}^{\overrightarrow{(\epsilon_{k-1})}}(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k)})[t] + \nabla_{\mathbf{r}^{(k)}} \hat{I}^{\overrightarrow{(\epsilon_k)}}(\mathbf{r}^{(k)}, \mathbf{r}^{(k+1)})[t] \\ &= \frac{1}{2}\nabla_{\mathbf{r}^{(k)}}(\log\det(\hat{\mathbf{R}}_{\mathbf{r}^{(k)}}[t] + \epsilon_{k-1}\boldsymbol{I}) - \frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}^{(k)}[t] - \frac{1}{\epsilon_k}\boldsymbol{W}_{ff}^{(k)^T}\overrightarrow{\mathbf{e}}^{(k+1)}[t]. \end{aligned} \quad (6.43)$$

In this scenario, both CMI expressions are based on forward prediction errors. The gradient expression in (6.43) 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 $\boldsymbol{W}_{ff}^{(k)^T}$. Hence, selecting (6.43) would lead to a derived network model with symmetric feedforward and feedback weights, which highlights the significance of our alternative formulation in Section 6.3.2.

This situation is commonly observed in predictive coding-based supervised approaches, as described in works [Whittington and Bogacz, 2017, Song et al., 2020],

where the loss function is constructed using feedforward prediction errors. The symmetric feedback pathway arises from the computation of gradients for this loss function. Recently, [Golkar et al., 2022] introduced a novel loss function that incorporates two different feedforward prediction error components for the same branch. By combining this approach with upper bound optimization techniques and a whitening assumption, they manage to circumvent the weight symmetry issue. In contrast, our CorInfoMax framework presents a natural solution to the weight transport issue observed in predictive coding networks. This is achieved by meticulously choosing alternative CMI expressions, thereby eliminating the requirement for the whitening assumption.

6.6.2 The Analytical Comparison of Forward and Backward Prediction Coefficients

Concerning the correlative mutual information component of the stochastic objective introduced in (6.1), the optimal solutions for both the feedforward and feedback weights are obtained by solving the problems (6.3) and (6.5), respectively. To provide further clarity, these expressions can be formulated as follows:

$$\boldsymbol{W}_{ff,*}^{(k)} = \boldsymbol{R}_{\mathbf{r}^{(k)}\mathbf{r}^{(k+1)}}^{T}(\boldsymbol{R}_{\mathbf{r}^{(k)}}+\epsilon_k\mathbf{I})^{-1}, \tag{6.44}$$

$$\boldsymbol{W}_{fb,*}^{(k)} = \boldsymbol{R}_{\mathbf{r}^{(k)}\mathbf{r}^{(k+1)}}(\boldsymbol{R}_{\mathbf{r}^{(k+1)}}+\epsilon_k\mathbf{I})^{-1}. \tag{6.45}$$

Thus, the condition $\boldsymbol{W}_{ff,}^{(k)} = \boldsymbol{W}_{fb,}^{(k)^T}$ is generally not satisfied. Symmetry may only be expected in specific cases, such as when the layer activations have a zero mean and the autocorrelation matrices satisfy 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 focuses solely on the mutual information maximization component of the objective, yet it provides insight into the anticipated asymmetry of the forward and backward weights.

6.7 Numerical Experiments

In the following subsections, we provide detailed information about our experimental setup. To evaluate the effectiveness of our proposed framework, we conducted experiments on four image classification tasks: MNIST [LeCun and Cortes, 2010],

Fashion MNIST [Xiao et al., 2017], CIFAR10, and CIFAR100 [Krizhevsky et al., 2009]. These image datasets were obtained from the PyTorch library [Paszke et al., 2019].

The MNIST dataset consists of 60000 grayscale training images of hand-written digits and 10000 test images, each with a dimension of 28×28 pixels. Similarly, the Fashion MNIST dataset follows the same format, size, and number of training and test images as MNIST. However, instead of digits, it contains images of clothing items categorized into 10 different classes.

On a larger scale, we have the CIFAR10 and CIFAR100 datasets,both comprising 32×32 RGB images. The CIFAR10 dataset consists of 50000 training images and 10000 test images, with each image labeled with one of 10 object categories. On the other hand, CIFAR100 maintains the same number of training and test images as CIFAR10 but features a broader range of 100 object categories.

In our preprocessing step, we begin by mapping the pixel values of the images to the range of $[0, 1]$ by dividing them by 255. For the CIFAR10 and CIFAR100 datasets, normalization is carried out using the mean and standard deviation values provided in the published codes of [Laborieux et al., 2021, Laborieux and Zenke, 2022]. To suit the training of MLP architectures, the images are flattened, converting them into one-dimensional vectors before being inputted into the neural networks. It is important to note that we did not employ any augmentation techniques in our numerical experiments.

Python scripts and notebooks for each experiment, which enable replication of the reported results, are available on GitHub [1].

6.7.1 On the Computation of Neural Dynamics

The input-output relationships of the CorInfoMax-$\mathcal{B}_{\infty,+}$ network for each segment are implicitly determined by the neural dynamics equations given by (6.21)-(6.22). To solve these equations, we employ a discrete-time method outlined in Algorithm 4. Similarly, the input-output relationships of the CorInfoMax-$\mathcal{B}_{1,+}$ network for each

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

segment, with an additional inhibition signal $q^{(k)}$, are implicitly determined by the equations given by (6.40), (6.41), and (6.42). Again, to solve these equations, we employ a discrete-time method outlined in Algorithm 5.

Algorithm 4 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**
  $\tau_{\mathbf{u}} \frac{d\mathbf{u}^{(k)}[t;s]}{ds} = -g_{lk}\mathbf{u}^{(k)}[t;s] + g_{A,k}(\mathbf{v}_A^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s]) + g_{B,k}(\mathbf{v}_B^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s])$,
  $\mathbf{u}^{(k)}[t;s+1] = \mathbf{u}^{(k)}[t;s] + \mu_{\mathbf{u}}[s]\tau_{\mathbf{u}} \frac{d\mathbf{u}^{(k)}[t;s]}{ds}$
  $\mathbf{r}^{(k)}[t;s+1] = \sigma_+(\mathbf{u}^{(k)}[t;s+1])$
  $s = s+1$, and adjust $\mu_{\mathbf{u}}[s]$ if necessary.
end while

Algorithm 5 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**
  $\tau_{\mathbf{u}} \frac{d\mathbf{u}^{(k)}[t;s]}{ds} = -g_{lk}\mathbf{u}^{(k)}[t;s] + g_{A,k}(\mathbf{v}_A^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s]) + g_{B,k}(\mathbf{v}_B^{(k)}[t;s] - \mathbf{u}^{(k)}[t;s])$,
  $\frac{da^{(k)}[t;s]}{ds} = -a^{(k)}[t;s] + \sum_{j=1}^{N_k} \mathbf{r}_j^{(k)}[t;s] - 1 + q^{(k)}[t;s]$
  $\mathbf{u}^{(k)}[t;s+1] = \mathbf{u}^{(k)}[t;s] + \mu_{\mathbf{u}}[s]\tau_{\mathbf{u}} \frac{d\mathbf{u}^{(k)}[t;s]}{ds}$
  $\mathbf{r}^{(k)}[t;s+1] = \mathrm{ReLU}(\mathbf{u}^{(k)}[t;s+1])$
  $a^{(k)}[t;s+1] = a^{(k)}[t;s]) + \mu_a[s] \frac{da^{(k)}[t;s]}{ds}$
  $q^{(k)}[t;s+1] = \mathrm{ReLU}(a^{(k)}[t;s+1])$
  $s = s+1$, and adjust $\mu_{\mathbf{u}}[s]$ and $\mu_a[s]$ if necessary.
end while

6.7.2 Image Classification with Two Layered Networks

To evaluate the effectiveness of our approach, we compare it against other contrastive methods, namely EP [Scellier and Bengio, 2017] and CSM [Qin et al., 2021], as well

as explicit methods including PC [Whittington and Bogacz, 2017] and PC-Nudge [Millidge et al., 2023], when training multilayer perceptron (MLP) architectures.

Table 6.1: Test accuracy results (mean ± 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 [Qin et al., 2021], while the remaining results stem from our own simulations.

	MNIST	FashionMNIST	CIFAR10
CorInfoMax-$\mathcal{B}_{\infty,+}$ (Appendix 6.3.3)	97.62 ± 0.1	88.14 ± 0.3	51.86 ± 0.3
CorInfoMax-$\mathcal{B}_{1,+}$ (Appendix 6.5)	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

We investigate two distinct constraints on the activations of CorInfoMax Networks: (i) $\mathcal{B}_{\infty,+}$, which represents the nonnegative part of the unit hypercube, and (ii) $\mathcal{B}_{1,+} = \{\mathbf{r} : \mathbf{r} \geq 0, \|\mathbf{r}\|_1 \leq 1\}$, which denotes the nonnegative part of the unit ℓ_1-norm ball [Tatli and Erdogan, 2021b]. Table 6.1 displays the test accuracy results for each algorithm that are trained for 50 epochs, averaged over 10 realizations, along with their corresponding standard deviations. These results indicate that CorInfoMax networks can achieve comparable or superior performance compared to state-of-the-art methods on the selected tasks. Indeed, it is worth noting that the performance of CorInfoMax networks for the CIFAR10 dataset exhibits significant improvement and more stable convergence compared to other algorithms. Additional information regarding these experiments, as well as further experiments, can be found in the subsequent sections and Appendix C.

Figure 6.4 presents the comparison of the test accuracy performances for the CorInfoMax-$\mathcal{B}_{\infty,+}$ network, CSM, EP, PC, and PC-Nudge algorithms for the MNIST dataset. Similarly, Figure 6.5 showcases the comparison of the test accuracy performance of the CorInfoMax-$\mathcal{B}_{\infty,+}$ network, CSM, EP, PC, and PC-Nudge algorithms for the Fashion MNIST dataset. Furthermore, Figure 6.6 presents a comparison of the test accuracy performance of the CorInfoMax-$\mathcal{B}_{\infty,+}$ network, CSM, EP, PC, and PC-Nudge algorithms for the CIFAR10 dataset. The accuracy values are plotted as a function of training epochs. Additionally, the plots indicates the $\pm$ standard deviations for 10 different realization of the experiments. The accuracy convergence plots for CorInfoMax-$\mathcal{B}_{1,+}$ networks evaluated on MNIST, Fashion MNIST, and CIFAR10 can be found in Appendix C.1.

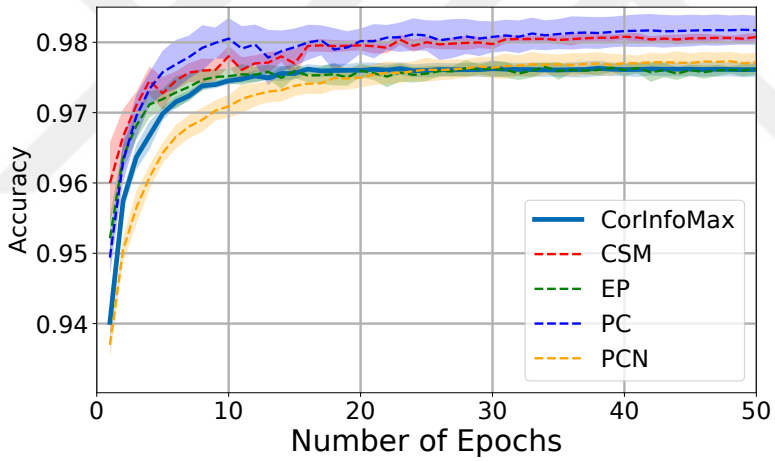


Figure 6.4: Test accuracy convergences of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks and other (CSM, EP, PC, and PC-Nudge) algorithms as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the MNIST dataset.

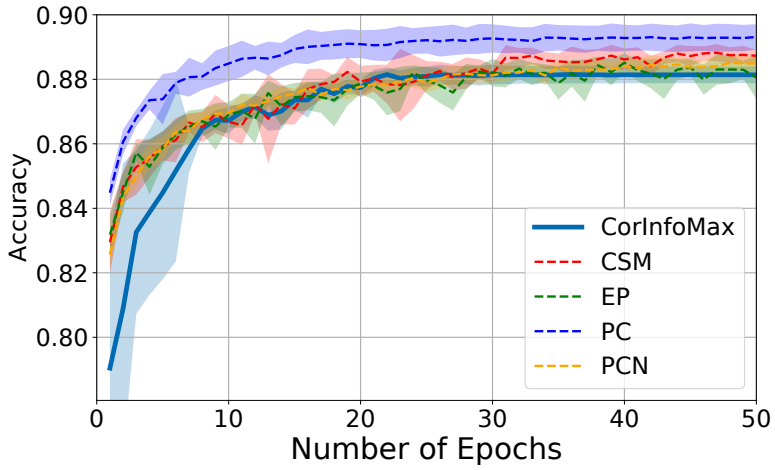


Figure 6.5: Test accuracy convergences of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks and other (CSM, EP, PC, and PC-Nudge) algorithms as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the Fashion MNIST dataset.

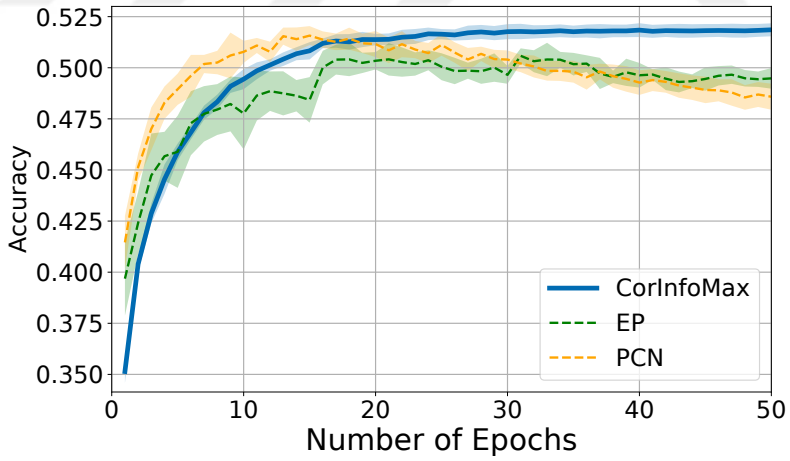


Figure 6.6: Test accuracy convergences of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks and other (CSM, EP, PC, and PC-Nudge) algorithms as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the CIFAR10 dataset.

6.7.3 Empirical Angle Between Forward and Backward Weights

To assess the degree of symmetry between the forward prediction synaptic weight matrix $\mathbf{W}_{ff}^{(k)}$ and the transpose of the backward prediction synaptic weight matrix $\boldsymbol{W}_{fb}^{(k)}$, we examine the cosine angle between them. This measure can be calculated using the method in [Lillicrap et al., 2016]:

$$\Theta^{(k)} = \arccos\left(\frac{\mathrm{Tr}\left(\mathbf{W}_{ff}^{(k)}\boldsymbol{W}_{fb}^{(k)}\right)}{\|\boldsymbol{W}_{ff}^{(k)}\|_F\|\boldsymbol{W}_{fb}^{(k)}\|_F}\right). \tag{6.46}$$

This metric acts as an indicator of alignment, where a cosine angle of $\Theta^{(k)} = 0$ degrees represents perfect symmetry, and $\Theta^{(k)} = 90$ degrees signifies orthogonality and, consequently, a significant level of asymmetry. To provide a more explicit demonstration of the disparity between the feedforward and feedback weights in the proposed CorInfoMax networks, we conducted a numerical assessment of this metric within the context of the experiments documented in Table 6.1. It is important to note that the feedforward and feedback weights were initialized independently with random values in these experiments.

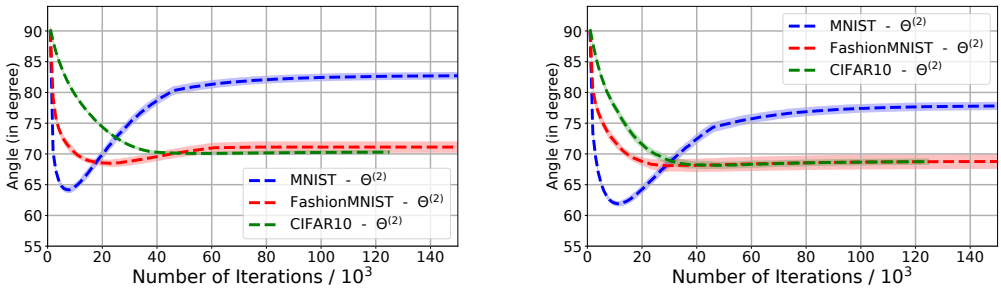


Figure 6.7: The angle between the feedforward and the transpose of the feedback weights between hidden and output layers (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) as a function of weight update iterations for a) CorInfoMax-$\mathcal{B}_{\infty,+}$, b) CorInfoMax-$\mathcal{B}_{1,+}$.

Figure 6.7 illustrates the alignment results obtained from the experiments conducted on the MNIST, Fashion MNIST, and CIFAR10 datasets, as a function of

iterations. More specifically, Figure 6.7a depicts the alignment results CorInfoMax-$\mathcal{B}_{\infty,+}$ networks, while Figure 6.7b depicts the alignment results CorInfoMax-$\mathcal{B}_{1,+}$ networks. As shown in the figure, the curves initially begin at 90 degrees, reflecting the independent random initialization of the weights. While the angle gradually decreases as the iterations proceed, the steady-state levels mostly high, indicating a significant level of asymmetry between the feedforward and feedback weights.

6.7.4 Three Layered CorInfoMax-$\mathcal{B}_{\infty,+}$ Network

In this section, we explore the extension of the CorInfoMax-$\mathcal{B}_{\infty,+}$ network illustrated in Figure 6.2 by incorporating an additional hidden layer.

The test accuracy results for the 3-layer CorInfoMax-$\mathcal{B}_{\infty,+}$ networks on the MNIST, CIFAR10, and CIFAR100 datasets are presented in Table 6.2.

Table 6.2: 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

Figure 6.8 depicts the test accuracy performance of the CorInfoMax-$\mathcal{B}_{\infty,+}$ network with two hidden layers as training epochs progress on the MNIST dataset. Similarly, Figure 6.9 showcases the test accuracy performance of the CorInfoMax-$\mathcal{B}_{\infty,+}$ network with two hidden layers for the CIFAR10 dataset. Additionally, Figure 6.10 illustrates the test accuracy performance of the CorInfoMax-$\mathcal{B}_{\infty,+}$ network with two hidden layers for the CIFAR100 dataset. Based on these figures, it can be inferred that the accuracy convergence behavior of CorInfoMax-$\mathcal{B}_{\infty,+}$ networks with two hidden layers remains predominantly stable.

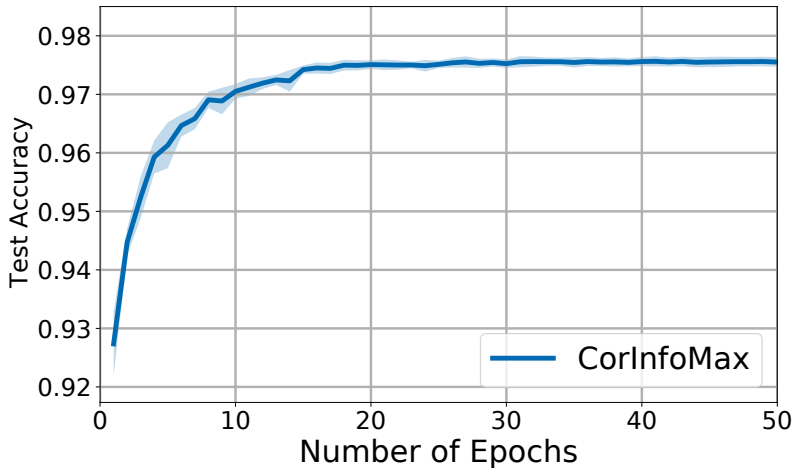


Figure 6.8: Test accuracy convergence of CorInfoMax-$\mathcal{B}_{\infty,+}$ network with two hidden layers as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the MNIST dataset.

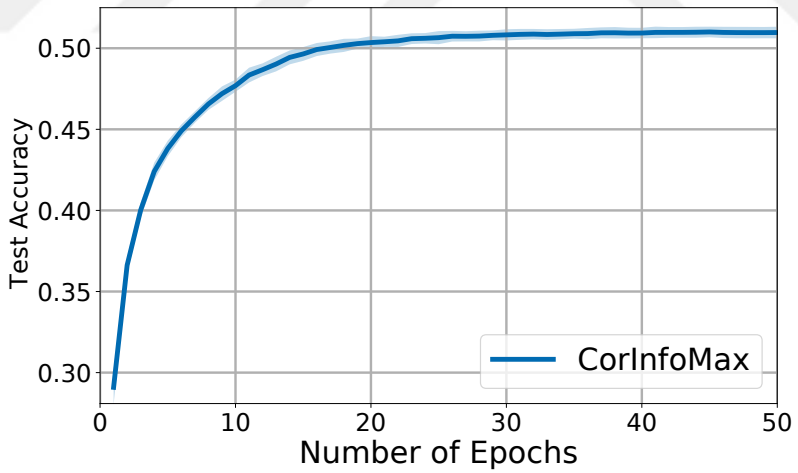


Figure 6.9: Test accuracy convergence of CorInfoMax-$\mathcal{B}_{\infty,+}$ network with two hidden layers as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the CIFAR10 dataset.

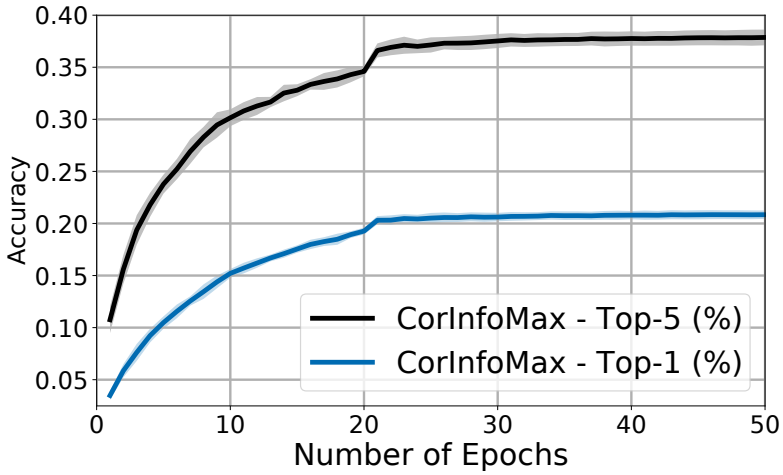
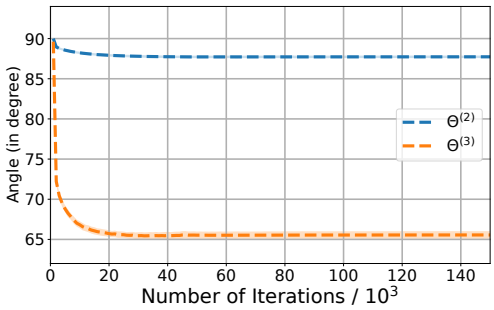
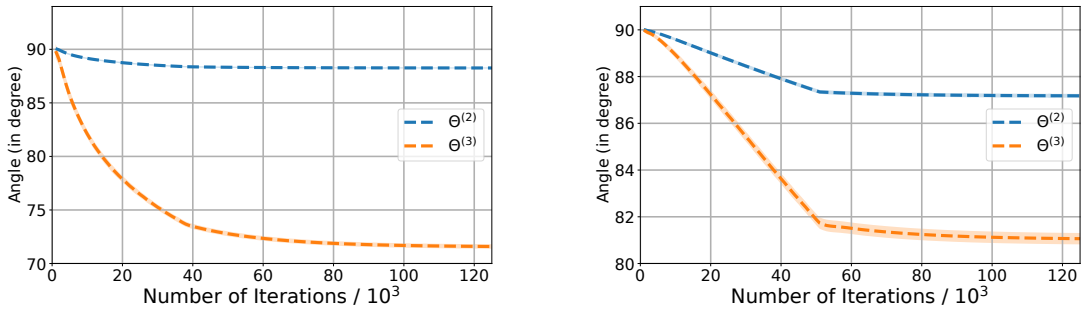


Figure 6.10: Top-1(%) and Top-5(%) test accuracy convergences of CorInfoMax-$\mathcal{B}_{\infty,+}$ network with two hidden layers as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the CIFAR100 dataset.

Figure 6.10 displays the results of the angle between the feedforward and the transpose of the feedback weights (as defined in Equation (6.46)) for the 3-layer CorInfoMax-$\mathcal{B}_{\infty,+}$ network on the MNIST (Figure 6.11a), CIFAR10 (Figure 6.10b), and CIFAR100 (Figure 6.10c) datasets. It is worth noting that two alignment angles are measured in this case, considering the two hidden layers of the CorInfoMax network. These results further support the observation that a significant degree of asymmetry between the feedforward and feedback weights persists, even when the corresponding network achieves a successful accuracy outcome.



(a)



(b) (c)

Figure 6.10: The angle between the feedforward and the transpose of the feedback weights (based on (6.46) 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.

6.8 Conclusion

In this chapter, we have introduced the Correlative Information Maximization (CorInfoMax) framework as a biologically plausible approach for constructing supervised neural network models. Our proposed method addresses the longstanding challenge of weight symmetry by offering a principled solution, resulting in asymmetric forward and backward prediction networks. Additionally, the CorInfoMax framework provides a normative approach for developing network models that incorporate multi-compartment pyramidal neuron models, aligning more closely with experimental findings on biological neural networks.

However, it is important to acknowledge a potential limitation of our framework, which is also shared by other supervised approaches. This limitation pertains to the need for model parameter search to enhance accuracy. We delve into a detailed discussion of this issue in Appendix C.3.

Chapter 7

CONCLUSION

In this thesis, we have explored novel approaches to matrix factorization, blind separation of correlated sources, and neural network training, with a main focus of biologically-plausible credit assignment and learning. Before we provided biologically-plausible learning algorithms for unsupervised and supervised tasks, we began by addressing the characterization of identifiable polytopes, an important problem encountered in the Polytopic Matrix Factorization framework. In Chapter 3, we proposed a computationally tractable approach using a graph automorphism algorithm, enhancing our understanding and applicability of the determinant maximization principle for blind source separation. Furthermore, this approach was utilized in Chapter 4 and 5, where we developed biologically-plausible neural networks capable of operating for feature-based polytopes that have been verified to be identifiable.

In Chapters 4 and 5, we developed biologically-plausible neural networks to address the blind source separation problem. In both chapters, we followed a normative principle by formulating a batched optimization problem, known to achieve perfect separation, in an online manner to obtain neural networks with local learning rules. In Chapter 4, we introduced a framework that unifies the determinant maximization principle with a weighted similarity matching objective for the blind separation of correlated sources. We illustrated the effectiveness of our approach by showcasing that cascading two weighted similarity matching constraints can enable arbitrary linear mappings from input to output, thus solving the blind source separation problem in combination with the determinant maximization principle. Our WSM-based Det-Max neural networks demonstrated effective separation of correlated synthetic and real-world data.

In Chapter 5, we formulated BSS neural networks based on the determinant

maximization principle from an information-theoretic perspective. The CorInfoMax framework allows for flexible modeling of complex latent structures by maximizing total correlative information from inputs to outputs while constraining the output vectors to a set where the source vectors are presumed to reside. Our online formulation of the CorInfoMax objective naturally led to neural networks with local learning rules while offering an infinite number of choices for the source domain. The numerical experiments demonstrated that the proposed method achieves superior results compared to other biologically-plausible algorithms while performing on par with batch algorithms.

Furthermore, in Chapter 6, we proposed correlative information maximization as an alternative normative approach to signal propagation in supervised biological neural networks. Our approach resolved the weight symmetry problem by leveraging alternative forms of the correlative mutual information objective. The resulting neural network structure mimics a biologically realistic network of multi-compartment pyramidal neurons, achieving successful learning on real-world image classification benchmarks.

Throughout the thesis, we conducted extensive numerical experiments, including ablation studies and assessments of limitations. To promote reproducibility, we made the codes and algorithms corresponding to each method available on GitHub, with links provided in the respective numerical experiment sections.

In conclusion, this thesis contributes to the advancement of matrix factorization, blind source separation, and neural network training by introducing biologically-plausible approaches and addressing important challenges. The proposed frameworks demonstrate their effectiveness and applicability through extensive experiments and comparisons, paving the way for further developments in these areas.

BIBLIOGRAPHY

[Abdolali and Gillis, 2021] Abdolali, M. and Gillis, N. (2021). Simplex-structured matrix factorization: Sparsity-based identifiability and provably correct algorithms. *SIAM Journal on Mathematics of Data Science*, 3(2):593–623.

[Asari et al., 2006] Asari, H., Pearlmutter, B. A., and Zador, A. M. (2006). Sparse representations for the cocktail party problem. *Journal of Neuroscience*, 26(28):7477–7490.

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

[Babatas and Erdogan, 2020] Babatas, E. and Erdogan, A. T. (2020). Time and frequency based sparse bounded component analysis algorithms for convolutive mixtures. *Signal Processing*, page 107590.

[Bahroun et al., 2021] Bahroun, Y., Chklovskii, D., and Sengupta, A. (2021). A normative and biologically plausible algorithm for independent component analysis. *Advances in Neural Information Processing Systems*, 34.

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

[Bee and Micheyl, 2008] Bee, M. A. and Micheyl, C. (2008). The cocktail party problem: what is it? how can it be solved? and why should animal behaviorists study it? *Journal of comparative psychology*, 122(3):235.

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

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

[Boyd et al., 2004] Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[Bozkurt and Erdogan, 2022] Bozkurt, B. and Erdogan, A. T. (2022). On identifiable polytope characterization for polytopic matrix factorization. *2022 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

[Bozkurt et al., 2023a] Bozkurt, B., İsfendiyaroğlu, A., Pehlevan, C., and Erdogan, A. T. (2023a). Correlative information maximization based biologically plausible neural networks for correlated source separation. In *The Eleventh International Conference on Learning Representations*.

[Bozkurt et al., 2022] Bozkurt, B., Pehlevan, C., and Erdogan, A. T. (2022). Biologically-plausible determinant maximization neural networks for blind separation of correlated sources. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

[Bozkurt et al., 2023b] Bozkurt, B., Pehlevan, C., and Erdogan, A. T. (2023b). Correlative information maximization: A biologically plausible approach to supervised deep neural networks without weight symmetry.

[Bremner et al., 2014] Bremner, D., Dutour Sikirić, M., Pasechnik, D. V., Rehn, T., and Schürmann, A. (2014). Computing symmetry groups of polyhedra. *LMS Journal of Computation and Mathematics*, 17(1):565–581.

[Bremner et al., 2009] Bremner, D., Sikiric, M. D., and Schürmann, A. (2009). Polyhedral representation conversion up to symmetries. In *CRM Proceedings & Lecture Notes: Polyhedral Computation*, volume 48, pages 45–72.

[Bronkhorst, 2000] Bronkhorst, A. W. (2000). The cocktail party phenomenon: A review of research on speech intelligibility in multiple-talker conditions. *Acta Acustica united with Acustica*, 86(1):117–128.

[Bödi and Herr, 3329] Bödi, R. and Herr, K. (2009, arXiv:0908.3329). Symmetries in linear and integer programs.

[Cardoso, 1989] Cardoso, J.-F. (1989). Source separation using higher order moments. In *International Conference on Acoustics, Speech, and Signal Processing,*, pages 2109–2112 vol.4.

[Chan et al., 2011] Chan, T.-H., Ma, W.-K., Ambikapathi, A., and Chi, C.-Y. (2011). A simplex volume maximization framework for hyperspectral endmember extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 49(11):4177–4193.

[Chen, 1984] Chen, J.-C. (1984). The nonnegative rank factorizations of nonnegative matrices. *Linear Algebra and Its Applications*, 62:207–217.

[Chen and Dongarra, 2005] Chen, Z. and Dongarra, J. J. (2005). Condition numbers of gaussian random matrices. *SIAM Journal on Matrix Analysis and Applications*, 27(3):603–620.

[Cichocki et al., 2009] Cichocki, A., Zdunek, R., Phan, A. H., and Amari, S.-i. (2009). *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons.

[Comon, 1994] Comon, P. (1994). Independent component analysis, a new concept? *Signal processing*, 36(3):287–314.

[Comon and Jutten, 2010] Comon, P. and Jutten, C. (2010). *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press.

[Cover, 1999] Cover, T. M. (1999). *Elements of information theory*. John Wiley & Sons.

[Crick, 1989] Crick, F. (1989). The recent excitement about neural networks. *Nature*, 337:129–132.

[Cruces, 2010] Cruces, S. (2010). Bounded component analysis of linear mixtures: A criterion of minimum convex perimeter. *IEEE Transactions on Signal Processing*, 58(4):2141–2154.

[Danielson and Borrelli, 2014] Danielson, C. and Borrelli, F. (2014). Identification of the symmetries of linear systems with polytopic constraints. In *2014 American Control Conference*, pages 4218–4223.

[Davies et al., 2021] Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G. A. F., Joshi, P., Plank, P., and Risbud, S. R. (2021). Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5):911–934.

[Dayan and Abbott, 2005] Dayan, P. and Abbott, L. F. (2005). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press.

[Donoho and Stodden, 2003] Donoho, D. and Stodden, V. (2003). When does non-negative matrix factorization give a correct decomposition into parts? *Advances in neural information processing systems*, 16:1141–1148.

[Donoho, 2006] Donoho, D. L. (2006). For most large underdetermined systems of equations, the minimal l1-norm near-solution approximates the sparsest near-solution. *Communications on Pure and Applied Mathematics*, 59(7):907–934.

[Eagleman et al., 2001] Eagleman, D. M., Coenen, O. J.-M. D., Mitsner, V., Bartol, T. M., Bell, A. J., and Sejnowski, T. J. (2001). Cerebellar glomeruli: Does limited extracellular calcium implement a sparse encoding strategy? In *Proceedings of the 8th Annual Joint Symposium on Neural Computation*.

[Elad, 2010] Elad, M. (2010). *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media.

[Elvira et al., 2016] Elvira, C., Chainais, P., and Dobigeon, N. (2016). Bayesian antisparse coding. *IEEE Transactions on Signal Processing*, 65(7):1660–1672.

[Erdogan, 2013] Erdogan, A. T. (2013). 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.

[Erdogan, 2022] Erdogan, A. T. (2022). 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. IEEE.

[Erdogan and Pehlevan, 2020] Erdogan, A. T. and Pehlevan, C. (2020). Blind bounded source separation using neural networks with local learning rules. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3812–3816. IEEE.

[Fraleigh and Katz, 2020] Fraleigh, J. B. and Katz, V. J. (2020). *A First Course in Abstract Algebra*. Pearson.

[Fu et al., 2018] Fu, X., Huang, K., and Sidiropoulos, N. D. (2018). On identifiability of nonnegative matrix factorization. *IEEE Signal Processing Letters*, 25(3):328–332.

[Fu et al., 2019] Fu, X., Huang, K., Sidiropoulos, N. D., and Ma, W.-K. (2019).

Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications. *IEEE Signal Process. Mag.*, 36(2):59–80.

[Fu et al., 2016] Fu, X., Huang, K., Yang, B., and Sidiropoulos, N. (2016). Robust volume minimization-based matrix factorization for remote sensing and document clustering. *IEEE Transactions on Signal Processing*, 64.

[Georgiev et al., 2005] Georgiev, P., Theis, F., and Cichocki, A. (2005). Sparse component analysis and blind source separation of underdetermined mixtures. *IEEE transactions on neural networks*, 16(4):992–996.

[Golkar et al., 2022] Golkar, S., Tesileanu, T., Bahroun, Y., Sengupta, A., and Chklovskii, D. (2022). Constrained predictive coding as a biologically plausible model of the cortical hierarchy. *Advances in Neural Information Processing Systems*, 35:14155–14169.

[Golumbic et al., 2013] Golumbic, E. M. Z., Ding, N., Bickel, S., Lakatos, P., Schevon, C. A., McKhann, G. M., Goodman, R. R., Emerson, R., Mehta, A. D., Simon, J. Z., et al. (2013). Mechanisms underlying selective neuronal tracking of attended speech at a "cocktail party". *Neuron*, 77(5):980–991.

[Grabska-Barwińska et al., 2017] Grabska-Barwińska, A., Barthelmé, S., Beck, J., Mainen, Z. F., Pouget, A., and Latham, P. E. (2017). A probabilistic approach to demixing odors. *Nature neuroscience*, 20(1):98–106.

[Gramfort et al., 2013] Gramfort, A., Luessi, M., Larson, E., Engemann, D. A., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L., and Hämäläinen, M. S. (2013). MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7(267):1–13.

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

[Grossberg, 1987b] Grossberg, S. (1987b). Competitive learning: From interactive activation to adaptive resonance. *Cogn. Sci.*, 11:23–63.

[Grünbaum et al., 1967] Grünbaum, B., Klee, V., Perles, M. A., and Shephard, G. C. (1967). *Convex polytopes*, volume 16. Springer.

[Guerguiev et al., 2017] Guerguiev, J., Lillicrap, T. P., and Richards, B. A. (2017). Towards deep learning with segregated dendrites. *Elife*, 6:e22901.

[Hjelm et al., 2019] Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2019). Learning deep representations by mutual information estimation and maximization. *International Conference on Learning Representations (ICLR)*.

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

[Hyvärinen et al., 2001] Hyvärinen, A., Karhunen, J., and Oja, E. (2001). Independent component analysis, adaptive and learning systems for signal processing, communications, and control. *John Wiley & Sons, Inc*, 1:11–14.

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

[Isomura et al., 2015] Isomura, T., Kotani, K., and Jimbo, Y. (2015). Cultured cortical neurons can perform blind source separation according to the free-energy principle. *PLoS Comput Biol*, 11(12):e1004643.

[Isomura and Toyoizumi, 2018] Isomura, T. and Toyoizumi, T. (2018). Error-gated hebbian rule: A local learning rule for principal and independent component analysis. *Scientific reports*, 8(1):1–11.

[Junttila and Kaski, 2007] Junttila, T. and Kaski, P. (2007). Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, page 135–149, USA. Society for Industrial and Applied Mathematics.

[Kailath et al., 2000] Kailath, T., Sayed, A. H., and Hassibi, B. (2000). *Linear estimation*. Prentice-Hall information and system sciences series. Prentice Hall.

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

[Kreutz-Delgado et al., 2003] Kreutz-Delgado, K., Murray, J. F., Rao, B. D., Engan, K., Lee, T.-W., and Sejnowski, T. J. (2003). Dictionary learning algorithms for sparse representation. *Neural computation*, 15(2):349–396.

[Krishnamurthy et al., 2017] Krishnamurthy, K., Hermundstad, A. M., Mora, T., Walczak, A. M., and Balasubramanian, V. (2017). Disorder and the neural representation of complex odors: smelling in the real world. *arXiv preprint arXiv:1707.01962*.

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

[Kunin et al., 2020] Kunin, D., Nayebi, A., Sagastuy-Brena, J., Ganguli, S., Bloom, J. M., and Yamins, D. L. K. (2020). Two routes to scalable credit assignment without weight symmetry. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

[Laborieux et al., 2021] Laborieux, A., Ernoult, M., Scellier, B., Bengio, Y., Grollier, J., and Querlioz, D. (2021). Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in Neuroscience*, 15:633674.

[Laborieux and Zenke, 2022] Laborieux, A. and Zenke, F. (2022). Holomorphic equilibrium propagation computes exact gradients through finite size oscillations. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

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

[LeCun and Cortes, 2010] LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/.

[Lewicki, 2002] Lewicki, M. S. (2002). Efficient coding of natural sounds. *Nature neuroscience*, 5(4):356–363.

[Li et al., 2004] Li, Y., Cichocki, A., and Amari, S.-i. (2004). Analysis of sparse representation and blind source separation. *Neural computation*, 16(6):1193–1234.

[Lillicrap et al., 2016] Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276.

[Lin et al., 2015] Lin, C.-H., Ma, W.-K., Li, W.-C., Chi, C.-Y., and Ambikapathi, A. (2015). Identifiability of the simplex volume minimization criterion for blind hyperspectral unmixing: The no-pure-pixel case. *IEEE Transactions on Geoscience and Remote Sensing*, 53(10):5530–5546.

[Lin et al., 2018] Lin, C.-H., Wu, R., Ma, W.-K., Chi, C.-Y., and Wang, Y. (2018). Maximum volume inscribed ellipsoid: A new simplex-structured matrix factorization framework via facet enumeration and convex optimization. *SIAM Journal on Imaging Sciences*, 11(2):1651–1679.

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

[Lipshutz et al., 2022] Lipshutz, D., Pehlevan, C., and Chklovskii, D. B. (2022). Biologically plausible single-layer networks for nonnegative independent component analysis. *Biological Cybernetics*, pages 1–12.

[Lübke et al., 1996] Lübke, J., Markram, H., Frotscher, M., and Sakmann, B. (1996). 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.

[Makeig et al., 1996] Makeig, S., Bell, A. J., Jung, T.-P., and Sejnowski, T. J. (1996). Independent component analysis of electroencephalographic data. In *Advances in neural information processing systems*, pages 145–151.

[Margot, 2010] Margot, F. (2010). Symmetry in integer linear programming. In Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., Pulleyblank, W. R., Reinelt, G., Rinaldi, G., and Wolsey, L. A., editors, *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, chapter 17, pages 647–686. Springer Berlin Heidelberg, Berlin, Heidelberg.

[McDermott, 2009] McDermott, J. H. (2009). The cocktail party problem. *Current Biology*, 19(22):R1024–R1027.

[McKay and Piperno, 2014] McKay, B. D. and Piperno, A. (2014). Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112.

[Mesgarani and Chang, 2012] Mesgarani, N. and Chang, E. F. (2012). Selective cortical representation of attended speaker in multi-talker speech perception. *Nature*, 485(7397):233–236.

[Millidge et al., 2023] Millidge, B., Song, Y., Salvatori, T., Lukasiewicz, T., and Bogacz, R. (2023). Backpropagation at the infinitesimal inference limit of energy-based models: Unifying predictive coding, equilibrium propagation, and con-

trastive hebbian learning. In *The Eleventh International Conference on Learning Representations*.

[Minden et al., 2018] Minden, V., Pehlevan, C., and Chklovskii, D. B. (2018). Biologically plausible online principal component analysis without recurrent neural dynamics. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 104–111. IEEE.

[Mukamel et al., 2009] Mukamel, E. A., Nimmerjahn, A., and Schnitzer, M. J. (2009). Automated analysis of cellular signals from large-scale calcium imaging data. *Neuron*, 63(6):747–760.

[Narayan et al., 2007] Narayan, R., Best, V., Ozmeral, E., McClaine, E., Dent, M., Shinn-Cunningham, B., and Sen, K. (2007). Cortical interference effects in the cocktail party problem. *Nature neuroscience*, 10(12):1601–1607.

[Olshausen and Field, 1996] Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609.

[Olshausen and Field, 1997] Olshausen, B. A. and Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325.

[Ozsoy et al., 2022] Ozsoy, S., Hamdan, S., Arik, S. O., Yuret, D., and Erdogan, A. T. (2022). Self-supervised learning with an information maximization criterion. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

[Paatero and Tapper, 1994] Paatero, P. and Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126.

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

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[Pehlevan, 2019] Pehlevan, C. (2019). A spiking neural network with local learning rules derived from nonnegative similarity matching. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7958–7962. IEEE.

[Pehlevan and Chklovskii, 2015] Pehlevan, C. and Chklovskii, D. B. (2015). A normative theory of adaptive dimensionality reduction in neural networks supplementary information. *Proceedings of Advances in Neural Information Processing Systems 28 (NIPS 2015)*.

[Pehlevan and Chklovskii, 2019] Pehlevan, C. and Chklovskii, D. B. (2019). Neuroscience-inspired online unsupervised learning algorithms: Artificial neural networks. *IEEE Signal Processing Magazine*, 36(6):88–96.

[Pehlevan et al., 2017a] Pehlevan, C., Genkin, A., and Chklovskii, D. B. (2017a). A clustering neural network model of insect olfaction. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pages 593–600. IEEE.

[Pehlevan et al., 2017b] Pehlevan, C., Mohan, S., and Chklovskii, D. B. (2017b). Blind nonnegative source separation using biological neural networks. *Neural computation*, 29(11):2925–2954.

[Petreanu et al., 2009] Petreanu, L., Mao, T., Sternson, S. M., and Svoboda, K.

(2009). The subcellular organization of neocortical excitatory connections. *Nature*, 457(7233):1142–1145.

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

[Qin et al., 2021] Qin, S., Mudur, N., and Pehlevan, C. (2021). Contrastive Similarity Matching for Supervised Learning. *Neural Computation*, 33(5):1300–1328.

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

[Richards and Lillicrap, 2019] Richards, B. A. and Lillicrap, T. P. (2019). Dendritic solutions to the credit assignment problem. *Current opinion in neurobiology*, 54:28–36.

[Richards et al., 2019] Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R. P., , Ganguli, S., Gillon, C. J., Hafner, D., Kepecs, A., Kriegeskorte, N., Latham, P., Lindsay, G. W., Miller, K. D., Naud, R., Pack, C. C., Poirazi, P., Roelfsema, P., Sacramento, J., Saxe, A., Scellier, B., Schapiro, A. C., Senn, W., Wayne, G., Yamins, D., Zenke, F., Zylberberg, J., Therien, D., and Kording, K. P. (2019). A deep learning framework for neuroscience. *Nature Neuroscience*, 22:1761–1770.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408.

[Rozell et al., 2007] Rozell, C., Johnson, D., Baraniuk, R., and Olshausen, B. (2007). Locally competitive algorithms for sparse approximation. In *2007 IEEE International Conference on Image Processing*, volume 4, pages IV–169. IEEE.

[Rozell et al., 2008] Rozell, C. J., Johnson, D. H., Baraniuk, R. G., and Olshausen, B. A. (2008). Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, 20(10):2526–2563.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

[Sacramento et al., 2018] Sacramento, J., Ponte Costa, R., Bengio, Y., and Senn, W. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31.

[Sayed, 2003] Sayed, A. H. (2003). *Fundamentals of adaptive filtering*. John Wiley & Sons.

[Scellier and Bengio, 2017] Scellier, B. and Bengio, Y. (2017). Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24.

[Scellier et al., 2018] Scellier, B., Goyal, A., Binas, J., Mesnard, T., and Bengio, Y. (2018). Generalization of equilibrium propagation to vector field dynamics.

[Schachtner et al., 2011] Schachtner, R., Pöppel, G., and Lang, E. W. (2011). Towards unique solutions of non-negative matrix factorization problems by a determinant criterion. *Digital Signal Processing*, 21(4):528–534.

[Sharma et al., 2000] Sharma, J., Angelucci, A., and Sur, M. (2000). Induction of visual orientation modules in auditory cortex. *Nature*, 404(6780):841–847.

[Simsek and Erdogan, 2019] Simsek, B. and Erdogan, A. T. (2019). Online bounded component analysis: A simple recurrent neural network with local update rule for unsupervised separation of dependent and independent sources. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 1639–1643.

[Singh et al., 2021] Singh, V., Tchernookov, M., and Balasubramanian, V. (2021). What the odor is not: Estimation by elimination. *Physical Review E*, 104(2):024415.

[Smith and Lewicki, 2006] Smith, E. C. and Lewicki, M. S. (2006). Efficient auditory coding. *Nature*, 439(7079):978–982.

[Song et al., 2020] Song, Y., Lukasiewicz, T., Xu, Z., and Bogacz, R. (2020). Can the brain do backpropagation?—exact implementation of backpropagation in predictive coding networks. *Advances in neural information processing systems*, 33:22566–22579.

[Studer et al., 2014] Studer, C., Goldstein, T., Yin, W., and Baraniuk, R. G. (2014). Democratic representations. *arXiv preprint arXiv:1401.3420*.

[Tatli and Erdogan, 2021a] Tatli, G. and Erdogan, A. T. (2021a). Generalized polytopic matrix factorization. *2021 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

[Tatli and Erdogan, 2021b] Tatli, G. and Erdogan, A. T. (2021b). Polytopic matrix factorization: Determinant maximization based criterion and identifiability. *IEEE Transactions on Signal Processing*, 69:5431–5447.

[The Sage Developers, 2021] The Sage Developers (2021). *SageMath, the Sage Mathematics Software System (Version 9.3)*. `https://www.sagemath.org`.

[Träuble et al., 2021] Träuble, F., Creager, E., Kilbertus, N., Locatello, F., Dittadi, A., Goyal, A., Schölkopf, B., and Bauer, S. (2021). On disentangled representations learned from correlated data. In *International Conference on Machine Learning*, pages 10401–10412. PMLR.

[Turrigiano, 1999] Turrigiano, G. G. (1999). Homeostatic plasticity in neuronal networks: the more things change, the more they stay the same. *Trends in neurosciences*, 22(5):221–227.

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

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

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

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

[Whittington et al., 2023] Whittington, J. C. R., Dorrell, W., Ganguli, S., and Behrens, T. (2023). Disentanglement with biological constraints: A theory of functional cell types. In *The Eleventh International Conference on Learning Representations*.

[Winter, 3483] Winter, M. (2021, arXiv:2108.13483). Capturing polytopal symmetries by coloring the edge-graph.

[Xiao et al., 2017] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[Xiao et al., 2019] Xiao, W., Chen, H., Liao, Q., and Poggio, T. (2019). Biologically-plausible learning algorithms can scale to large datasets. In *International Conference on Learning Representations*.

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

[Zador et al., 2022] Zador, A., Richards, B., Ölveczky, B., Escola, S., Bengio, Y., Boahen, K., Botvinick, M., Chklovskii, D., Churchland, A., Clopath, C., DiCarlo, J., Ganguli, S., Hawkins, J., Koerding, K., Koulakov, A., LeCun, Y., Lillicrap, T., Marblestone, A., Olshausen, B., Pouget, A., Savin, C., Sejnowski, T., Simoncelli, E., Solla, S., Sussillo, D., Tolias, A. S., and Tsao, D. (2022). Toward next-generation artificial intelligence: Catalyzing the neuroai revolution.

[Zhang and Sharpee, 2016] Zhang, Y. and Sharpee, T. O. (2016). A robust feedforward model of the olfactory system. *PLoS computational biology*, 12(4):e1004850.

[Ziegler, 1995] Ziegler, G. M. (1995). *Lectures on polytopes*. Springer-Verlag, New York.

Appendix A

SUPPLEMENTARY ON DET-MAX WSM NETWORKS

A.1 The Simplification of the Similarity Matching Cost Functions

In this section, we aim to simplify the similarity matching cost functions J_1 and J_2 presented in Section 4.5.1 by retaining only the quadratic terms that are relevant for online optimization with respect to $\mathbf{h}(t)$ and $\mathbf{y}(t)$.

Using the matrix partitions $\boldsymbol{\mathcal{X}}(t) = \begin{bmatrix} \gamma\boldsymbol{\mathcal{X}}(t-1) & \mathbf{x}(t) \end{bmatrix}$ and $\boldsymbol{\mathcal{H}}(t) = \begin{bmatrix} \gamma\boldsymbol{\mathcal{H}}(t-1) & \mathbf{h}(t) \end{bmatrix}$, $J_1(\mathbf{H}(t), \mathbf{D}_1(t))$ can be written more explicitly as

$$
\begin{aligned}
&J_1(\mathbf{H}(t), \mathbf{D}_1(t)) \\
&= \frac{\gamma^2}{2\tau^2} \left\| \begin{bmatrix} \gamma\boldsymbol{\mathcal{X}}(t-1)^T\boldsymbol{\mathcal{X}}(t-1) - \gamma\boldsymbol{\mathcal{H}}(t-1)^T\mathbf{D}_1(t)\boldsymbol{\mathcal{H}}(t-1) & \boldsymbol{\mathcal{X}}(t-1)^T\mathbf{x}(t) - \boldsymbol{\mathcal{H}}(t-1)^T\mathbf{D}_1(t)\mathbf{h}(t) \\ \mathbf{x}(t)^T\boldsymbol{\mathcal{X}}(t-1) - \mathbf{h}(t)^T\mathbf{D}_1(t)\boldsymbol{\mathcal{H}}(t-1) & \frac{\|\mathbf{x}(t)\|_2^2 - \|\mathbf{h}(t)\|_{\mathbf{D}_1(t)}^2}{\gamma} \end{bmatrix} \right\|_F^2 \\
&= \frac{\gamma^4}{\tau^2}\|\boldsymbol{\mathcal{X}}(t-1)^T\boldsymbol{\mathcal{X}}(t-1)\|_F^2 + \frac{\gamma^4}{\tau^2}\|\boldsymbol{\mathcal{H}}(t-1)^T\mathbf{D}_1(t)\boldsymbol{\mathcal{H}}(t-1)\|_F^2 \\
&- \frac{2\gamma^4}{\tau^2}\mathrm{Tr}(\boldsymbol{\mathcal{H}}(t-1)^T\mathbf{D}_1(t)\boldsymbol{\mathcal{H}}(t-1)\boldsymbol{\mathcal{X}}(t-1)^T\boldsymbol{\mathcal{X}}(t-1)) \\
&+ \frac{2\gamma^2}{\tau^2}\|\mathbf{x}(t)^T\boldsymbol{\mathcal{X}}(t-1)\|_F^2 + \frac{2\gamma^2}{\tau^2}\|\mathbf{h}(t)^T\mathbf{D}_1(t)\boldsymbol{\mathcal{H}}(t-1)\|_F^2 \\
&- \frac{4\gamma^2}{\tau^2}\mathbf{h}(t)^T\mathbf{D}_1(t)\boldsymbol{\mathcal{H}}(t-1)\boldsymbol{\mathcal{X}}(t-1)^T\mathbf{x}(t) \\
&+ \frac{1}{\tau^2}(\|\mathbf{x}(t)\|^4 + \|\mathbf{h}(t)\|_{\mathbf{D}_1(t)}^4 - 2\|\mathbf{x}(t)\|^2\|\mathbf{h}(t)\|_{\mathbf{D}_1(t)}^2).
\end{aligned}
$$

By focusing on the relevant part of the cost function for online optimization with respect to $\mathbf{h}(t)$, scaling it with τ/γ^2, and neglecting the small final term, we can derive the effective online cost function corresponding to J_1 as follows:

$$
c_1(\mathbf{h}(t)) \;=\; 2\mathbf{h}(t)^T\mathbf{D}_1\mathbf{M}_H(t)\mathbf{D}_1(t)\mathbf{h}(t) - 4\mathbf{h}(t)^T\mathbf{D}_1(t)\mathbf{W}_{HX}(t)\mathbf{x}(t),
$$

where

$$\mathbf{M}_H(t) = \frac{1}{\tau}\boldsymbol{\mathcal{H}}(t-1)\boldsymbol{\mathcal{H}}(t-1)^T = \frac{1}{\tau}\sum_{k=1}^{t-1}(\gamma^2)^{t-1-k}\mathbf{h}(k)\mathbf{h}(k)^T \tag{A.1}$$

$$\mathbf{W}_{HX}(t) = \frac{1}{\tau}\boldsymbol{\mathcal{H}}(t-1)\boldsymbol{\mathcal{X}}(t-1)^T = \frac{1}{\tau}\sum_{k=1}^{t-1}(\gamma^2)^{t-1-k}\mathbf{h}(k)\mathbf{x}(k)^T. \tag{A.2}$$

We apply the same procedure to $J_2(\mathbf{H}(t), \mathbf{D}_2(t), \mathbf{Y}(t))$ as follows:

$$\begin{aligned}
&J_2(\mathbf{H}(t), \mathbf{D}_2(t), \mathbf{Y}(t)) \\
&= \frac{\gamma^2}{2\tau^2}\left\|\begin{bmatrix} \gamma\boldsymbol{\mathcal{H}}(t-1)^T\boldsymbol{\mathcal{H}}(t-1) - \gamma\boldsymbol{\mathcal{Y}}(t-1)^T\mathbf{D}_2(t)\boldsymbol{\mathcal{Y}}(t-1) & \boldsymbol{\mathcal{H}}(t-1)^T\mathbf{h}(t) - \boldsymbol{\mathcal{Y}}(t-1)^T\mathbf{D}_2(t)\mathbf{y}(t) \\ \mathbf{h}(t)^T\boldsymbol{\mathcal{H}}(t-1) - \mathbf{y}(t)^T\mathbf{D}_2(t)\boldsymbol{\mathcal{Y}}(t-1) & \frac{\|\mathbf{h}(t)\|_2^2 - \|\mathbf{y}(t)\|_{\mathbf{D}_2(t)}^2}{\gamma} \end{bmatrix}\right\|_F^2 \\
&= \frac{\gamma^4}{\tau^2}\|\boldsymbol{\mathcal{H}}(t-1)^T\boldsymbol{\mathcal{H}}(t-1)\|_F^2 + \frac{\gamma^4}{\tau^2}\|\boldsymbol{\mathcal{Y}}(t-1)^T\mathbf{D}_2(t)\boldsymbol{\mathcal{Y}}(t-1)\|_F^2 \\
&-\frac{2\gamma^4}{\tau^2}\mathrm{Tr}(\boldsymbol{\mathcal{Y}}(t-1)^T\mathbf{D}_2(t)\boldsymbol{\mathcal{Y}}(t-1)\boldsymbol{\mathcal{H}}(t-1)^T\boldsymbol{\mathcal{H}}(t-1)) \\
&+\frac{2\gamma^2}{\tau^2}\|\mathbf{h}(t)^T\boldsymbol{\mathcal{H}}(t-1)\|_F^2 + \frac{2\gamma^2}{\tau^2}\|\mathbf{y}(t)^T\mathbf{D}_2(t)\boldsymbol{\mathcal{Y}}(t-1)\|_F^2 \\
&-\frac{4\gamma^2}{\tau^2}\mathbf{y}(t)^T\mathbf{D}_2(t)\boldsymbol{\mathcal{Y}}(t-1)\boldsymbol{\mathcal{H}}(t-1)^T\mathbf{h}(t) \\
&+\frac{1}{\tau^2}(\|\mathbf{h}(t)\|^4 + \|\mathbf{y}(t)\|_{\mathbf{D}_2(t)}^4 - 2\|\mathbf{h}(t)\|^2\|\mathbf{y}(t)\|_{\mathbf{D}_2(t)}^2).
\end{aligned}$$

Similar to the simplification of J_1, we can simplify the part of the J_2 cost function that is dependent on $\mathbf{h}(t)$ and $\mathbf{y}(t)$ as follows:

$$\begin{aligned}
c_2(\mathbf{h}(t), \mathbf{y}(t)) = {} & 2\mathbf{y}(t)^T\mathbf{D}_2(t)\mathbf{M}_Y(t)\mathbf{D}_2(t)\mathbf{y}(t) - 4\mathbf{y}(t)^T\mathbf{D}_2(t)\mathbf{W}_{YH}(t)\mathbf{h}(t) \\
& + 2\mathbf{h}(t)^T\mathbf{M}_H(t)\mathbf{h}(t),
\end{aligned}$$

where

$$\mathbf{W}_{YH}(t) = \frac{1}{\tau}\boldsymbol{\mathcal{Y}}(t-1)\boldsymbol{\mathcal{H}}(t-1)^T = \frac{1}{\tau}\sum_{k=1}^{t-1}(\gamma^2)^{t-1-k}\mathbf{y}(k)\mathbf{h}(k)^T, \tag{A.3}$$

$$\mathbf{M}_Y(t) = \frac{1}{\tau}\boldsymbol{\mathcal{Y}}(t-1)\boldsymbol{\mathcal{Y}}(t-1)^T = \frac{1}{\tau}\sum_{k=1}^{t-1}(\gamma^2)^{t-1-k}\mathbf{y}(k)\mathbf{y}(k)^T. \tag{A.4}$$

As a result of the simplifications made to the cost functions J_1 and J_2, we can express the effective online cost function $\mathcal{J}$ corresponding to $\mathbf{h}(t)$ and $\mathbf{y}(t)$ as

$$C(\mathbf{h}(t), \mathbf{y}(t)) = \beta c_1(\mathbf{h}(t)) + (1-\beta)c_2(\mathbf{h}(t), \mathbf{y}(t)). \tag{A.5}$$

A.2 Derivatives of the WSM Cost Function

In this section, we present the expressions for the gradients of the online WSM cost function $\mathcal{J}$ in (4.5) with respect to $\mathbf{h}(t)$ and $\mathbf{y}(t)$. We utilize the simplified version of $\mathcal{J}$, denoted as $C(\mathbf{h}(t), \mathbf{y}(t))$ in equation A.5, which was derived in Section A.1. The gradients are given by:

- The (scaled) gradient with respect to $\mathbf{h}(t)$:

$$\begin{aligned}\frac{1}{4}\nabla_{\mathbf{h}(t)}\mathcal{J}(\mathbf{H}(t), \mathbf{D}_1(t), \mathbf{D}_2(t), \mathbf{Y}(t)) &= \frac{1}{4}\nabla_{\mathbf{h}(t)}C(\mathbf{h}(t), \mathbf{y}(t)) \\ &= ((1-\beta)\mathbf{M}_H(t) + \beta\mathbf{D}_1(t)\mathbf{M}_H(t)\mathbf{D}_1(t))\mathbf{h}(t) \\ &\quad -\beta\mathbf{D}_1(t)\mathbf{W}_{HX}(t)\mathbf{x}(t) - (1-\beta)\mathbf{W}_{YH}(t)^T\mathbf{D}_2(t)\mathbf{y}(t). \quad \text{(A.6)}\end{aligned}$$

  We apply the decomposition $\mathbf{M}_H(t) = \bar{\mathbf{M}}_H(t) + \mathbf{\Gamma}_H(t)$, where

$$\mathbf{\Gamma}_H(t) \;=\; \text{diag}(\mathbf{M}_{H11}(t), \mathbf{M}_{H22}(t), \ldots, \mathbf{M}_{Hdd}(t)).$$

  Thus, we can rewrite the gradient expression in (A.6) as

$$\begin{aligned}\frac{1}{4}\nabla_{\boldsymbol{h}(t)}C(\mathbf{h}(t), \mathbf{y}(t)) = \mathbf{v}(t) &+ ((1-\beta)\bar{\mathbf{M}}_H(t) + \beta\mathbf{D}_1(t)\bar{\mathbf{M}}_H(t)\mathbf{D}_1(t))\mathbf{h}(t) \\ &- \beta\mathbf{D}_1(t)\mathbf{W}_{HX}(t)\mathbf{x}(t) - (1-\beta)\mathbf{W}_{YH}(t)^T\mathbf{D}_2(t)\mathbf{y}(t). \quad \text{(A.7)}\end{aligned}$$

  Note that we used the following substitution in (A.7),

$$\mathbf{v}(t) = ((1-\beta)\mathbf{\Gamma}_H(t) + \beta\mathbf{D}_1(t)\mathbf{\Gamma}_H(t)\mathbf{D}_1(t)))\boldsymbol{h}(t). \quad \text{(A.8)}$$

- The scaled gradient with respect to $\mathbf{y}(t)$:

$$\frac{1}{4}\nabla_{\mathbf{y}(t)}C(\mathbf{h}(t), \mathbf{y}(t)) = (1-\beta)(-\mathbf{D}_2(t)\mathbf{W}_{YH}(t)\mathbf{h}(t) + \mathbf{D}_2(t)\mathbf{M}_Y(t)\mathbf{D}_2(t)\mathbf{y}(t)).$$

  Since $\mathbf{D}_2(t)$ is positive, we note that

$$-\frac{1}{4(1-\beta)}\mathbf{D}_2(t)^{-1}\nabla_{\mathbf{y}(t)}\mathcal{J}(\mathbf{h}(t), \mathbf{y}(t)) = \mathbf{W}_{YH}(t)\mathbf{h}(t) - \mathbf{M}_Y(t)\mathbf{D}_2(t)\mathbf{y}(t) \quad \text{(A.9)}$$

  is a descent direction. Moreover, using the decomposition $\mathbf{M}_Y(t) = \bar{\mathbf{M}}_Y(t) + \mathbf{\Gamma}_Y(t)$, where

$$\mathbf{\Gamma}_Y(t) \;=\; \text{diag}(\mathbf{M}_{Y11}(t), \mathbf{M}_{Y22}(t), \ldots, \mathbf{M}_{Ydd}(t)),$$

we can rewrite the the descent direction in (A.9) as

$$-\frac{1}{4(1-\beta)}\mathbf{D}_2(t)^{-1}\nabla_{\mathbf{y}(t)}C(\mathbf{h}(t),\mathbf{y}(t)) = -\mathbf{u}(t) + \mathbf{W}_{YH}(t)\mathbf{h}(t) - \bar{\mathbf{M}}_Y(t)\mathbf{D}_2(t)\mathbf{y}(t), \tag{A.10}$$

where we used the substitution

$$\mathbf{u}(t) = \mathbf{\Gamma}_Y(t)\mathbf{D}_2(t)\mathbf{y}(t). \tag{A.11}$$

- The derivative with respect to $D_{1,ii}(t)$:

$$\begin{aligned}
&\frac{\partial \mathcal{J}(\mathbf{h}(t),\mathbf{y}(t),\mathbf{D}_1(t),\mathbf{D}_2(t))}{\partial D_{1,ii}(t)} \\
&= \lambda_{SM}\beta\mathrm{Tr}((\boldsymbol{\mathcal{H}}(t)^T\mathbf{E}_{ii}\boldsymbol{\mathcal{H}}(t))^T(\boldsymbol{\mathcal{H}}(t)^T\mathbf{D}_1(t)\boldsymbol{\mathcal{H}}(t) - \boldsymbol{\mathcal{X}}(t)^T\boldsymbol{\mathcal{X}}(t))) \\
&+ \frac{1-\lambda_{SM}}{D_{1,ii}(t)} \\
&= \lambda_{SM}\beta\mathrm{Tr}(\boldsymbol{\mathcal{H}}(t)_{i,:}^T\boldsymbol{\mathcal{H}}(t)_{i,:}(\boldsymbol{\mathcal{H}}(t)^T\mathbf{D}_1(t)\boldsymbol{\mathcal{H}}(t) - \boldsymbol{\mathcal{X}}(t)^T\boldsymbol{\mathcal{X}}(t))) \\
&+ \frac{1-\lambda_{SM}}{D_{1,ii}(t)} \\
&= \lambda_{SM}\beta(\boldsymbol{\mathcal{H}}(t)_{i,:}\boldsymbol{\mathcal{H}}(t)^T\mathbf{D}_1(t)\boldsymbol{\mathcal{H}}(t)\boldsymbol{\mathcal{H}}(t)_{i,:}^T - \boldsymbol{\mathcal{H}}(t)_{i,:}\boldsymbol{\mathcal{X}}(t)^T\boldsymbol{\mathcal{X}}(t))\boldsymbol{\mathcal{H}}(t)_{i,:}^T) \\
&+ \frac{1-\lambda_{SM}}{D_{1,ii}(t)} \\
&= \lambda_{SM}\beta(\|\mathbf{M}_{Hi,:}\|^2_{\mathbf{D}_1(t)} - \|\mathbf{W}_{HXi,:}\|_2^2) + \frac{1-\lambda_{SM}}{D_{1,ii}(t)}.
\end{aligned} \tag{A.12}$$

- The derivative with respect to $D_{2,ii}(t)$:

$$\begin{aligned}
&\frac{\partial \mathcal{J}(\mathbf{h}(t), \mathbf{y}(t), \mathbf{D}_1(t), \mathbf{D}_2(t))}{\partial D_{2,ii}(t)} \\
&= \lambda_{SM}(1-\beta)\mathrm{Tr}((\boldsymbol{\mathcal{Y}}(t)^T\mathbf{E}_{ii}\boldsymbol{\mathcal{Y}}(t))^T(\boldsymbol{\mathcal{Y}}(t)^T\mathbf{D}_2(t)\boldsymbol{\mathcal{Y}}(t) - \boldsymbol{\mathcal{H}}(t)^T\boldsymbol{\mathcal{H}}(t))) \\
&+ \frac{1-\lambda_{SM}}{D_{2,ii}(t)} \\
&= \lambda_{SM}(1-\beta)\mathrm{Tr}(\boldsymbol{\mathcal{Y}}(t)_{i,:}^T\boldsymbol{\mathcal{Y}}(t)_{i,:}(\boldsymbol{\mathcal{Y}}(t)^T\mathbf{D}_2(t)\boldsymbol{\mathcal{Y}}(t) - \boldsymbol{\mathcal{H}}(t)^T\boldsymbol{\mathcal{H}}(t))) \\
&+ \frac{1-\lambda_{SM}}{D_{2,ii}(t)} \\
&= \lambda_{SM}(1-\beta)(\boldsymbol{\mathcal{Y}}(t)_{i,:}\boldsymbol{\mathcal{Y}}(t)^T\mathbf{D}_2(t)\boldsymbol{\mathcal{Y}}(t)\boldsymbol{\mathcal{Y}}(t)_{i,:}^T - \boldsymbol{\mathcal{Y}}(t)_{i,:}\boldsymbol{\mathcal{H}}(t)^T\boldsymbol{\mathcal{H}}(t))\boldsymbol{\mathcal{Y}}(t)_{i,:}^T) \\
&+ \frac{1-\lambda_{SM}}{D_{2,ii}(t)} \\
&= \lambda_{SM}(1-\beta)(\|\mathbf{M}_{Yi,:}\|^2_{\mathbf{D}_2(t)} - \|\mathbf{W}_{YHi,:}\|_2^2) + \frac{1-\lambda_{SM}}{D_{2,ii}(t)}. \qquad \text{(A.13)}
\end{aligned}$$

A.3 Hyperparameters Used in the Numerical Experiments

In this section, we elaborate on the hyperparameter selections and training details for the numerical experiments presented in 4.8.

A.3.1 Synthetically Correlated Source Separation with Nonnegative Anti-sparse Sources

For the experiment with nonnegative antisparse WSM network in Section 4.8.2, we utilized the following hyperparameter selections:

- $\mathbf{D}_1 = \mathbf{I}$, $\mathbf{D}_2 = \mathbf{I}$, $\mathbf{W}_{HX} = \mathbf{I}, \mathbf{W}_{YH} = \mathbf{I}$, $\mathbf{M}_H = 2\mathbf{I}$, and $\mathbf{M}_Y = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

- $\beta = 0.5,\ \lambda_{SM} = 1 - 10^{-5}$, $\mu_{\mathbf{D}_1} = 1$, and $\mu_{\mathbf{D}_2} = 10^{-2}$.

- The term $1-\gamma^2$ is dynamically adjusted using the following equation: $1-\gamma^2 = max\{\nu/(1+\log(1+t)), 0.001\}$. Here t represents the data sample index, and $\nu = \begin{cases} 0.1 & \rho \leq 0.4, \\ 0.05 & \text{otherwise.} \end{cases}$.

- Learning rate for the neural dynamic iterations is determined using the following formula: $max\{0.75/(1+\tau \times 0.005), 0.05\}$. Here, τ represents the count of neural dynamic iteration.

- If the stopping condition is not satisfied, the maximum number of neural dynamic iterations is restricted to $\tau_{\max} = 500$.

- To ensure the stability of the learning process, we constrain the diagonal weights of $\mathbf{D}_1$ and $\mathbf{D}_2$ within a predetermined range, i.e., $0.2 \prec \text{diag}(\mathbf{D}_1) \prec 10^6$ and $0.2 \prec \text{diag}(\mathbf{D}_2) \prec 5$.

A.3.2 Synthetically Correlated Source Separation with Anti-sparse Sources

For the experiment with antisparse WSM network in Section 4.8.3, we utilized the following hyperparameter selections:

- $\mathbf{D}_1 = \mathbf{I}$, $\mathbf{D}_2 = \mathbf{I}$, $\mathbf{W}_{HX} = \mathbf{I}, \mathbf{W}_{YH} = \mathbf{I}$, $\mathbf{M}_H = 2\mathbf{I}$, and $\mathbf{M}_Y = \mathbf{I}$,

- $\beta = 0.5,\ \lambda_{SM} = 1 - 5 \times 10^{-5}$, $\mu_{\mathbf{D}_1} = 1.125$, and $\mu_{\mathbf{D}_2} = 0.2$,

- $1 - \gamma^2 = max\{\nu/(1 + \log(1 + t)), 0.001\}$. Here $\nu = \begin{cases} 0.6 & \rho \leq 0.4, \\ 0.25 & \text{otherwise.} \end{cases}$.

- Learning rate for the neural dynamic iterations: $max\{0.75/(1+\tau\times 0.005), 0.05\}$, and $\tau_{\max} = 750$

- Diagonal weight constraints: $0.2 \prec \text{diag}(\mathbf{D}_1) \prec 10^6$, $0.2 \prec \text{diag}(\mathbf{D}_2) \prec 5$.

A.3.3 Image Separation

For the image separation experiment in Section 4.8.4, we utilized the following hyperparameter selections:

- $\mathbf{D}_1 = \mathbf{I}$, $\mathbf{D}_2 = \mathbf{I}$, $\mathbf{W}_{HX} = \mathbf{I}, \mathbf{W}_{YH} = \mathbf{I}$, $\mathbf{M}_H = 2\mathbf{I}$, and $\mathbf{M}_Y = \mathbf{I}$.

- $\beta = 0.5,\ \lambda_{SM} = 1 - 10^{-5}$, $\mu_{\mathbf{D}_1} = 3.725$, and $\mu_{\mathbf{D}_2} = 1.125$.

- $1-\gamma^2 = max\{0.11/(1+\log(1+t)), 0.001\}$.
- Learning rate for the neural dynamic iterations: $max\{0.75/(1+\tau\times 0.005), 0.05\}$, and $\tau_{\max} = 500$.
- Diagonal weight constraints: $10^{-3} \prec \text{diag}(\mathbf{D}_1) \prec 10^6$, $10^{-3} \prec \text{diag}(\mathbf{D}_2) \prec 20$.

A.3.4 Sparse Source Separation

For the sparse source separation experiment in Section 4.8.5, we utilized the following hyperparameter selections:

- $\mathbf{D}_1 = 8\mathbf{I}$, $\mathbf{D}_2 = \mathbf{I}$, $\mathbf{M}_H = 0.02\mathbf{I}$, and $\mathbf{M}_Y = 0.02\mathbf{I}$.
- The $\mathbf{W}$ matrices are first initialized with i.i.d. standard normal random variables. Subsequently, we normalize the Euclidean norm of all rows to a value of 0.0033 through appropriate scaling.
- $\beta = 0.5$, $\lambda_{SM} = 1-10^{-5}$, $\mu_{\mathbf{D}_1} = 20$, and $\mu_{\mathbf{D}_2} = 10^{-2}$.
- $1-\gamma^2 = max\{0.25/(1+\log(1+t)), 0.001\}$.
- Learning rate for the neural dynamic iterations is constant and equal to 0.5, and $\tau_{\max} = 750$.
- Diagonal weight constraints: $10^{-6} \prec \text{diag}(\mathbf{D}_1) \prec 10^6$, $1 \prec \text{diag}(\mathbf{D}_2) \prec 1.001$.

A.3.5 Source Separation with Mixed Latent Attributes

For the *special polytope example* experiment in Section 4.8.7, we utilized the following hyperparameter selections:

- $\mathbf{D}_1 = 4\mathbf{I}$, and $\mathbf{D}_2 = \mathbf{I}$, $\mathbf{M}_H = 0.02\mathbf{I}$, and $\mathbf{M}_Y = 0.02\mathbf{I}$.
- The $\mathbf{W}$ matrices are first initialized with i.i.d. standard normal random variables. Subsequently, we normalize the Euclidean norm of all rows to a value of 0.0033 through appropriate scaling.

- $\beta = 0.5,\ \lambda_{SM} = 1 - 10^{-4}$, $\mu_{\mathbf{D}_1} = 5.725$, and $\mu_{\mathbf{D}_2} = 10^{-2}$.
- $1 - \gamma^2 = max\{0.25/(1 + \log(1 + t)), 0.001\}$.
- Learning rate for the neural dynamic iterations is constant and equal to 0.5, and $\tau_{\max} = 750$.
- Diagoanl weight constraints: $10^{-6} \prec \text{diag}(\mathbf{D}_1) \prec 10^6$, $1 \prec \text{diag}(\mathbf{D}_2) \prec 1.001$.

For the *Mixed anti-sparse and nonnegative anti-sparse sources* experiment in Section 4.8.7, we utilized the following hyperparameter selections:

- $\mathbf{D}_1 = \mathbf{I}$, $\mathbf{D}_2 = \mathbf{I}$, $\mathbf{W}_{HX} = \mathbf{I}, \mathbf{W}_{YH} = \mathbf{I}$, $\mathbf{M}_H = 2\mathbf{I}$, and $\mathbf{M}_Y = \mathbf{I}$.
- $\beta = 0.5,\ \lambda_{SM} = 1 - 5 \times 10^{-5}$, $\mu_{\mathbf{D}_1} = 1.125$, and $\mu_{\mathbf{D}_2} = 0.1$.
- $1 - \gamma^2 = max\{0.4/(1 + \log(1 + t)), 0.001\}$.
- Learning rate for the neural dynamic iterations: $max\{0.75/(1+\tau\times 0.005), 0.05\}$, and $\tau_{\max} = 750$.
- Diagonal weight constraints: $0.2 \prec \text{diag}(\mathbf{D}_1) \prec 10^6$, $0.5 \prec \text{diag}(\mathbf{D}_2) \prec 5$.

For the *Mixed sparse and nonnegative anti-sparse sources* experiment in Section 4.8.7, we utilized the following hyperparameter selections:

- $\mathbf{D}_1 = 8\mathbf{I}$, $\mathbf{D}_2 = \mathbf{I}$, $\mathbf{M}_H = 0.02\mathbf{I}$, and $\mathbf{M}_Y = 0.02\mathbf{I}$.
- The $\mathbf{W}$ matrices are first initialized with i.i.d. standard normal random variables. Subsequently, we normalize the Euclidean norm of all rows to a value of 0.0033 through appropriate scaling.
- $\beta = 0.5,\ \lambda_{SM} = 1 - 10^{-4}$, $\mu_{\mathbf{D}_1} = 6$, and $\mu_{\mathbf{D}_2} = 0.1$.
- $1 - \gamma^2 = max\{0.25/(1 + \log(1 + t)), 0.001\}$.

- Learning rate for the neural dynamic iterations: $max\{0.5/(1+\tau\times 0.005), 0.01\}$, and $\tau_{\max} = 750$.

- Diagonal weight constraints: $10^{-6} \prec \text{diag}(\mathbf{D}_1) \prec 10^6$, $1 \prec \text{diag}(\mathbf{D}_2) \prec 5$.

A.3.6 Digital Communication Example: 4-PAM Modulation Scheme

For the digital communication experiment in Section 4.8.8, we utilized the following hyperparameter selections:

- $\mathbf{D}_1 = 0.5\mathbf{I}$, $\mathbf{D}_2 = 0.5\mathbf{I}$, $\mathbf{M}_H = 2\mathbf{I}$, and $\mathbf{M}_Y = \mathbf{I}$.

- The $\mathbf{W}$ matrices are first initialized with i.i.d. standard normal random variables. Subsequently, we normalize the Euclidean norm of all rows to a value of 0.005 through appropriate scaling.

- $\beta = 0.5$, and $\lambda_{SM} = 1 - 5 \times 10^{-3}$, $\mu_{\mathbf{D}_1} = 0.01$, and $\mu_{\mathbf{D}_2} = 0.01$.

- $1 - \gamma^2 = max\{0.3/(1 + \log(1 + t)), 0.05\}$.

- Learning rate for the neural dynamic iterations: $max\{0.5/(1+\tau\times 0.005), 0.01\}$, and $\tau_{\max} = 750$.

- Diagonal weight constraints: $0.2 \prec \text{diag}(\mathbf{D}_1) \prec 10^6$, $0.2 \prec \text{diag}(\mathbf{D}_2) \prec 25$.

A.4 Ablation Study on Hyperparameter Selection for Nonnegative Sparse Sources

The Det-Max WSM framework proposed in this study involves several hyperparameters that need to be carefully selected. Finding optimal values for these hyperparameters often requires trial and error as well as sensitivity analysis. To determine the optimal values, it is useful to conduct ablation studies, similar to grid search, to analyze the effects of different hyperparameter selections.

In this section, we present ablation studies focusing on the effects of the selection of hyperparameters such as λ_{SM}, $\mathbf{D}_1$, $\mu_{\mathbf{D}_1}$, and γ. We specifically chose to highlight

the impact of λ_{SM} since we observed that it is one of the most sensitive parameters. Even though the other parameters may have a relatively smaller effect on the final results compared to λ_{SM}, it is important to consider the cumulative impact of all the hyperparameter choices, as they can collectively influence the overall performance of the Det-Max WSM framework.

For the ablation studies, we choose to consider nonnegative sparse source separation setup, i.e., $\mathcal{P} = \mathcal{B}_{\infty,+}$. We generate $n = 5$ dimensional source vectors uniformly from $\mathcal{B}\infty, +$. The mixing matrix is a 10×5 matrix with i.i.d. standard normal entries. These mixtures are used to train the nonnegative sparse-WSM Det-Max network, as depicted in Figure 4.4. In the ablation studies, we specifically focus on the effect of hyperparameter selection for $1-\lambda_{\text{SM}}$, initial $\mathbf{D}1$, $\mu_{\mathbf{D}_1}$, and initial $1-\gamma^2$. For each of these hyperparameters, we consider the following choices:

- $1 - \lambda_{\text{SM}} \in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$,
- $\mathbf{D}_1 \in \{4\mathbf{I}, 8\mathbf{I}, 12\mathbf{I}, 16\mathbf{I}\}$,
- $\mu_{\mathbf{D}_1} \in \{5, 10, 15, 20\}$,
- initial $1 - \gamma^2 \in \{0.15, 0.20, 0.25, 0.30\}$

While experimenting with one hyperparameter, we keep the rest of the hyperparameters fixed as follows:

- $\mathbf{D}_1 = 4\mathbf{I}$, and $\mathbf{D}_2 = \mathbf{I}$.
- $\mu_{\mathbf{D}_1} = 15$, and $\mu_{\mathbf{D}_2} = 0.01$.
- $\beta = 0.5$, and $\lambda_{SM} = 1 - 10^{-4}$.
- $1-\gamma^2$ is dynamically adjusted using $1-\gamma^2 = max\{0.25/(1+\log(1+t)), 10^{-3}\}$, where t is the data sample index.
- $\mathbf{M}_H = 0.02\mathbf{I}, \mathbf{M}_Y = 0.02\mathbf{I}$.

- **W** matrices are initially initialized with i.i.d. standard normal random variables. After initialization, we perform a row-wise normalization such that the Euclidean norm of each row is scaled to 0.0033

- The learning rate for the neural dynamic iterations is adjusted using the formula $max\{0.5/(1+\tau \times 0.005), 0.2\}$, where τ represents the neural dynamic iteration count.

- If the stopping condition is not satisfied, the maximum number of neural dynamic iterations is restricted to $\tau_{\max} = 750$.

- To ensure the stability of the learning process, we constrain the diagonal weights of $\mathbf{D}_1$ and $\mathbf{D}_2$ in specific ranges, i.e., $10^{-6} \prec \text{diag}(\mathbf{D}_1) \prec 10^6$ and $1 \prec \text{diag}(\mathbf{D}_2) \prec 1.001$.

Figure A.1a demonstrates the sensitivity of the SINR performance of the WSM Det-Max network to the hyperparameter $1 - \lambda_{\text{SM}}$. The plot shows that different choices of $1 - \lambda_{\text{SM}}$ significantly impact the final SINR behavior of the approach. Based on the results, we suggest that selecting $\lambda_{\text{SM}} = 1 - 10^{-4}$ is near-optimal for nonnegative sparse source separation with the WSM Det-Max network.

In Figure A.1b, we analyze the effect of the initial $\mathbf{D}_1$ gain on the final SINR. The plot shows that the WSM Det-Max network for nonnegative sparse sources maintains relatively consistent performance across different initial gain parameters. However, initializing $\mathbf{D}_1$ with $\mathbf{D}_1 = 4\mathbf{I}$ leads to the best performance with significantly lower variance compared to other initialization choices.

Figure A.1c visualizes the effect of the learning rate choice for $\mathbf{D}_1$. The plot indicates that $\mu_{\mathbf{D}_1}$ has a less pronounced effect on the SINR performance compared to other considered hyperparameters. However, $\mu_{\mathbf{D}_1} = 15$ achieves the best average result with a lower variance.

Lastly, we consider the initial value of $1 - \gamma^2$, which is dynamically adjusted using the function $max\{\nu/(1+\log(1+t)), 10^{-3}\}$, where t is the data sample index and ν is the initial value. Figure A.1d illustrates the effect of the initial value ν

on the SINR performance. The plot shows that an improved result is obtained for $\nu = 0.25$.

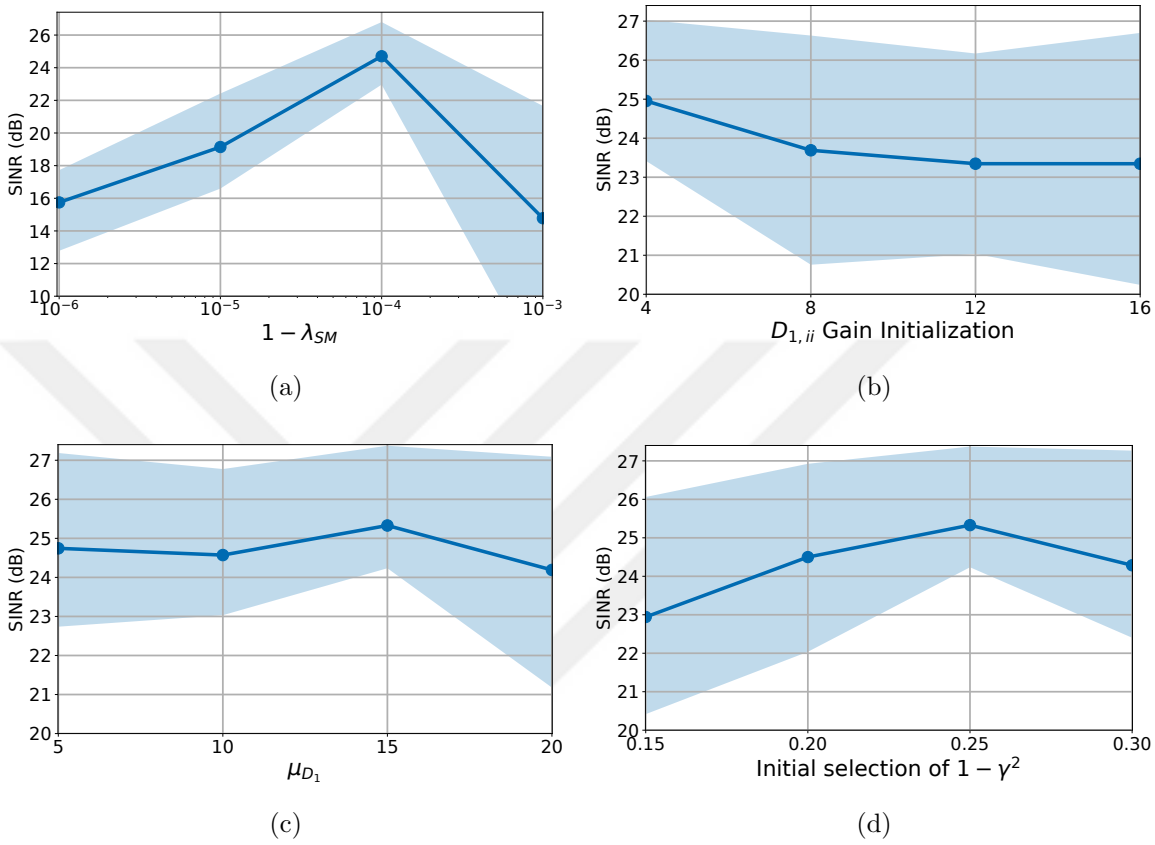


Figure A.1: SINR performances of WSM Det-Max networks for different hyper-parameter selections (averaged over 50 realizations, mean solid lines with 25/75-percentile envelopes): (a) averaged SINR performance with respect to $1 - \lambda_{\text{SM}}$, (b) averaged SINR performance with respect to initial $\mathbf{D}_1$, (c) averaged SINR performance with respect to $\mu_{\mathbf{D}_1}$, (d) averaged SINR performance with respect to initial $1 - \gamma^2$.

A.5 Discussion on the Complexity of the Proposed Approach

In this section, we examine the computational complexity of our proposed WSM Det-Max neural network implementations. To maintain simplicity, we focus on the

antisparse source separation scenarios discussed in Section 4.5.2. Notably, the overall complexity primarily arises from the computations involved in determining the network's outputs, which are governed by (4.9)-(4.10) and (4.11)-(4.12). It should be noted that these differential equations can be naturally solved in neuromorphic hardware implementations. However, in digital computer simulations, we need to utilize iterative solutions through the implementation of loops, as outlined in Algorithm 2. As explained in Section 2.1, let us assume that there are n sources and m mixtures, denoted as $\mathbf{x}(t) \in \mathbb{R}^m$, and $\mathbf{h}(t), \mathbf{y}(t) \in \mathbb{R}^n$ for all t. If we consider that the factors $(1-\beta)\bar{\mathbf{M}}_H(t) + \beta\mathbf{D}_1(t)\bar{\mathbf{M}}_H(t)\mathbf{D}_1(t)$, $\beta\mathbf{D}_1(t)\mathbf{W}_{HX}(t)$, $\mathbf{W}_{YH}(t)^T\mathbf{D}_2(t)$, and $\bar{\mathbf{M}}_Y(t)\mathbf{D}_2(t)$ are pre-computed outside the iterative loop of Algorithm 2, the expressions in (4.9) and (4.11) require $2n^2+mn$ and $2n^2$ multiplications, respectively. Assuming that the neural dynamic loop reaches the predetermined maximum number of iterations τmax, meaning that the convergence check based on the numerical relative error is not satisfied, the total number of multiplications is dominated by the term $\tau_{\max}(4n^2+mn)$. Upon analyzing the computational requirements of the factors $(1-\beta)\bar{\mathbf{M}}_H(t) + \beta\mathbf{D}_1(t)\bar{\mathbf{M}}_H(t)\mathbf{D}_1(t)$, $\beta\mathbf{D}_1(t)\mathbf{W}_{HX}(t)$, $\mathbf{W}_{YH}(t)^T\mathbf{D}_2(t)$, and $\bar{\mathbf{M}}_Y(t)\mathbf{D}_2(t)$, we find that these calculations involve the following number of multiplications: $(n^2-n)/2+n^2+3n$, $mn+n$, n^2, and n^2, respectively. This is due to the diagonal nature of $\mathbf{D}_1(t)$ and $\mathbf{D}_2(t)$, as well as the symmetry of $\bar{\mathbf{M}}_H(t)$ and $\bar{\mathbf{M}}Y(t)$. Consequently, the computational complexity of the neural dynamics in our proposed approach is primarily dominated by the term $\tau_{\max}(4n^2+mn)$. Regarding the update rules of the gain variables given in equations (4.13) and (4.14), their complexity is dominated by a multiplication factor of $3n$ for each of the $2n$ variables, resulting in a total multiplication factor of $6n^2$. Furthermore, the update rules of the synaptic weight updates expressed in equation (4.16) are dominated by a multiplication factor of n^2 or mn, contributing to a total of $4(n^2+mn)$ multiplications. Considering the worst-case scenario, the overall complexity of our proposed method per sample, in terms of big-O notation, is $\mathcal{O}(\tau_{\max}mn)$.

Let us compare the complexity of the WSM algorithm with the NSM and BSM algorithms. First, we focus on the complexity of the prewhitening layer, which is required by both the NSM and BSM algorithms. Referring to equations (28),

(29), and (30) in the paper [Pehlevan et al., 2017b] for output computation and synaptic weight updates of the prewhitening layer, the computational complexity can be found as $\mathcal{O}(\tau_{\max}^{(\mathrm{NSM})}(m+k)n)$ in terms of big-O notation. Here, $k \geq n$ is an integer introduced as a result of the Lagrangian multiplier in equation (12) of the same reference [Pehlevan et al., 2017b], and $\tau_{\max}^{(\mathrm{NSM})}$ is the maximum predetermined number of iterations for the neural dynamic loop of NSM (see equation (28) and (33) in the reference). The output dynamics and synaptic weight updates of the second layer in the online NSM network are described by equations (33), (34), and (35) in [Pehlevan et al., 2017b], leading to a complexity of $\mathcal{O}(n^2)$ in terms of big-O notation. Consequently, the overall complexity of the NSM algorithm per sample can be expressed as $\mathcal{O}(\tau_{\max}^{\mathrm{NSM}}(m+k)n)$. Similarly, the neural dynamic loop of the BSM algorithm exhibits a complexity of $\mathcal{O}(\tau_{\max}^{(\mathrm{BSM})}mn)$ due to the recursion defined in Equation (17) of [Erdogan and Pehlevan, 2020], where $\tau_{\max}^{(\mathrm{BSM})}$ represents the predetermined maximum number of iterations for the neural dynamic loop in BSM. Additionally, the synaptic weight and gain updates introduce a complexity of $\mathcal{O}(n^2)$, similar to the WSM algorithm. Hence, when combined with the prewhitening step, the overall complexity of the BSM algorithm per sample becomes $\mathcal{O}(\tau_{\max}^{(\mathrm{BSM})}(m+k)n)$.

In conclusion, when considering biologically plausible neural network solutions for the blind source separation problem, the overall complexity is primarily determined by the recursive neural dynamic loops, which arise due to the implicit definition of the network output. While this characteristic may pose challenges for implementation on digital hardware, it presents an advantage for low-power implementations in future analog neuromorphic systems with local learning constraints.

Appendix B

SUPPLEMENTARY ON CORINFOMAX-BASED BSS NETWORKS

B.1 Gradient Derivations for Online Optimization Objective

Given that the mapping $\mathbf{W}(k)$ changes gradually over time, the current output $\mathbf{y}(k)$ can be implicitly determined using the projected gradient ascent technique within neural dynamics. To obtain the corresponding neural dynamics for $\mathbf{y}(k)$, it is necessary to compute the gradient of the objective function 5.3a with respect to $\mathbf{y}(k)$. Initially, we focus on calculating the derivatives of both $\log\det(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})$ and $\log\det(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})$ with respect to $\mathbf{y}(k)$.

$$\frac{\partial \log\det(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})}{\partial y_i(k)} = Tr\left(\nabla_{(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})}\log\det(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})\frac{\partial\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)}{\partial y_i(k)}\right),$$

where $y_i(k)$ is the i-th element of the vector $\mathbf{y}(k)$, and

$$\nabla_{(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})}\log\det(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I}) = (\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})^{-1}, \tag{B.1}$$

$$\frac{\partial\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)}{\partial y_i(k)} = \frac{1-\zeta_{\mathbf{y}}}{1-\zeta_{\mathbf{y}}^k}(\mathbf{y}(k)\mathbf{e}^{(i)^T}+\mathbf{e}^{(i)}\mathbf{y}(k)^T). \tag{B.2}$$

In equation (B.2), the notation $\mathbf{e}^{(i)}$ represents the standard basis vector with a value of 1 at position i. It should not be mistaken for the error vector $\mathbf{e}(k)$. By combining equations (B.1) and (B.2), we can derive the following outcome:

$$\frac{\partial \log\det(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})}{\partial y_i(k)} = 2\frac{1-\zeta_{\mathbf{y}}}{1-\zeta_{\mathbf{y}}^k}\mathbf{e}^{(i)^T}(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k),$$

which leads to

$$\nabla_{\mathbf{y}(k)}\log\det(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I}) = 2\frac{1-\zeta_{\mathbf{y}}}{1-\zeta_{\mathbf{y}}^k}(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k). \tag{B.3}$$

If we follow a similar procedure to calculate the gradient of $\log\det(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})$ with respect to $\mathbf{e}(k)$, we arrive at the following expression:

$$\nabla_{\mathbf{e}(k)}\log\det(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})=2\frac{1-\zeta_{\mathbf{e}}}{1-\zeta_{\mathbf{e}}^{k}}(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})^{-1}\mathbf{e}(k).$$

By employing the composition rule, we can derive the gradient of $\log\det(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})$ with respect to $\mathbf{y}(k)$ in the following manner:

$$\begin{aligned}\nabla_{\mathbf{y}(k)}\log\det(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})&=\underbrace{\frac{\partial\mathbf{e}(k)}{\partial\mathbf{y}(k)}}_{\boldsymbol{I}_n}\nabla_{\mathbf{e}(k)}\log\det(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})\\&=2\frac{1-\zeta_{\mathbf{e}}}{1-\zeta_{\mathbf{e}}^{k}}(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})^{-1}\mathbf{e}(k).\end{aligned}\tag{B.4}$$

By consolidating the outcomes presented in (B.3) and (B.4), we can determine the derivative of the objective function $\mathcal{J}(\mathbf{y}(k))$ with respect to $\mathbf{y}(k)$ as:

$$\begin{aligned}\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k))&=\frac{1}{2}\nabla_{\mathbf{y}(k)}\log\det(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})-\frac{1}{2}\nabla_{\mathbf{y}(k)}\log\det(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})\\&=\frac{1-\zeta_{\mathbf{y}}}{1-\zeta_{\mathbf{y}}^{k}}(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)-\frac{1-\zeta_{\mathbf{e}}}{1-\zeta_{\mathbf{e}}^{k}}(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})^{-1}\mathbf{e}(k)\end{aligned}\tag{B.5}$$

To simplify equation (B.5) further, we introduce recursive definitions for $\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)^{-1}$ and $\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)^{-1}$ using the recursive definitions of the respective correlation matrices. Referring to the definition provided in equation (5.2), we can express the equation as follows:

$$\begin{aligned}\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I}&=\frac{1-\zeta_{\mathbf{y}}^{k-1}}{1-\zeta_{\mathbf{y}}^{k}}\zeta_{\mathbf{y}}(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)+\epsilon\boldsymbol{I})+\frac{1-\zeta_{\mathbf{y}}}{1-\zeta_{\mathbf{y}}^{k}}\mathbf{y}(k)\mathbf{y}(k)^{T}+\frac{1-\zeta_{\mathbf{y}}}{1-\zeta_{\mathbf{y}}^{k}}\epsilon\boldsymbol{I}\\&\approx\frac{1-\zeta_{\mathbf{y}}^{k-1}}{1-\zeta_{\mathbf{y}}^{k}}\zeta_{\mathbf{y}}(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)+\epsilon\boldsymbol{I})+\frac{1-\zeta_{\mathbf{y}}}{1-\zeta_{\mathbf{y}}^{k}}\mathbf{y}(k)\mathbf{y}(k)^{T}\end{aligned}\tag{B.6}$$

By utilizing the assumption stated in (B.6), we invert both sides of the equation and employ the matrix inversion lemma (similar to its application in the derivation of the RLS algorithm [Kailath et al., 2000]). This leads us to the following expression:

$$\begin{aligned}(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})^{-1}=\frac{1-\zeta_{\mathbf{y}}^{k}}{\zeta_{\mathbf{y}}-\zeta_{\mathbf{y}}^{k}}\Big((&\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)+\epsilon\boldsymbol{I})^{-1}\\&-\gamma_{\mathbf{y}}(k)(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)\mathbf{y}(k)^{T}(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)+\epsilon\boldsymbol{I})^{-1}\Big),\end{aligned}\tag{B.7}$$

where

$$\gamma_{\mathbf{y}}(k)=\left(\frac{\zeta_{\mathbf{y}}-\zeta_{\mathbf{y}}^{k}}{1-\zeta_{\mathbf{y}}}+\mathbf{y}(k)^{T}(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)\right)^{-1}. \tag{B.8}$$

We follow a similar procedure to compute the inverse of $(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})^{-1}$:

$$\begin{aligned}(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})^{-1}=\frac{1-\zeta_{\mathbf{e}}^{k}}{\zeta_{\mathbf{e}}-\zeta_{\mathbf{e}}^{k}}\Big((&\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)+\epsilon\boldsymbol{I})^{-1}\\&-\gamma_{\mathbf{e}}(k)(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{e}(k)\mathbf{e}(k)^{T}(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)+\epsilon\boldsymbol{I})^{-1}\Big),\end{aligned} \tag{B.9}$$

where

$$\gamma_{\mathbf{e}}(k)=\left(\frac{\zeta_{\mathbf{e}}-\zeta_{\mathbf{e}}^{k}}{1-\zeta_{\mathbf{e}}}+\mathbf{e}(k)^{T}(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{e}(k)\right)^{-1}. \tag{B.10}$$

Note that substituting (B.7) into the first part of (B.5) results in the following simplification:

$$\begin{aligned}&\frac{1-\zeta_{\mathbf{y}}}{1-\zeta_{\mathbf{y}}^{k}}(\hat{\mathbf{R}}(k)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)=\\&\frac{1-\zeta_{\mathbf{y}}}{1-\zeta_{\mathbf{y}}^{k}}\frac{1-\zeta_{\mathbf{y}}^{k}}{\zeta_{\mathbf{y}}-\zeta_{\mathbf{y}}^{k}}\Big((\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\\&-\gamma_{\mathbf{y}}(k)(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)\mathbf{y}(k)^{T}(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\Big)\mathbf{y}(k)\\&=\frac{1-\zeta_{\mathbf{y}}}{\zeta_{\mathbf{y}}-\zeta_{\mathbf{y}}^{k}}\Bigg((\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\\&-\frac{(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)\mathbf{y}(k)^{T}(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}}{\frac{\zeta_{\mathbf{y}}-\zeta_{\mathbf{y}}^{k}}{1-\zeta_{\mathbf{y}}}+\mathbf{y}(k)^{T}(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)}\Bigg)\mathbf{y}(k)\\&=\frac{1-\zeta_{\mathbf{y}}}{\zeta_{\mathbf{y}}-\zeta_{\mathbf{y}}^{k}}\Big(\frac{\frac{\zeta_{\mathbf{y}}-\zeta_{\mathbf{y}}^{k}}{1-\zeta_{\mathbf{y}}}(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)}{\frac{\zeta_{\mathbf{y}}-\zeta_{\mathbf{y}}^{k}}{1-\zeta_{\mathbf{y}}}+\mathbf{y}(k)^{T}(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)}\Big)\\&+\frac{1-\zeta_{\mathbf{y}}}{\zeta_{\mathbf{y}}-\zeta_{\mathbf{y}}^{k}}\Big(\frac{(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)\mathbf{y}(k)^{T}(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)}{\frac{\zeta_{\mathbf{y}}-\zeta_{\mathbf{y}}^{k}}{1-\zeta_{\mathbf{y}}}+\mathbf{y}(k)^{T}(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)}\Big)\\&-\frac{1-\zeta_{\mathbf{y}}}{\zeta_{\mathbf{y}}-\zeta_{\mathbf{y}}^{k}}\Big(\frac{(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)\mathbf{y}(k)^{T}(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)}{\frac{\zeta_{\mathbf{y}}-\zeta_{\mathbf{y}}^{k}}{1-\zeta_{\mathbf{y}}}+\mathbf{y}(k)^{T}(\hat{\mathbf{R}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k)}\Big)\\&=\gamma_{\mathbf{y}}(k)(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)+\epsilon\boldsymbol{I})^{-1}\mathbf{y}(k).\end{aligned}$$

We can also simplify equation (B.9) in a similar manner. By incorporating these simplifications into equation (B.5), we obtain the following expression:

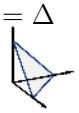
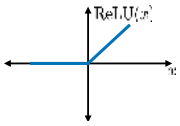
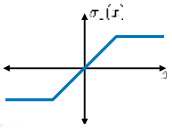
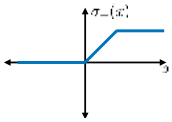
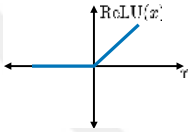
$$\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k)) = \gamma_{\mathbf{y}}(k)\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k) - \gamma_{\mathbf{e}}(k)\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)\mathbf{e}(k), \tag{B.11}$$

where we denote $(\hat{\mathbf{R}}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)+\epsilon\boldsymbol{I})^{-1}$ and $(\hat{\mathbf{R}}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)+\epsilon\boldsymbol{I})^{-1}$ by $\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k+1)$ and $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k+1)$ for simplicity, respectively.

B.2 Online CorInfoMax Algorithm Implementations for Special Source Domains

In this section, we provide a detailed algorithmic description of the proposed approach. Algorithm 6 outlines the dynamics of the CorInfoMax network and learning rules, while Table B.1 summarizes the output dynamics for specific source domains discussed in Section 5.3. In the algorithm, the recurrent and feedforward weight updates follow equations (5.12) and (5.13), respectively, and the learning steps are indicated in the 4th and 5th lines of the pseudo-code. The 3rd line expresses the recursive neural dynamics for obtaining the network's output, and its implementation varies depending on the chosen domain. Based on the derivations in Section 5.4, Algorithm 7, 8, 9, and 10 summarize the neural dynamic iterations for different example domains. As an example, Algorithm 9 outlines the procedure to obtain the output $\mathbf{y}(k)$ of the antisparse CorInfoMax network at the time step k corresponding to the mixture vector $\mathbf{x}(k)$. To facilitate implementation in digital hardware, two variables are introduced: 1) a numerical convergence tolerance ϵ_t, and 2) a maximum number of neural dynamic iterations $\nu_{\max}$. The proposed neural dynamic iterations are executed until either convergence is achieved, indicated by $||\mathbf{y}(k;\nu) - \mathbf{y}(k;\nu-1)||/||\mathbf{y}(k;\nu)|| > \epsilon_t$, or the loop counter reaches the predetermined maximum number of iterations, i.e., $\nu = \nu_{\max}$. In contrast to the antisparse and nonnegative antisparse networks, the other CorInfoMax neural networks incorporate additional inhibitory neurons due to the Lagrangian Min-Max settings. The activation of these inhibitory neurons are linked to the network's output. Consequently, the update of inhibitory neurons in the neural dynamics is determined by the gradient of the Lagrangian objective.

Table B.1: Example source domains and the corresponding CorInfoMax network dynamics.

Source Domain	**Output Dynamics**	**Output Activation**
$\mathcal{P}=\Delta$ 	$\nabla_{\mathbf{y}(k)}J(\mathbf{y}(k;\nu))=\gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)\mathbf{y}(k;\nu)-\gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)\mathbf{e}(k;\nu)$, $\mathbf{y}(k;\nu+1)=\text{ReLU}\left(\mathbf{y}(k;\nu)+\eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}J(\mathbf{y}(k;\nu))-\lambda(\nu)\right)$, $\lambda(k;\nu+1)=\lambda(k;\nu)-\eta_{\lambda}(\nu)\left(1-\left(\sum_{i=1}^{n}y_i(k;\nu+1)\right)\right)$.	
$\mathcal{P}=\mathcal{B}_{\ell_\infty}$	$\nabla_{\mathbf{y}(k)}J(\mathbf{y}(k;\nu))=\gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)\mathbf{y}(k;\nu)-\gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)\mathbf{e}(k;\nu)$, $\mathbf{y}(k;\nu+1)=\sigma_1\left(\mathbf{y}(k;\nu)+\eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}J(\mathbf{y}(k;\nu))\right)$,	
$\mathcal{P}=\mathcal{B}_{\ell_\infty,+}$	$\nabla_{\mathbf{y}(k)}J(\mathbf{y}(k;\nu))=\gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k;\nu)-\gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)\mathbf{e}(k;\nu)$, $\mathbf{y}(k;\nu+1)=\sigma_+\left(\mathbf{y}(k;\nu)+\eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}J(\mathbf{y}(k;\nu))\right)$,	
$\mathcal{P}=\mathcal{B}_{\ell_1,+}$	$\nabla_{\mathbf{y}(k)}J(\mathbf{y}(k;\nu))=\gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k-1)\mathbf{y}(k;\nu)-\gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k-1)\mathbf{e}(k;\nu)$, $\mathbf{y}(k;\nu+1)=\text{ReLU}\left(\mathbf{y}(k;\nu)+\eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}J(\mathbf{y}(k;\nu))\right)$, $\lambda(k;\nu+1)=\text{ReLU}\left(\lambda(k;\nu)-\eta_{\lambda}(\nu)\left(1-\left(\sum_{i=1}^{n}y_i(k;\nu+1)\right)\right)\right)$.	

Algorithm 6 Online CorInfoMax pseudo-code

Input: A streaming data of $\{\mathbf{x}(k)\in\mathbb{R}^m\}_{k=1}^N$.

Output: $\{\mathbf{y}(k)\in\mathbb{R}^n\}_{k=1}^N$.

1: Initialize $\zeta_{\mathbf{y}},\zeta_{\mathbf{e}},\mu(1)_{\mathbf{W}},\mathbf{W}(1),\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)$, $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)$, and select $\mathcal{P}$.

2: **for** k = 1, 2, . . . , N **do**

3: **run neural dynamics (in Algorithms 7 to 9 below according to $\mathcal{P}$) until convergence** .

4: $\mathbf{W}(k+1)=\mathbf{W}(k)+\mu_{\mathbf{W}}(k)\mathbf{e}(k)\mathbf{x}(k)^T$

5: $\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k+1)=\frac{1}{\zeta_{\mathbf{y}}}(\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)-\frac{1-\zeta_{\mathbf{y}}}{\zeta_{\mathbf{y}}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k)\mathbf{y}(k)^T\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k))$

6: Adjust $\mu_{\mathbf{W}}(k+1)$ if necessary.

7: **end for**

Algorithm 7 Online CorInfoMax neural dynamic iterations: sources in unit simplex

Initialize $\nu_{\max}$, ϵ_t, $\eta_{\mathbf{y}}(1)$, $\eta_{\lambda}(1)$, $\lambda(1)(= 0$ in general) and $\nu = 1$

while $(||\mathbf{y}(k;\nu) - \mathbf{y}(k;\nu-1)||/||\mathbf{y}(k;\nu)|| > \epsilon_t)$ and $\nu < \nu_{\max}$ **do**

$\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) = \gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k;\nu) - \gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)\mathbf{e}(k;\nu)$

$\mathbf{y}(k;\nu+1) = \text{ReLU}\left(\mathbf{y}(k;\nu) + \eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) - \lambda(\nu)\right)$

$\nabla_{\lambda(k)}\mathcal{L}(\mathbf{y}(k;\nu)) = 1 - \|\mathbf{y}(k;\nu+1)\|_1$

$\lambda(k;\nu+1) = \lambda(k;\nu) - \eta_{\lambda}(\nu)\nabla_{\lambda(k)}\mathcal{L}(\mathbf{y}(k;\nu))$

$\nu = \nu + 1$, and adjust $\eta_{\mathbf{y}}(\nu), \eta_{\lambda}(\nu)$ if necessary.

end while

Algorithm 8 Online CorInfoMax neural dynamic iterations: sparse sources

Initialize $\nu_{\max}$, ϵ_t, $\eta_{\mathbf{y}}(1)$, $\eta_{\lambda}(1)$, $\lambda(1)(= 0$ in general) and $\nu = 1$

while $(||\mathbf{y}(k;\nu) - \mathbf{y}(k;\nu-1)||/||\mathbf{y}(k;\nu)|| > \epsilon_t)$ and $\nu < \nu_{\max}$ **do**

$\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) = \gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k;\nu) - \gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)\mathbf{e}(k;\nu)$

$\mathbf{y}(k;\nu+1) = ST_{\lambda(\nu)}\left(\mathbf{y}(k;\nu) + \eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) - \lambda(\nu)\right)$

$\nabla_{\lambda(k)}\mathcal{L}(\mathbf{y}(k;\nu)) = 1 - \|\mathbf{y}(k;\nu+1)\|_1$

$\lambda(k;\nu+1) = \text{ReLU}\left(\lambda(k;\nu) - \eta_{\lambda}(\nu)\nabla_{\lambda(k)}\mathcal{L}(\mathbf{y}(k;\nu))\right)$

$\nu = \nu + 1$, and adjust $\eta_{\mathbf{y}}(\nu), \eta_{\lambda}(\nu)$ if necessary.

end while

Algorithm 9 Online CorInfoMax neural dynamic iterations: antisparse sources

Initialize $\nu_{\max}$, ϵ_t, $\eta_{\mathbf{y}}(1)$ and $\nu = 1$

while $(||\mathbf{y}(k;\nu) - \mathbf{y}(k;\nu-1)||/||\mathbf{y}(k;\nu)|| > \epsilon_t)$ and $\nu < \nu_{\max}$ **do**

$\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) = \gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k;\nu) - \gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)\mathbf{e}(k;\nu)$

$\mathbf{y}(k;\nu+1) = \sigma_1\left(\mathbf{y}(k;\nu) + \eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu))\right)$

$\nu = \nu + 1$, and adjust $\eta_{\mathbf{y}}(\nu)$ if necessary.

end while

Algorithm 10 Online CorInfoMax neural dynamic iterations for Canonical Form

Initialize $\nu_{\max}$, ϵ_t, $\eta_{\mathbf{y}}(1)$, $\eta_{\boldsymbol{\lambda}}(1)$, $\boldsymbol{\lambda}(1)(=\mathbf{0}$ in general) and $\nu = 1$

while $(||\mathbf{y}(k;\nu) - \mathbf{y}(k;\nu-1)||/||\mathbf{y}(k;\nu)|| > \epsilon_t)$ and $\nu < \nu_{\max}$ **do**

$\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu)) = \gamma_{\mathbf{y}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k;\nu) - \gamma_{\mathbf{e}}\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)\mathbf{e}(k;\nu) - \mathbf{A}_{\mathcal{P}}^{T}\boldsymbol{\lambda}(\nu)$

$\mathbf{y}(k;\nu+1) = \mathbf{y}(k;\nu) + \eta_{\mathbf{y}}(\nu)\nabla_{\mathbf{y}(k)}\mathcal{J}(\mathbf{y}(k;\nu))$

$\nabla_{\boldsymbol{\lambda}(k)}\mathcal{L}(\mathbf{y}(k;\nu)) = -\mathbf{A}_{\mathcal{P}}\mathbf{y}(k;\nu) + \mathbf{b}_{\mathcal{P}}$

$\boldsymbol{\lambda}(k;\nu+1) = \text{ReLU}\left(\boldsymbol{\lambda}(k;\nu) - \eta_{\boldsymbol{\lambda}}(\nu)\nabla_{\boldsymbol{\lambda}(k)}\mathcal{L}(\mathbf{y}(k;\nu))\right)$

$\nu = \nu + 1$, and adjust $\eta_{\mathbf{y}}(\nu), \eta_{\boldsymbol{\lambda}}(\nu)$ if necessary.

end while

B.3 Hyperparameter Selections

The selection of hyperparameters plays a crucial role in determining the performance of the neural networks presented in this article. In this section, we present the list of hyperparameter selections used for the experiments conducted in the article. These parameters were chosen based on ablation studies and some trials, which are discussed in detail in Appendix B.4.

Table B.2 presents the selected hyperparameters for the special source domains discussed in Section 5.3. It is notable that the hyperparameter sets for different special source domains exhibit similarities. However, certain parameters show domain-specific variations, such as the initial learning rate for neural dynamics ($\eta_{\mathbf{y}}(1)$) and the learning rate for the Lagrangian variable ($\eta_{\lambda}(\nu)$).

Table B.3 presents the hyperparameter selections for the additional experiments, such as feature-based polytope and sparse dictionary learning.

B.4 Ablation Studies on Hyperparameter Selections

In this section, we present an ablation study focused on the selection of two hyperparameters for the nonnegative antisparse CorInfoMax network. While this study could be extended to include a grid search approach, for simplicity, we only consider two parameters: the learning rate of the feedforward weights, denoted as $\mu_{\mathbf{W}}$, and the initialization of the inverse error correlation matrix, denoted as $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}$. We explore

Table B.2: CorInfoMax network hyperparameter selections for special source domains in 5.3.

Source Domain	**Hyperparameters**
$\mathcal{P}=\mathcal{B}_{\ell_\infty}$	$\mathbf{W}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)=5\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)=5000\boldsymbol{I}$ $\zeta_{\mathbf{y}}=1-10^{-2}, \quad \zeta_{\mathbf{e}}=1-2\times 10^{-2}, \quad \mu_{\mathbf{W}}=3\times 10^{-2}$ $\nu_{\max}=500, \quad \eta_{\mathbf{y}}(\nu)=0.9/\nu, \quad \epsilon_t=10^{-6}$
$\mathcal{P}=\mathcal{B}_{\ell_\infty,+}$	$\mathbf{W}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)=5\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)=2000\boldsymbol{I}$ $\zeta_{\mathbf{y}}=1-10^{-2}, \quad \zeta_{\mathbf{e}}=1-10^{-1}/3, \quad \mu_{\mathbf{W}}=3\times 10^{-2}$ $\nu_{\max}=500, \quad \eta_{\mathbf{y}}(\nu)=\max\{0.9/\nu,10^{-3}\}, \quad \epsilon_t=10^{-6}$
$\mathcal{P}=\mathcal{B}_1$	$\mathbf{W}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)=1000\boldsymbol{I}$ $\zeta_{\mathbf{y}}=1-10^{-2}, \quad \zeta_{\mathbf{e}}=1-10^{-2}, \quad \mu_{\mathbf{W}}=3\times 10^{-2}$ $\nu_{\max}=500, \quad \eta_{\mathbf{y}}(\nu)=\max\{0.1/\nu,10^{-3}\},$ $\eta_{\lambda}(\nu)=1, \quad \epsilon_t=10^{-6}$
$\mathcal{P}=\mathcal{B}_{1,+}$	$\mathbf{W}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)=5\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)=1000\boldsymbol{I}$ $\zeta_{\mathbf{y}}=1-10^{-2}, \quad \zeta_{\mathbf{e}}=1-10^{-2}, \quad \mu_{\mathbf{W}}=3\times 10^{-2}$ $\nu_{\max}=500, \quad \eta_{\mathbf{y}}(\nu)=\max\{0.1/\nu,10^{-3}\},$ $\eta_{\lambda}(\nu)=1, \quad \epsilon_t=10^{-6}$
$\mathcal{P}=\Delta$	$\mathbf{W}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)=5\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)=1000\boldsymbol{I}$ $\zeta_{\mathbf{y}}=1-10^{-2}, \quad \zeta_{\mathbf{e}}=1-10^{-2}, \quad \mu_{\mathbf{W}}=3\times 10^{-2}$ $\nu_{\max}=500, \quad \eta_{\mathbf{y}}(\nu)=\max\{0.1/\nu,10^{-3}\},$ $\eta_{\lambda}(\nu)=0.05, \quad \epsilon_t=10^{-6}$

the following choices for these hyperparameters:

- $\mu_{\mathbf{W}} \in \{5\times 10^{-3}, 10^{-2}, 3\times 10^{-2}, 5\times 10^{-2}\}$,
- $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}} \in \{10^3, 2\times 10^3, 5\times 10^3, 10^4\}$.

In this ablation study, we conducted experiments using the experimental setup described in Section 5.5.1. We considered both uncorrelated sources and correlated sources with a correlation parameter of $\rho=0.6$. For the ablation study on $\mu_{\mathbf{W}}$, we

Table B.3: CorInfoMax network hyperparameter selections for the polytope with mixed latent attributes in (5.24) and application examples sparse dictionary learning, digital communication, video separation, and image separation.

Experiment	**Hyperparameters**
Polytope in (5.24) with Mixed Attributes (Canonical)	$\mathbf{W}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)=1000\boldsymbol{I}$ $\zeta_{\mathbf{y}}=1-10^{-2}, \quad \zeta_{\mathbf{e}}=1-10^{-2}, \quad \mu_{\mathbf{W}}=5\times10^{-2}$ $\nu_{\max}=500, \quad \eta_{\mathbf{y}}(\nu)=\max\{0.25/\nu, 10^{-4}\},$ $\eta_{\boldsymbol{\lambda}}(\nu)=0.1, \quad \epsilon_t=10^{-6}$
Polytope in (5.24) with Mixed Attributes (Feature-based)	$\mathbf{W}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)=5\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)=2500\boldsymbol{I}$ $\zeta_{\mathbf{y}}=1-10^{-2}, \quad \zeta_{\mathbf{e}}=1-10^{-2}, \quad \mu_{\mathbf{W}}=5\times10^{-2}$ $\nu_{\max}=500, \quad \eta_{\mathbf{y}}(\nu)=\max\{0.1/\nu, 10^{-10}\},$ $\eta_{\lambda_1}(\nu)=\eta_{\lambda_2}(\nu)=1, \quad \epsilon_t=10^{-6}$
Sparse Dictionary Learning	$\mathbf{W}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)=20000\boldsymbol{I}$ $\zeta_{\mathbf{y}}=1-10^{-3}/7, \quad \zeta_{\mathbf{e}}=1-10^{-3}/7, \quad \mu_{\mathbf{W}}=0.25\times10^{-3}$ $\nu_{\max}=500, \quad \eta_{\mathbf{y}}(\nu)=\max\{1.5/(\nu\times0.01), 10^{-8}\},$ $\eta_{\lambda}(\nu)=3\times10^{-2}, \quad \epsilon_t=10^{-6}$
Digital Communications	$\mathbf{W}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)=5\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)=1000\boldsymbol{I}$ $\zeta_{\mathbf{y}}=1-10^{-2}, \quad \zeta_{\mathbf{e}}=1-10^{-2}, \quad \mu_{\mathbf{W}}=3\times10^{-2}$ $\nu_{\max}=500, \quad \eta_{\mathbf{y}}(\nu)=\max\{0.9/\nu, 10^{-3}\}, \quad \epsilon_t=10^{-6}$
Video Separation	$\mathbf{W}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)=65\boldsymbol{I}$ $\zeta_{\mathbf{y}}=1-10^{-1}/5, \quad \zeta_{\mathbf{e}}=0.3, \quad \mu_{\mathbf{W}}=6\times10^{-2}$ $\nu_{\max}=500, \quad \eta_{\mathbf{y}}(\nu)=\max\{0.5/\nu, 10^{-3}\}, \quad \epsilon_t=10^{-6}$
Image Separation	$\mathbf{W}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(1)=\boldsymbol{I}, \quad \mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(1)=100\boldsymbol{I}$ $\zeta_{\mathbf{y}}=1-10^{-1}/15, \quad \zeta_{\mathbf{e}}=0.5, \quad \mu_{\mathbf{W}}=5\times10^{-2}$ $\nu_{\max}=500, \quad \eta_{\mathbf{y}}(\nu)=\max\{0.5/\nu, 10^{-3}\}, \quad \epsilon_t=10^{-6}$

kept the value of $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}$ fixed, as specified in Table B.2. Conversely, for the ablation study on $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}$, we used a fixed value of $\mu_{\mathbf{W}}$ based on the corresponding hyperparameter selection in Table B.2.

Figures B.1a and B.1b present the mean SINR with standard deviation envelope

results with respect to $\mu_{\mathbf{W}}$ for the uncorrelated and correlated source separation settings, respectively. It is worth noting that while the selection of $\mu_{\mathbf{W}} = 5 \times 10^{-2}$ appears to yield better performance for uncorrelated sources, its performance degrades for correlated sources, as indicated by instances where the algorithm diverges. Based on these results, we conclude that selecting $\mu_{\mathbf{W}} = 3 \times 10^{-2}$ is near-optimal in this setting, as it achieves good SINR results for both uncorrelated and correlated sources. Figures B.1c and B.1d demonstrate the SINR performances of CorInfoMax for uncorrelated and correlated source separation experiments, respectively, with respect to the initialization of $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}$. It can be observed that the selections of $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}} = 2000\boldsymbol{I}$ and $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}} = 5000\boldsymbol{I}$ yield favorable results in both settings.

B.5 Ablation Studies on the Effect of Number of Mixtures

The number of mixtures in relation to the number of sources can have a significant impact on the blind source separation problem, potentially influencing the overall performance of the proposed method. To investigate the influence of the number of mixtures on the SINR performance of our proposed method, we conducted experiments where the number of mixtures varied while keeping the number of sources fixed. Specifically, we considered the nonnegative antisparse setting, i.e., $\mathcal{P} = \mathcal{B}_{\infty,+}$, and sparse setting, i.e., $\mathcal{P} = \mathcal{B}_1$. In these experiments, we maintained a fixed number of 5 sources while gradually increasing the number of mixtures from 5 to 10.

Figure B.2a illustrates the overall SINR performance of the nonnegative antisparse CorInfoMax network with respect to the number of mixtures. The results, averaged over 50 realizations, are presented with a standard deviation envelope. For each realization, we randomly generated a mixing matrix with i.i.d. standard normal entries. We observe that the performance of the CorInfoMax network improves monotonically as the number of mixtures increases. This aligns with theoretical expectations that the condition of the random mixing matrix improves with an increasing number of mixtures [Chen and Dongarra, 2005], leading to enhanced numerical performance of the algorithm.

Similarly, Figure B.2b showcases the overall SINR performance of the sparse

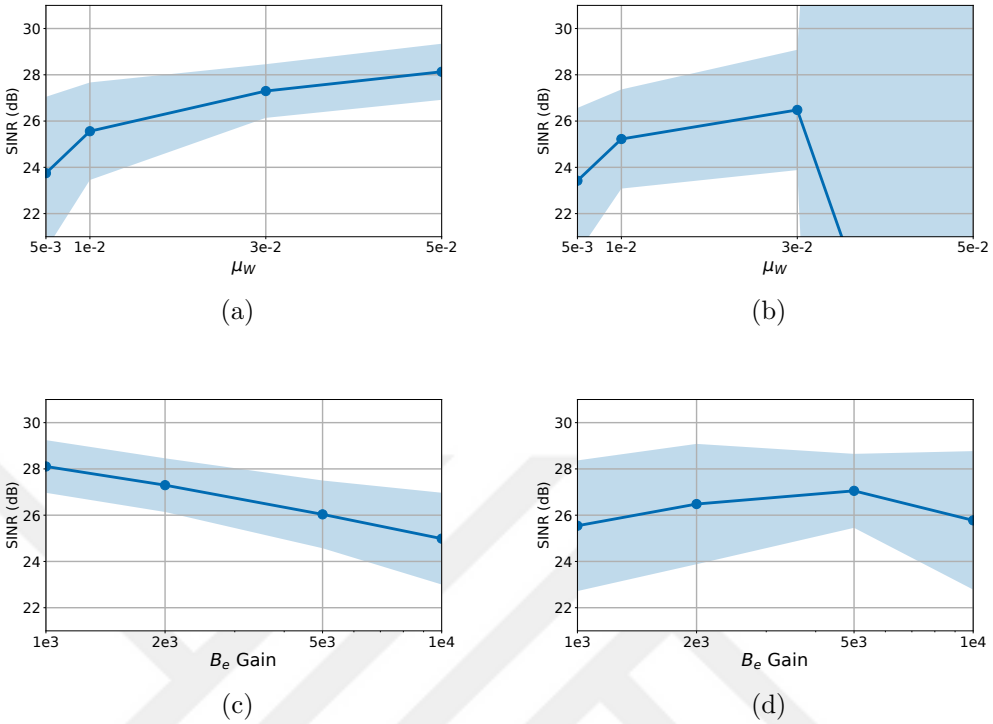
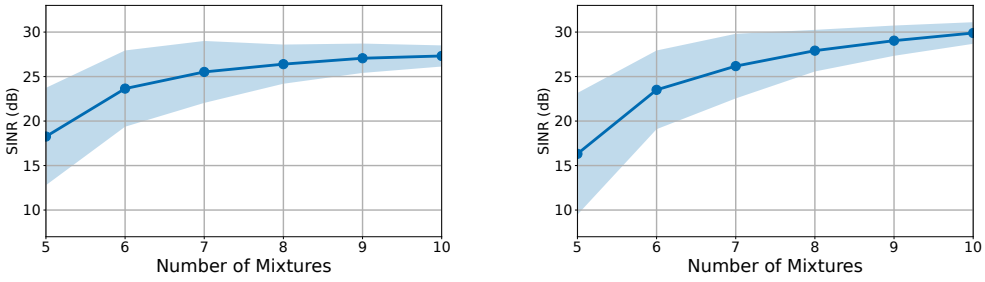


Figure B.1: CorInfoMax ablation studies on the hyperparameter selections of $\mu_{\mathbf{W}}$ and $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}$: mean SINR with standard deviation envelopes corresponding to 50 realizations. (a) ablation study of $\mu_{\mathbf{W}}$ on uncorrelated nonnegative antisparse sources, (b) ablation study of $\mu_{\mathbf{W}}$ on correlated (with $\rho = 0.6$) nonnegative antisparse sources, (c) ablation study of $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}$ on uncorrelated nonnegative antisparse sources, (d) ablation study of $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}$ on correlated (with $\rho = 0.6$) nonnegative antisparse sources,

CorInfoMax network with respect to the number of mixtures. The results, averaged over 50 realizations, are displayed with a standard deviation envelope. Once again, for each realization, we randomly generated a mixing matrix with i.i.d. standard normal entries. It can be observed that the performance of the CorInfoMax network exhibits a monotonic improvement as the number of mixtures increases.

B.6 Computational Complexity of the Proposed Approach

To analyze the complexity of the proposed approach, we can consider the antisparse CorInfoMax network depicted in Figure 5.3 for simplicity. In this analysis, we denote



(a) (b)

Figure B.2: CorInfoMax ablation studies on the impact of number of mixtures with fixed number of sources (5 sources). (a) illustrates the SINR performances of nonnegative antisparse CorInfoMax network as a function of number of mixtures, (b) illustrates the SINR performances of sparse CorInfoMax network as a function of number of mixtures.

$\nu_{\max}$ as the maximum count for the neural dynamic iterations. The complexity of the proposed approach can be evaluated by examining the computational requirements of the output dynamics and weight updates.

Neural Dynamics' Complexity: For one neural dynamic iteration, we observe the following number of operations

- Error vector calculation $\mathbf{e}(k;\nu) = \mathbf{y}(k;\nu) - \mathbf{W}(k)\mathbf{x}(k)$ involves nm multiplications,
- $\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k;\nu)$ involves n^2 multiplications,
- $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k)\mathbf{e}(k;\nu)$ involves n multiplications as $\mathbf{B}_{\mathbf{e}}^{\zeta_{\mathbf{e}}}(k) = \frac{1}{\epsilon}\boldsymbol{I}$.

Consequently, the computation of each output requires $\nu_{max}(mn + n^2 + n) \approx \nu_{max}mn$ multiplications due to neural dynamics iterations.

Weight Updates' Complexity: Regarding the weight updates, we observe the following number of operations.

- $\mathbf{W}(k+1) = \mathbf{W}(k) + \mu_{\mathbf{W}}(k)\mathbf{e}(k)\mathbf{x}(k)^T$ involves $2mn$ multiplication,

- $\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k+1) = \frac{1}{\zeta_{\mathbf{y}}}(\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k) - \frac{1-\zeta_{\mathbf{y}}}{\zeta_{\mathbf{y}}}\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k)\mathbf{y}(k)\mathbf{y}(k)^T\mathbf{B}_{\mathbf{y}}^{\zeta_{\mathbf{y}}}(k))$ involves $n^2 + n^2 + n^2 + 1 + \frac{n(n+1)}{2} = \frac{7n^2+n+2}{2}$ multiplications.

Hence, the weight updates for learning necessitate $2mn + \frac{7n^2+n+2}{2}$ multiplications per input sample.

Considering both computation components, the complexity is primarily governed by the neural dynamics iterations, which approximately require $\mathcal{O}(\nu_{max}mn)$ operations. This complexity is consistent with the findings reported in [Bozkurt et al., 2022] for biologically plausible WSM, NSM, and BSM networks (see also Appendix A.5 in this thesis).

Appendix C

SUPPLEMENTARY ON CORINFOMAX-BASED SUPERVISED NETWORKS

C.1 Test Accuracy Results of CorInfoMax-$\mathcal{B}_{1,+}$ Network

In this section, we provide the accuracy convergence plots of the CorInfoMax-$\mathcal{B}_{1,+}$ networks with single hidden layer whose results are presented in Table 6.1. Figure C.1 presents the test accuracy performance of the CorInfoMax-$\mathcal{B}_{1,+}$ network with a single hidden layer as training epochs progress on the MNIST dataset. Similarly, Figure C.2 showcases the test accuracy performance of the CorInfoMax-$\mathcal{B}_{1,+}$ network with a single hidden layer for the Fashion MNIST dataset. Additionally, Figure C.3 displays the test accuracy performance of the CorInfoMax-$\mathcal{B}_{1,+}$ network with a single hidden layer as training epochs progress on the CIFAR10 dataset.

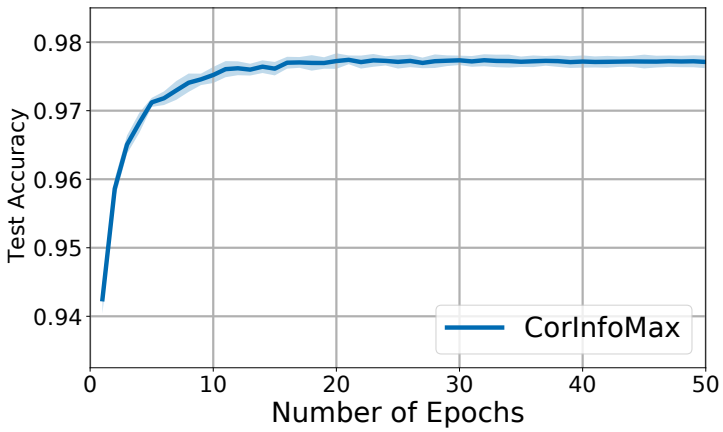


Figure C.1: Test accuracy convergence of CorInfoMax-$\mathcal{B}_{1,+}$ network with single hidden layer as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the MNIST dataset.

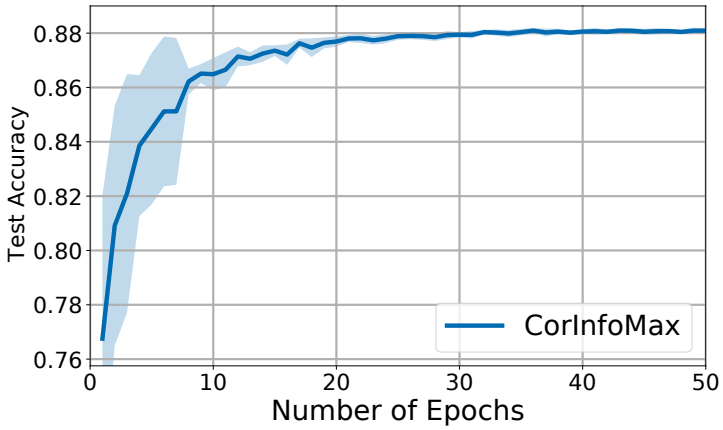


Figure C.2: Test accuracy convergence of CorInfoMax-$\mathcal{B}_{1,+}$ network with single hidden layer as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the Fashion MNIST dataset.

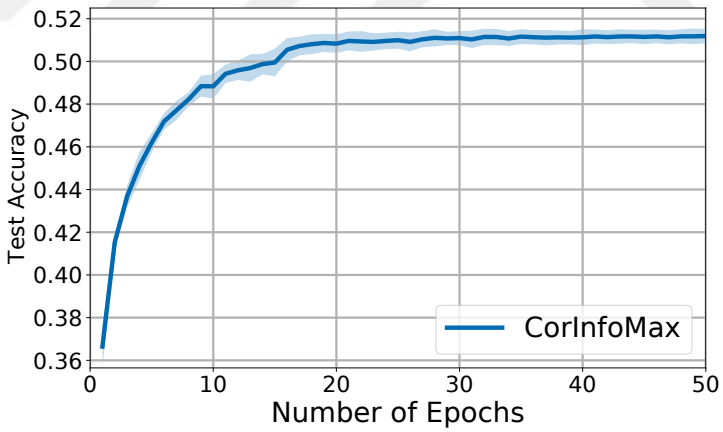


Figure C.3: Test accuracy convergence of CorInfoMax-$\mathcal{B}_{1,+}$ network with single hidden layer as a function of epochs (averaged over $n = 10$ runs associated with the corresponding $\pm$ std envelopes) for the CIFAR10 dataset.

C.2 Hyperparameters

This section provides the details regarding the hyperparameter selections for the supervised CorInfoMax network experiments discussed in Section 6.7. Thus, Table C.1 first provides a description of the notation used for the hyperparameters involved in the CorInfoMax neural dynamics and learning updates.

Table C.1: 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 (6.6).
ϵ_k	Perturbation coefficient for autocorrelation matrices in (6.2) and (6.4).
β'	Nudging parameter for the nudged neural dynamics.
g_{lk}	Leak coefficient for the neural dynamics in (6.22) and (6.41).
$\mu_{\mathbf{u}}$	Learning rate for the neural dynamics in Algorithm 4 and 5.
μ_a	Learning rate for the inhibition neuron in Algorithm 5.
lr decay	Learning rate decay that multiplies the μ_{ff} and μ_{fb} after each epoch.

Table C.2 presents the hyperparameters used to train MNIST, Fashion MNIST, and CIFAR10 datasets using 2-layer CorInfoMax-$\mathcal{B}_{\infty,+}$ networks. Similarly, Table C.3 outlines the hyperparameters employed for training 3-layer CorInfoMax-$\mathcal{B}_{\infty,+}$ networks on MNIST, CIFAR10, and CIFAR100. Additionally, Table C.4 lists the hyperparameters utilized to train 2-layer CorInfoMax-$\mathcal{B}_{1,+}$ networks on MNIST, Fashion MNIST, and CIFAR10.

Table C.2: 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	0.3
$\mu_{\mathbf{u}}$	$\max\{\frac{0.05}{s\times 10^{-2}+1}, 10^{-3}\}$	$\max\{\frac{0.07}{s\times 10^{-2}+1}, 10^{-3}\}$	0.05
lr decay	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$	$\begin{cases} 0.95 & \text{ep.} < 20 \\ 0.9 & 20 \leq \text{ep.} < 25 \\ 0.8 & \text{O/W.} \end{cases}$	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$

C.3 Ablation Studies on the Effect of Hyperparameters

In this section, we analyze the impact of different hyperparameters on the performance of our proposed framework. To investigate this, we conducted a series of simulations using a predefined grid of parameters. We present the results of these experiments, including training and test accuracy, in the following subsections.

C.3.1 Neural Dynamic's Learning Rate and Leakage Conductance g_{lk}

The influence of the learning rate of the neural dynamics and the value of g_{lk} on the training and test accuracies of the CorInfoMax-$\mathcal{B}_{\infty,+}$ network, with a single hidden layer, in the MNIST image classification task is depicted in Table C.5. It can be

Table C.3: 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	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$	$\begin{cases} 0.99 & \text{ep.} < 20 \\ 0.9 & \text{O/W.} \end{cases}$

observed that the performance remains relatively consistent across the chosen range of values.

Similarly, The impact of the learning rate for the neural dynamics and the leakage conductance g_{lk} on the training and test accuracies of the CorInfoMax-$\mathcal{B}_{1,+}$ network with a single hidden layer is illustrated in Table C.6. It is worth noting that the results in the first row of Table C.6 exhibit low accuracy with high variance. Indeed, the divergence of one out of the ten simulations after epoch 35 for this particular combination of hyperparameters is the cause of the observed discrepancy in the results.

Table C.4: 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	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$	$\begin{cases} 0.95 & \text{ep.} < 11 \\ 0.9 & \text{O/W.} \end{cases}$	$\begin{cases} 0.95 & \text{ep.} < 15 \\ 0.9 & \text{O/W.} \end{cases}$

C.3.2 Learning Rates for Synapses and Network Dynamics

The performance variation for different combinations of neural dynamic and synaptic learning rates is presented in Table C.7 for the CorInfoMax-$\mathcal{B}_{\infty,+}$ network with a single hidden layer in the Fashion MNIST classification task. It can be observed that better performance is achieved when using relatively higher feedforward synapse learning rates and relatively smaller feedback synapse learning rates.

C.3.3 Forgetting Factor and Learning Rates

The effect of the forgetting factor $\lambda_{\mathbf{r}}$ in combination with synaptic learning rates on the CIFAR10 classification accuracy for the CorInfoMax-$\mathcal{B}_{\infty,+}$ network with a single hidden layer is presented in Table C.8. Overall, a slightly improved performance is

Table C.5: Train and test accuracies (mean ± 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 C.2.

$\mu_{\mathbf{u}}$	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

Table C.6: Train and test accuracies (mean ± 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 C.4.

$\mu_{\mathbf{u}}$	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

observed when using relatively higher values of μ_{ff} (feedforward synaptic learning rate) and relatively smaller values of μ_{fb} (feedback synaptic learning rate). Similarly, the impact of variations in the same set of hyperparameters for the CorInfoMax-$\mathcal{B}_{1,+}$ network with a single hidden layer on the CIFAR10 classification task is demonstrated in Table C.9. In comparison to the results presented in Table C.8 for CorInfoMax-$\mathcal{B}_{\infty,+}$, the resulting performances show a moderate level of stability.

Table C.7: Train and test accuracies (mean ± 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 C.2.

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

Table C.8: Train and test accuracies (mean ± 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 C.2.

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

C.3.4 Learning Rate Decay, Free Phase Iterations, and Synaptic Learning Rates

The influence of learning rate decay, the number of free phase iterations, and synaptic learning rates on the classification accuracy for the CorInfoMax-$\mathcal{B}_{\infty,+}$ network with two hidden layers in the CIFAR100 classification task is investigated in Table C.10. The best train and test results are achieved by utilizing higher values of μ_{ff} and μ_{fb},

Table C.9: Train and test accuracies (mean ± 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 C.4.

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

along with implementing a learning rate decay during the initial epochs.

C.4 Compute Sources

The simulations were conducted on an HPC cluster using either a single Tesla T4 or a single Tesla V100 GPU. Although the simulation times depend on memory utilization and the number of CPUs, the following list provides approximate training durations for our experiments (which may vary depending on specific hardware configurations and computational resources available):

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

- One epoch of training for CorInfoMax-$\mathcal{B}_{\infty,+}$ networks with two hidden layers on the CIFAR10 dataset, using a batch size of 20, takes approximately 2-3 minutes.

- One epoch of training for CorInfoMax-$\mathcal{B}_{\infty,+}$ networks with two hidden layers on the CIFAR100 dataset, using a batch size of 20, takes approximately 3-4

Table C.10: Train and test accuracies (mean ± 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 C.3. (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

minutes.

- One epoch of training for CorInfoMax-$\mathcal{B}_{1,+}$ networks with a single hidden layer on MNIST, Fashion MNIST, and CIFAR10 datasets, using a batch size of 20, takes approximately 1-2 minutes.