Effect of Top-Down Connections in Hierarchical Sparse Coding

Victor Boutin, Angelo Franciosini, Franck Ruffier, Laurent Perrinet

LETTER Communicated by Zhongming Liu

Effect of Top-Down Connections in Hierarchical Sparse Coding

Victor Boutin
boutin.victor@gmail.com
CNRS, INT, Institut de Neurosciences de la Timone, Aix-Marseille Université, Marseille, France and CNRS, ISM, Aix Marseille Université, Marseille, France

Angelo Franciosini
angelo.FRANCIOSINI@univ-amu.fr
CNRS, Institut de Neurosciences de la Timone, Aix-Marseille Université, 13005 Marseille, France

Franck Ruffier
franck.ruffier@univ-amu.fr
CNRS, Institut des Sciences du Mouvement, Aix-Marseille Université, 13009 Marseille, France

Laurent Perrinet
Laurent.Perrinet@univ-amu.fr
CNRS, Institut de Neurosciences de la Timone, Aix-Marseille Université, 13005 Marseille, France

Hierarchical sparse coding (HSC) is a powerful model to efficiently represent multidimensional, structured data such as images. The simplest solution to solve this computationally hard problem is to decompose it into independent layer-wise subproblems. However, neuroscientific evidence would suggest interconnecting these subproblems as in predictive coding (PC) theory, which adds top-down connections between consecutive layers. In this study, we introduce a new model, 2-layer sparse predictive coding (2L-SPC), to assess the impact of this interlayer feedback connection. In particular, the 2L-SPC is compared with a hierarchical Lasso (Hi-La) network made out of a sequence of independent Lasso layers. The 2L-SPC and a 2-layer Hi-La networks are trained on four different databases and with different sparsity parameters on each layer. First, we show that the overall prediction error generated by 2L-SPC is lower thanks to the feedback mechanism as it transfers prediction error between layers. Second, we demonstrate that the inference stage of the 2L-SPC is faster to converge and generates a refined representation in the second layer compared to the Hi-La model. Third, we show that the 2L-SPC top-down connection accelerates the learning process of the HSC

Neural Computation **32, 2279–2309** (2020)
https://doi.org/10.1162/neco_a_01325

problem. Finally, the analysis of the emerging dictionaries shows that the 2L-SPC features are more generic and present a larger spatial extension.

1 Introduction

Sparse coding (SC) has proven to be one of the most successful methods to find an efficient representation for sensory signals such as natural images. It holds the idea that signals (e.g., images) can be encoded as a linear combination of a few features from a bigger set of features (Elad, 2010). The set of features (also called atoms) is called the dictionary, and SC is thus an inverse problem that is of prominent importance to the machine learning community as it is complex to solve when this dictionary is unknown and as the dimensionality of the signals increases. The pursuit of optimal coding is usually decomposed into two complementary subproblems: inference (coding) and dictionary learning. Inference involves finding an accurate sparse representation of the input data considering the dictionaries are fixed; it could be performed using algorithms like ISTA and FISTA (Beck & Teboulle, 2009), Matching Pursuit (Mallat & Zhang, 1993), Coordinate Descent (Li & Osher, 2009), or ADMM (Heide, Heidrich, & Wetzstein, 2015). Once the representation is inferred, one can learn the atoms from the data using methods like gradient descent (Kreutz-Delgado et al., 2003; Rubinstein, Bruckstein, & Elad, 2010; Sulam, Papyan, Romano, & Elad, 2018) or online dictionary learning (Mairal, Bach, Ponce, & Sapiro, 2009). Consequently, SC offers an unsupervised framework to learn simultaneously the dictionary and the corresponding input representation. SC has been applied with success to image restoration (Mairal, Bach, Ponce, Sapiro, & Zisserman, 2009), feature extraction (Szlam, Kavukcuoglu, & LeCun, 2010), and classification (Perrinet & Bednar, 2015; Yang, Zhang, Yang, & Zhang, 2011).

Interestingly, SC is also a field of interest for computational neuroscience. Olshausen & Field (1997) first demonstrated that adding a sparse prior to a shallow neural network was sufficient to account for the emergence of neurons whose receptive fields (RFs) are spatially localized, bandpass and oriented filters, analogous to those found in the primary visual cortex (V1) of mammals (Hubel & Wiesel, 1962). Because most SC algorithms are limited to single-layer networks, this method cannot easily be extended to model the hierarchical structure of the cortical areas constituting the visual pathways. Still, some solutions have been proposed to tackle this problem of hierarchical sparse coding (HSC) as a global optimization problem (Aberdam, Sulam, & Elad, 2019; Makhzani & Frey, 2013, 2015; Sulam, Aberdam, Beck, & Elad, 2019; Sulam et al., 2018; Zeiler, Taylor, & Fergus, 2011). However, these methods are looking for optimal solutions of HSC without bearing in mind their plausibility in terms of neuronal implementation. Consequently, the quest for an efficient HSC formulation that is compatible with such a neural implementation remains open.

Rao and Ballard (1999) introduced the predictive coding (PC) framework to model the effect of the interaction of two cortical areas in the visual cortex. PC intends to solve the inverse problem of vision by combining bottom-up (feedforward) and top-down (feedback) activities. In PC, feedback connections carry a prediction of the neural activity of the afferent lower cortical area, while the feedforward connection carries a prediction error to the next higher cortical area. In such a framework, the activity of the neural population is updated to minimize the unexpected component of the neural signal (Friston, 2010). PC has been applied to supervised object recognition (Wen et al., 2018; Han et al., 2018; Spratling, 2017) and unsupervised prediction of future video frames (Lotter, Kreiman, & Cox, 2016). Interestingly, it is flexible enough to allow the introduction of a sparse prior within each layer. Therefore, one might consider PC as a bio-plausible formulation of the HSC problem.

Interestingly, when recast into an HSC problem, SC and PC could be used to model different types of computation in the brain. On the one hand, SC might be considered as an intralayer computational mechanism that exacerbates competition between neurons by selecting only the strongly activated ones. This mechanism, called explaining away, could be used to model intracortical recurrent connectivity. On the other hand, PC is describing the flow of information between consecutive layers and could be used to model intercortical feedback connections. To the best of our knowledge, no study has revealed the advantages of interconnecting SC layers using the PC principle. What is the effect of the top-down connection of PC? What are the consequences in terms of computations and convergence? What are the qualitative differences concerning the learned atoms and representations?

The objective of this study is to experimentally answer these questions and show that the PC framework could be successfully used for improving the solutions to HSC problems in a 2-layer network. The letter is organized as follows. We start our study by defining the two mathematical formulations to solve the HSC problem: the hierarchical Lasso (Hi-La), which consists of stacking two independent Lasso subproblems, and the 2-layer sparse predictive coding (2L-SPC), which leverages PC into a deep and sparse network of bidirectionally connected layers. To experimentally compare both models, we train the 2L-SPC and Hi-La networks on four databases. First, we vary the sparsity of each layer and compare the overall prediction error for the two models, analyzing it layer-by-layer to understand their respective roles. Second, we analyze the number of inference iterations needed for the state variables of each network to reach their stability. Third, we study the evolution of the representations generated by the Hi-La and the 2L-SPC during their inference process. Fourth, we compare the convergence of both models during the dictionary learning stage. Finally, we discuss the differences between the features that both networks learned in light of their activation probability and their spatial extension.

2 Methods

In the following mathematical description, italic letters (e.g., α) are used as symbols for scalars, bold lowercase letters (e.g., $\boldsymbol{x}$) for column vectors, and bold uppercase letters (e.g., $\mathbf{D}$) for matrices, and $\nabla_x \mathcal{L}$ denotes the gradient of a function $\mathcal{L}$ with respect to x.

2.1 Background. The core objective of a hierarchical sparse coding (HSC) model with L-layers is to infer for each input image $\boldsymbol{x}^{(k)}$ (with $k \leq N$ and N is the number of images in the batch) the internal state variables $\{\boldsymbol{\gamma}_i^{(k)}\}_{i=1}^L$ and to learn the dictionaries $\{\boldsymbol{D}_i\}_{i=1}^L$ that fit the constrained generative model formulated by equation 2.1:

$$\begin{cases} \boldsymbol{x}^{(k)} = \mathbf{D}_1^T \boldsymbol{\gamma}_1^{(k)} + \boldsymbol{\epsilon}_1^{(k)} & \text{s.t.} \quad \|\boldsymbol{\gamma}_1^{(k)}\|_0 < \alpha_1 \text{ and } \boldsymbol{\gamma}_1^{(k)} > 0 \\ \boldsymbol{\gamma}_1^{(k)} = \mathbf{D}_2^T \boldsymbol{\gamma}_2^{(k)} + \boldsymbol{\epsilon}_2^{(k)} & \text{s.t.} \quad \|\boldsymbol{\gamma}_2^{(k)}\|_0 < \alpha_2 \text{ and } \boldsymbol{\gamma}_2^{(k)} > 0 \\ \dots \\ \boldsymbol{\gamma}_{L-1}^{(k)} = \mathbf{D}_L^T \boldsymbol{\gamma}_L^{(k)} + \boldsymbol{\epsilon}_L^{(k)} & \text{s.t.} \quad \|\boldsymbol{\gamma}_L^{(k)}\|_0 < \alpha_L \text{ and } \boldsymbol{\gamma}_L^{(k)} > 0 \end{cases} . \tag{2.1}$$

The term $\boldsymbol{\epsilon}_i^{(k)}$ is historically called "prediction error," but it is actually quantifying the local reconstruction error between $\mathbf{D}_i^T \boldsymbol{\gamma}_i^{(k)}$ and $\boldsymbol{\gamma}_{i-1}^{(k)}$. The vector $\boldsymbol{\gamma}_i^{(k)}$ could be viewed as the projection of $\boldsymbol{\gamma}_{i-1}^{(k)}$ in the basis described by the atoms composing $\mathbf{D}_i$. The sparsity of the internal state variables, specified by the ℓ_0 pseudo-norm, is constrained at layer i by the scalar α_i. In practice, we use four-dimensional tensors to represent both vectors and matrices. The input $\boldsymbol{x}^{(k)}$ is a tensor of size $[1, c_x, w_x, h_x]$, where c_x is the number of channels of the image (i.e., 1 for grayscale images, and 3 for colored ones), and w_x and h_x are, respectively, the width and height of the image. In the mathematical description above, we can flatten $\boldsymbol{x}^{(k)}$ as a vector of dimension $[c_x \times w_x \times h_x]$. Furthermore, we impose a 2-dimensional convolutional structure to the parameters $\{\mathbf{D}_i\}_{i=1}^L$. If one describes the dictionary as a 4-dimensional tensor of size $[n_d, c_d, w_d, h_d]$, one can derive $\mathbf{D}_i$ as a matrix of n_d local features (size: $c_d \times w_d \times h_d$) that cover every possible location of the input $\boldsymbol{x}^{(k)}$ (Sulam et al., 2018). In other words, $\mathbf{D}_i$ is a Toeplitz matrix. For the sake of concision in our mathematical descriptions, we use matrix/vector multiplication in place of convolution as it is mathematically strictly equivalent. Notice that in equation 2.1, we have constrained the latent variables to be positive only (i.e., $\boldsymbol{\gamma}_i^{(k)} > 0$). This constraint is not a priori imposed by HSC problems but nonnegative SC is known to keep the same expressiveness as standard SC (Papyan, Romano, & Elad, 2017) and allows us to strengthen the link with both convolutional neural networks (CNNs; see section 2.5) and neuroscience. Indeed, replaced in a biological context, the element of $\boldsymbol{\gamma}_i$ could be interpreted as firing rates, and $\mathbf{D}_i$ could be viewed

as the synaptic weights between two neural populations at layers $i-1$ and i.

2.2 From Hierarchical Lasso .... One possibility for solving equation 2.1 while keeping the locality of the processing required by a plausible neural implementation is to minimize a loss independently for each layer. First, we transform the constrained problem to a regularization one, such that the loss function is equal to the addition of the squared ℓ_2-norm of the prediction error with a sparsity penalty. To guarantee a convex cost, we relax the ℓ_0 sparsity into a ℓ_1-penalty such that, finally, this defines a loss function for each layer in the form of a standard Lasso problem (see equation 2.2), that could be minimized using proximal gradient-based methods:

$$\begin{aligned}
&\mathcal{F}\big(\mathbf{D}_i, \boldsymbol{\gamma}_i^{(k)}\big) = \mathcal{G}\big(\mathbf{D}_i, \boldsymbol{\gamma}_i^{(k)}\big) + \lambda_i \|\boldsymbol{\gamma}_i^{(k)}\|_1 \text{ with} \\
&\mathcal{G}\big(\mathbf{D}_i, \boldsymbol{\gamma}_i^{(k)}\big) = \frac{1}{2}\|\boldsymbol{\gamma}_{i-1}^{(k)} - \mathbf{D}_i^T \boldsymbol{\gamma}_i^{(k)}\|_2^2.
\end{aligned} \tag{2.2}$$

In particular, we use the iterative shrinkage thresholding algorithm (ISTA) to minimize $\mathcal{F}$ with respect to $\boldsymbol{\gamma}_i^{(k)}$ (see equation 2.3) as it is proven to be computationally efficient (Beck & Teboulle, 2009). In practice, we use an accelerated version of the ISTA algorithm called FISTA. We will operate over steps t until a criterion is met and at every step, from the first layer ($i = 0$) to the last ($i = L$). In equation 2.3, we have removed image indexation to keep the notation concise:

$$\begin{aligned}
\boldsymbol{\gamma}_i^{t+1} &= \mathcal{T}_{\eta_{c_i}\lambda_i}\big(\boldsymbol{\gamma}_i^t - \eta_{c_i} \nabla_{\boldsymbol{\gamma}_i^t} \mathcal{G}\big) \\
&= \mathcal{T}_{\eta_{c_i}\lambda_i}\big(\boldsymbol{\gamma}_i^t + \eta_{c_i} \mathbf{D}_i(\boldsymbol{\gamma}_{i-1}^{t+1} - \mathbf{D}_i^T \boldsymbol{\gamma}_i^t)\big).
\end{aligned} \tag{2.3}$$

In equation 2.3, $\mathcal{T}_\alpha(\cdot)$ denotes the nonnegative soft-thresholding operator (see equation 2.4), η_{c_i} is the time constant of the inference process, and $\boldsymbol{\gamma}_i^t$ is the state variable $\boldsymbol{\gamma}_i$ at step t:

$$\mathcal{T}_\alpha(x) = \begin{cases} x - \alpha & \text{if} \quad x \geq \alpha \\ 0 & \text{if} \quad x \leq \alpha \end{cases}. \tag{2.4}$$

Interestingly, one can interpret equation 2.3 as one loop of a recurrent layer that we will call the Lasso layer (Gregor & LeCun, 2010). Following equation 2.3, $\mathbf{D}_i^T$ is a decoding dictionary that back-projects $\boldsymbol{\gamma}_i$ into the space of the $(i-1)$th layer. This back-projection is used to elicit an error with respect to $\boldsymbol{\gamma}_{i-1}$, and that will be encoded by $\mathbf{D}_i$ to update the state variables $\boldsymbol{\gamma}_i$. Finally, Lasso layers can be stacked together to form a hierarchical Lasso (Hi-La) network (see Figure 1 without the left blue arrow). In our convolutional case and for the proximal operator corresponding to the Lasso, it was

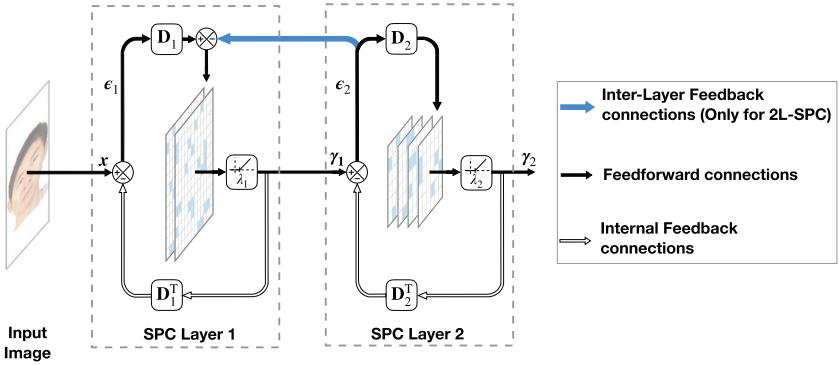

Figure 1: Inference update scheme for the 2L-SPC network. $\boldsymbol{x}$ is the input image, and λ_i tunes the sparseness level of $\boldsymbol{\gamma}_i$. The encoding and decoding dictionaries are denoted $\mathbf{D}_i$ and $\mathbf{D}_i^T$, respectively. The sparse maps ($\boldsymbol{\gamma}_i$) are updated through a bidirectional dynamical process (plain and empty arrows). This recursive process alone describes the Hi-La network. If we add the top-down influence, the interlayer feedback connection (blue arrow), it then becomes a 2L-SPC network.

proven that FISTA has the advantage of converging faster than other sparse coding algorithms such as coordinate descent (Chalasani, Principe, & Ramakrishnan, 2013). The inference of the overall Hi-La network consists in updating recursively all the sparse maps $\boldsymbol{\gamma}_i$ until they reach a stable point.

2.3 ... to Hierarchical Predictive Coding. Another alternative for solving equation 2.1 is to use the predictive coding (PC) theory. Unlike the Lasso loss function, PC is not only minimizing the bottom-up prediction error; it also adds a top-down prediction error that takes into consideration the influence of the upper layer on the current layer (see equation 2.5). In other words, finding the $\boldsymbol{\gamma}_i$ that minimizes $\mathcal{L}$ involves finding a trade-off between a representation that best predicts the lower-level activity and another one that is best predicted by the upper layer:

$$\mathcal{L}\big(\mathbf{D}_i, \boldsymbol{\gamma}_i^{(k)}\big) = \mathcal{G}\big(\mathbf{D}_i, \boldsymbol{\gamma}_i^{(k)}\big) + \frac{1}{2}\|\boldsymbol{\gamma}_i^{(k)} - \mathbf{D}_{i+1}^T \boldsymbol{\gamma}_{i+1}^{(k)}\|_2^2 + \lambda_i \|\boldsymbol{\gamma}_i^{(k)}\|_1. \tag{2.5}$$

For consistency, we also use the ISTA algorithm to minimize $\mathcal{L}$ with respect to $\boldsymbol{\gamma}_i$. A similar update scheme follows in equation 2.6 (without image indexation for concision):

$$\begin{aligned} \boldsymbol{\gamma}_i^{t+1} &= \mathcal{T}_{\eta_{c_i}\lambda_i}\big(\boldsymbol{\gamma}_i^t - \eta_{c_i} \nabla_{\boldsymbol{\gamma}_i^t} \mathcal{L}\big) \\ &= \mathcal{T}_{\eta_{c_i}\lambda_i}\big(\boldsymbol{\gamma}_i^t + \eta_{c_i} \mathbf{D}_i(\boldsymbol{\gamma}_{i-1}^{t+1} - \mathbf{D}_i^T \boldsymbol{\gamma}_i^t) - \eta_{c_i}(\boldsymbol{\gamma}_i^t - \mathbf{D}_{i+1}^T \boldsymbol{\gamma}_{i+1}^t)\big). \end{aligned} \tag{2.6}$$

Algorithm 1: Alternation of Inference and Learning for Training and Testing.

for $\boldsymbol{x}^{(k)}$ in the training set **do**

  $\forall i, \boldsymbol{\gamma}_i^{(k)} = \mathbf{0}$ # initialization

  while *convergence not reached* **do**

    for $i = 1$ **to** L **do**

      $\nabla_{\boldsymbol{\gamma}_i^t}\mathcal{L} = \mathbf{D}_i(\boldsymbol{\gamma}_{i-1}^{t+1} - \mathbf{D}_i^T\boldsymbol{\gamma}_i^t) - (\boldsymbol{\gamma}_i^t - \mathbf{D}_{i+1}^T\boldsymbol{\gamma}_{i+1}^t)$

      $\boldsymbol{\gamma}_i^{(k)} \leftarrow \mathcal{T}_{\eta_{c_i}\lambda_i}\big(\boldsymbol{\gamma}_i^{(k)} - \eta_{c_i}\nabla_{\boldsymbol{\gamma}_i^{(k)}}\mathcal{L}\big)$ # inference

  for $i = 1$ **to** L **do**

    $\mathbf{D}_i \leftarrow \mathbf{D}_i - \eta_i\nabla_{\boldsymbol{D}_i}\mathcal{L}$ # learning (only during training)

Figure 1 shows how we can interpret this update scheme as a recurrent loop. This recurrent layer, called the sparse predictive coding (SPC) layer, forms the building block of the 2-layer sparse predictive coding (2L-SPC) network (see algorithm 2 in the appendix for the detailed implementation of the 2L-SPC inference). The only difference from the Hi-La architecture is that the 2L-SPC includes this top-down, feedback interlayer connection to materialize the influence coming from upper layers (see the blue arrow in Figure 1). Equations 2.3 and 2.6 explain the link between the encoding and the decoding dictionary: the decoding dictionary ($\mathbf{D}_i^T$) is the transpose of the encoding dictionary ($\mathbf{D}_i$) because of the derivation of the convolution $\mathbf{D}_i^T\boldsymbol{\gamma}_i^t$ with regard to $\boldsymbol{\gamma}_i^t$. Note that other approaches exist for which the decoding and encoding dictionaries are decoupled (Han et al., 2018; Wen et al., 2018). We do not consider them here.

2.4 Coding Stopping Criterion and Unsupervised Learning. For both networks, the inference process is finalized once the relative variation of $\boldsymbol{\gamma}_i^t$ with respect to $\boldsymbol{\gamma}_i^{t-1}$ is below a threshold denoted T_{stab}. In practice, the number of iterations needed to reach the stopping criterion is between 30 and 200 (see Figure 4 for details). Once the convergence is achieved, we update the dictionaries using gradient descent (see algorithm 1). Sulam et al. (2018) demonstrated that this alternation of inference and learning offers a reasonable guarantee for convergence. The learning of both Hi-La and 2L-SPC involves minimizing the problem over batches of the training data set as defined in equation 2.7. Note that the learning occurs during the training

phase only and the inference process is the same during both training and testing phases:

$$\min_{\{\mathbf{D}_i\}} \left(\frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{L} \mathcal{F}(\mathbf{D}_i, \boldsymbol{\gamma}_i^{(k)}) \right). \tag{2.7}$$

For both models, dictionaries are randomly initialized using the standard normal distribution (mean 0 and variance 1), and all sparse maps are initialized to zero at the beginning of the inference process. After every dictionary update, we regularize it by ℓ_2-normalizing each column of the dictionary (i.e., we normalize according to the dimension n_d as defined in section 2.1). Interestingly, although the inference update scheme is different for the two models, the dictionary learning loss is the same in both cases since the top-down prediction error term in $\mathcal{L}$ does not depend on $\boldsymbol{D}_i$ (see equation 2.7). This loss is then a good evaluation point to assess the impact of both the 2L-SPC and Hi-La inference processes on the layer prediction error $\boldsymbol{\epsilon}_i$. We used PyTorch 1.0 to implement, train, and test all the models described above. The code of the two models and the simulations of this letter are available at www.github.com/VictorBoutin/SPC_2L/.

2.5 Link between the Hi-La, the 2L-SPC, and CNNs. Papyan et al. (2017) exhibited the link between CNNs and the nonnegative convolutional HSC problem. In this section, we show that this relationship could be extended to both Hi-La and 2L-SPC if we consider the first time step of their update scheme (see equations 2.3 and 2.6 for Hi-La and 2L-SPC, respectively). At $t = 1$, since all latent variables have been initialized at $\mathbf{0}$, the feedback term of the 2L-SPC is removed, and the updates of both networks are reducible to the same feedforward process (see equation 2.8) with an activation function being the soft-thresholding operator:

$$\boldsymbol{\gamma}_i^{t=1} = \begin{cases} \mathcal{T}_{\eta_{c_i}\lambda_i}\left(\eta_{c_i}\mathbf{D}_i\boldsymbol{x}\right) & = \eta_{c_i}\mathcal{T}_{\lambda_i}\left(\mathbf{D}_i\boldsymbol{x}\right) \quad \text{if} \quad i = 1 \\ \mathcal{T}_{\eta_{c_i}\lambda_i}\left(\eta_{c_i}\mathbf{D}_i\boldsymbol{\gamma}_{i-1}^{t=1}\right) & = \eta_{c_i}\mathcal{T}_{\lambda_i}\left(\mathbf{D}_i\boldsymbol{\gamma}_{i-1}^{t=1}\right) \quad \text{if} \quad i \in [\![2; L]\!] \end{cases}. \tag{2.8}$$

The soft-thresholding operator could be viewed as a biased rectified linear unit (ReLU) in which every entry of the bias vector is the same (i.e., the entries are all equal to λ_i). Note that the ReLu is the most popular activation function used for CNNs. Therefore, the first flows of information of both the Hi-La and the 2L-SPC are equivalent to feedforward CNNs. In the following inference iterations, both models are refining the latent variables $\boldsymbol{\gamma}_i$ to make them fit the inverse problem defined in equation 2.1.

3 Experimental Settings: Data Sets and Parameters

We use four databases to train and test both networks.

STL-10. The STL-10 database (Coates, Ng, & Lee, 2011) has 100,000 colored images of size 96×96 pixels (px) representing 10 classes of objects (e.g., airplane, bird). STL-10 presents a high diversity of object viewpoints and backgrounds. This set is partitioned into a training set composed of 90,000 images and a testing set of 10,000 images.

CFD. The Chicago Face Database (CFD) (Ma, Correll, & Wittenbrink, 2015) consists of 1804 high-resolution (2444×1718 px), color, standardized photographs of male and female faces of varying ethnicity between the ages of 18 and 40 years. We resized the pictures to 170×120 px to keep the computational time reasonable. The CFD database is partitioned into batches of 10 images. This data set is split into a training set composed of 721 images and a testing set of 486 images.

MNIST. MNIST (LeCun, 1998) is composed of 28×28 px, 70,000 grayscale images representing handwritten digits. We decomposed this data set into batches of 32 images, split into a training set composed of 60,000 digits and a testing set of 10,000 digits.

AT&T. The AT&T database (AT&T, 1994) has 400 grayscale images of size 92×112 pixels (px) representing faces of 40 distinct persons with different lighting conditions, facial expressions, and details. This set is partitioned into batches of 20 images. The training set is composed of 330 images (33 subjects), and the testing set is composed of 70 images (7 subjects).

All of these databases are preprocessed using local contrast normalization (LCN) first and then whitening (see Figure 17 for sample examples on all databases). LCN is inspired by neuroscience and consists of a local subtractive and divisive normalization (Jarrett, Kavukcuoglu, & LeCun, 2009). In addition, we use whitening to reduce dependency between pixels (Olshausen & Field, 1997).

To draw a fair comparison between the 2L-SPC and Hi-La models, we train both models using the same set of hyperparameters. We summarize these parameters in Table 1 for the STL-10, MNIST, and CFD databases and in section A.3 for the ATT database. Note that the parameter η_{c_i} is omitted in the table because it is computed as the inverse of the largest eigenvalue of $\mathbf{D}_i^T\mathbf{D}_i$ (Beck & Teboulle, 2009). To learn the dictionary $\mathbf{D}_i$, we use stochastic gradient descent on the training set only, with a learning rate η_{L_i} and a momentum equal to 0.9. In this study, we consider only 2-layered networks, and we vary the sparsity parameters of each layer (λ_1 and λ_2) to assess their effect on both the 2L-SPC and the Hi-La networks.

Table 1: Network Architectures, Training, and Simulation Hyperparameters.

		Databases		
		STL-10	CFD	MNIST
Network parameters	$\boldsymbol{D}_1$ size	[64, 1, 8, 8] (2)	[64, 3, 9, 9] (3)	[32, 1, 5, 5] (2)
	$\boldsymbol{D}_2$ size	[128, 64, 8, 8] (1)	[128, 64, 9, 9] (1)	[64, 32, 5, 5] (1)
	T_{stab}	1e-4	5e-3	5e-4
Training parameters	Number of epochs	10	250	100
	η_{L_1}	1e-4	1e-4	5e-2
	η_{L_2}	5e-3	5e-3	1e-3
Simulation parameters	λ_1 range	[0.2 : 0.6 :: 0.1, 1.6]	[0.3 : 0.7 :: 0.1, 1.8]	[0.1 : 0.3 :: 0.05, 0.3]
	λ_2 range	[0.4, 1.4 : 1.8 :: 0.1]	[0.5, 1 : 1.8 :: 0.2]	[0.2, 0.2 : 0.4 :: 0.05]

Note: The sizes of the convolutional kernels are shown in the format: [# features, # channels, width, height] (stride). To describe the range of explored parameters during simulations, we use the format [0.3 : 0.7 :: 0.1, 0.5], which means that we vary λ_1 from 0.3 to 0.7 by step of 0.1 while λ_2 is fixed to 0.5.

4 Results

For cross-validation, we ran all the simulations presented in this section seven times, each time with a different random seed for the initialization of the dictionary. We define the central tendency of our curves by the median of the runs and its variation by the median absolute deviation (MAD) (Pham-Gia & Hung, 2001). We prefer this measure to the classical mean $\pm$ standard deviation because a few measures did not exhibit a normal distribution. All presented curves are obtained on the testing set.

4.1 2L-SPC Converges to a Lower Prediction Error. As a first analysis, we report the cost $\mathcal{F}(\boldsymbol{D}_i, \boldsymbol{\gamma}_i)$ (see equation 2.2) for each layer and for both networks. To refine our analysis, we decompose for each layer this cost into a quadratic cost (i.e., the ℓ_2 term in $\mathcal{F}$, that is, $\mathcal{G}$) and a sparsity cost (i.e., the ℓ_1 term in $\mathcal{F}$), and we monitor these quantities when varying the first- and second-layer sparse penalties (see Figure 2). For scaling reasons and because the error bars are small, we cannot display them on Figure 2; we thus include them in Figure 8. For all the simulations shown in Figure 2, we observe that the total cost (i.e., $\mathcal{F}(\boldsymbol{D}_1, \boldsymbol{\gamma}_1) + \mathcal{F}(\boldsymbol{D}_2, \boldsymbol{\gamma}_2)$) is lower for the 2L-SPC than for the Hi-La model. As expected, in both models, the total cost increases when we increase λ_1 or λ_2.

For all databases, Figure 2 shows that the feedback connection of the 2L-SPC tends to increase the first-layer quadratic cost. For example, when λ_1 is increased, the average variation of the first-layer quadratic cost of the 2L-SPC compared to the one of the Hi-La is +126% for STL-10, +110% for CFD, +100% for MNIST, and +73% for AT&T. On the contrary, the

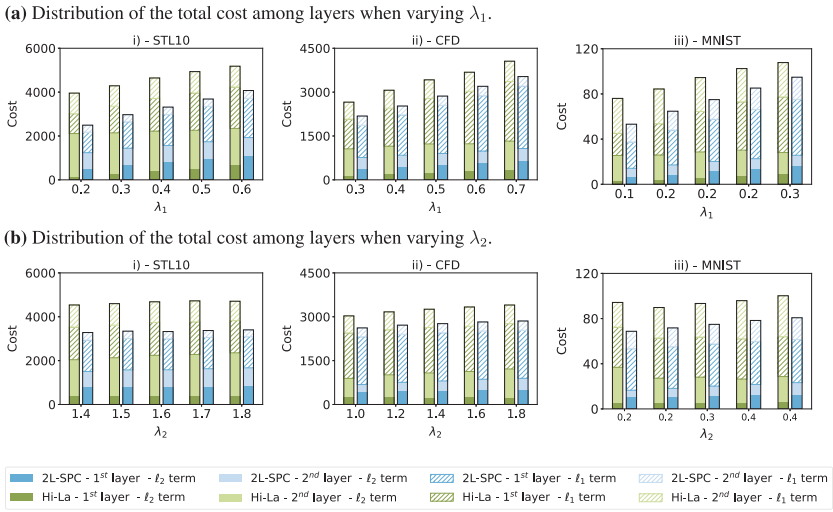

Figure 2: Evolution of the cost, evaluated on the testing set, for both 2L-SPC and Hi-La networks and trained on STL-10, CFD and MNIST databases. We vary the first layer sparsity in the (a) top three graphs and the second layer sparsity in the (b) bottom three graphs. For each layer, the cost is decomposed into a quadratic cost (i.e., ℓ_2 term) represented with solid bars and a sparsity cost (i.e., ℓ_1 term) represented with hashed bars. First-layer cost is represented with darker colors and second-layer cost with lighter colors.

second-layer quadratic cost is strongly decreasing when the feedback connection is activated. In particular, when λ_1 is increased, the average variation of the second-layer quadratic cost of the 2L-SPC compared to the one of the Hi-La is -57% for STL-10, -58% for CFD, -61% for MNIST, and -44% for AT&T. These observations are holding when the second-layer sparse penalty is increased. This is expected: while the Hi-La first layer is fully specialized in minimizing the quadratic cost with the lower level, the 2L-SPC finds a trade-off between lower, and higher-level quadratic cost.

In addition, when λ_1 is increased, the Hi-La first-layer quadratic cost is increasing faster ($+337\%$ for STL-10, $+152\%$ for CFD, $+214\%$ for MNIST, and $+260\%$ for AT&T) than the 2L-SPC first-layer quadratic cost ($+117\%$ for STL-10, $+83\%$ for CFD, $+148\%$ for MNIST, and $+147\%$ for AT&T). This phenomenon is amplified if we consider the evolution of the first-layer sparsity cost when increasing λ_1. The first-layer sparsity cost of the Hi-La exhibits a stronger increase ($+108\%$ for STL-10, $+99\%$ for CFD, $+150\%$ for MNIST, and $+60\%$ for AT&T) than the one of the 2L-SPC ($+92\%$ for STL-10, $+94\%$ for CFD, $+112\%$ for MNIST, and $+47\%$ for AT&T). This suggests

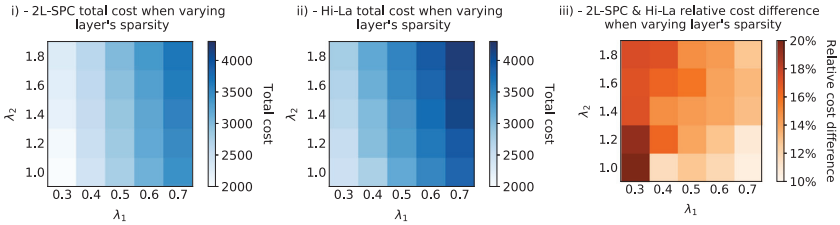

Figure 3: Heat maps of the total cost when varying layers' sparsity for 2L-SPC (i) and Hi-La (ii) on the CFD database. (iii) The heat map of the relative difference between the Hi-La and the 2L-SPC total cost when varying the layers' sparsity.

that the extra penalty induced by the increase of λ_1 is better mitigated by the 2L-SPC.

When λ_2 is increased, the quadratic cost of the first layer of the Hi-La model is almost stable (1% for STL-10, 0% for CFD, +2% for MNIST, and 0% for AT&T), whereas the 2L-SPC first layer ℓ_2 cost is increasing (+6% for STL-10, +16% for CFD, +20% for MNIST, and +8% for AT&T). The explanation here is straightforward: while the first layer of the 2L-SPC includes the influence of the upper layer, the Hi-La does not have such a mechanism. It suggests that the feedback connection of the 2L-SPC transfers a part of the extra penalty coming from the increase of λ_2 in the first-layer quadratic cost.

Figures 3i and 3ii show the mapping of the total cost when we vary the sparsity of each layer for the 2L-SPC and Hi-La, respectively. These heat maps confirm what has been observed in Figure 2 and they extend it to a larger range of sparsity values: both models are more sensitive to a variation of λ_1 than to a change in λ_2. Figure 3iii is a heat map of the relative difference between the 2L-SPC and the Hi-La total cost. It shows that the minimum relative difference between 2L-SPC and Hi-La (10.6%) is reached when λ_1 is maximal and λ_2 is minimal, and the maximum relative difference (19.9%) is reached when both λ_1 and λ_2 are minimal. It suggests that the previously observed mitigation mechanism originated by the feedback connection is more efficient when the sparsity of the first layer is lower.

All these observations point in the same direction: the 2L-SPC framework mitigates the total cost with a better distribution of the cost among layers. This mechanism is even more pronounced when the sparsity of the first layer is lower. Surprisingly, while the feedback connection of the 2L-SPC imposes more constraints on the state variables, it also happens to generate less total cost.

4.2 2L-SPC Has a Faster Inference Process. One may wonder if this better convergence is not achieved at the cost of a slower inference process. To address this concern, we report for both models the number of

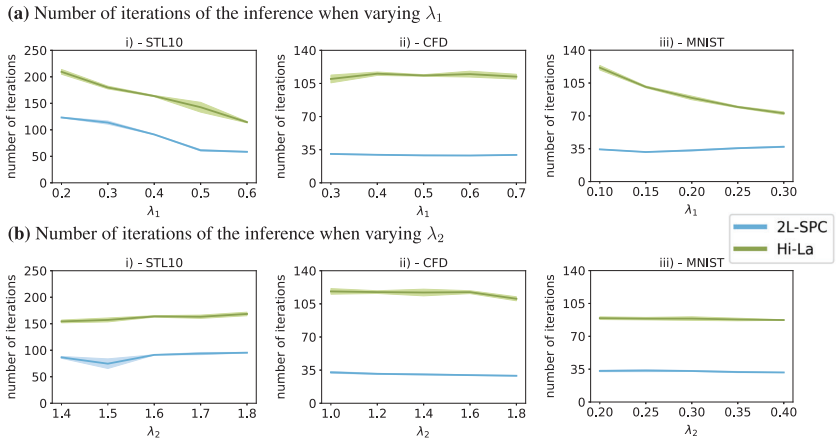

Figure 4: Evolution of the number of iterations needed to reach stability criterion for both 2L-SPC and Hi-La networks on the testing sets of STL-10, CFD, and MNIST databases. We vary the sparsity in the (a) first layer in the top three graphs and sparsity in the (b) second layer in the bottom three graphs. Shaded areas correspond to the mean absolute deviation on seven runs.

iterations the inference process needs to converge toward a stable state on the testing set. Figure 4 shows the evolution of this quantity, for STL-10, CFD, and MNIST databases (see section A.5 for the AT&T database), when varying both layers' sparsity. For all the simulations, results demonstrate that the 2L-SPC needs less iteration than the Hi-La model to converge toward a stable state. We also observe that data dispersion is in general more pronounced for the Hi-La model. In addition to converging to lower cost, the 2L-SPC is also decreasing the number of iterations in the inference process to converge toward a stable state.

4.3 2L-SPC Refines the Second Layer. We now study the evolution of the quadratic term of the prediction error during the inference process. As a representative example, we report these errors for both layers and both models when they are trained on the STL-10 database with $\lambda_1 = 0.4$ and $\lambda_2 = 1.4$ (see Figure 5a). We distinguish three states, denoted A, B, and C, corresponding to the inference at iterations 1, 7, and 20, respectively. At state A, the inference process has just started, and the first-layer prediction error is very high in contrast to the second-layer prediction error. Around state B and for both models, all layers' prediction errors are getting closer to each other. At state C, even if the inference process has not yet fully converged, the evolution of the layer's prediction errors gets smoother until a final error is reached (at iteration 155 for the Hi-La and 75 for the 2L-SPC).

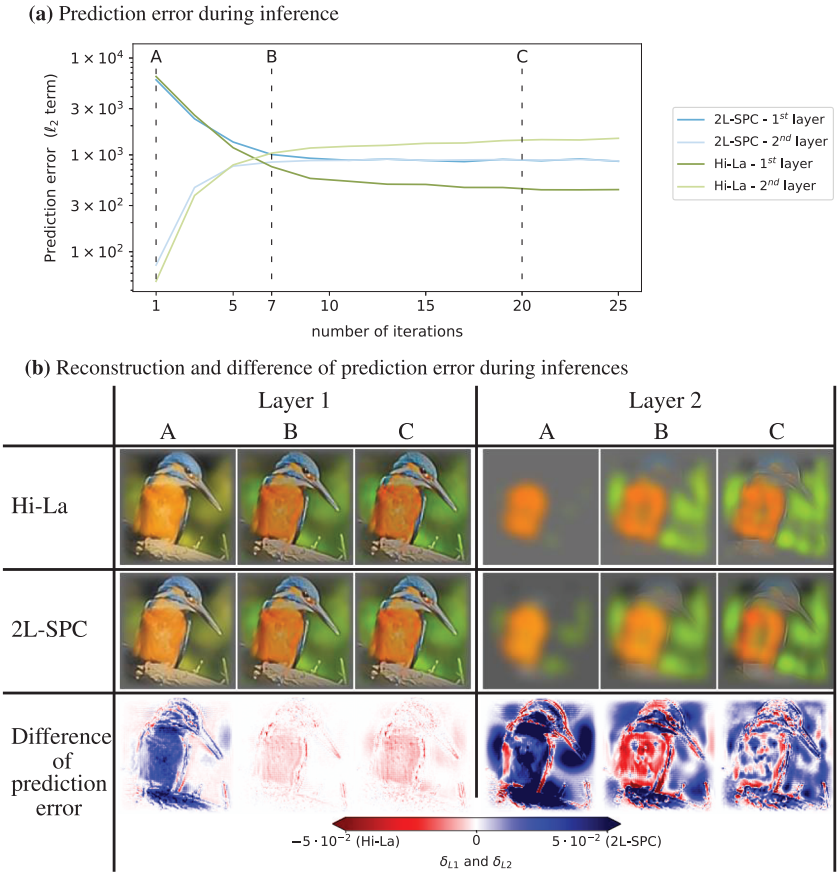

Figure 5: Evolution of the prediction error (ℓ_2-term) during the first inference iterations of the Hi-La and 2L-SPC when trained on the STL-10 database (a). The letters A, B, and C correspond to the states of latent variables at iterations 1, 7, and 20, respectively. We illustrate these states for both models and both layers by back-projecting the latent variables into the input space (first and middle rows of panel b). We also plot the difference of prediction error (see equation 4.1) in states A, B, and C (last row of panel b). Blue indicates that the 2L-SPC has a lower prediction error than the Hi-La, and inversely for red.

We now back-project all the latent variables into the input space to assess qualitatively how both models and layers are representing the input image (see Figure 12 for more details on the back-projection mechanism). These back-projections are shown in the two first lines of Figure 5b. We observe that the reconstructions of the first-layer latent variables are highly similar

for both the Hi-La and the 2L-SPC models. Note also that the quantitative difference between the first-layer prediction error of the Hi-La and the 2L-SPC at state C is not perceptible in the corresponding reconstructions. In state A, both models' first layers have already perceived all the details of the input. It suggests that feedforward networks are sufficient to provide accurate first-layer reconstruction (see section 2.5). In contrast, both models' second-layer reconstructions are very rough in state A and get refined along the inference process. In states B and C, we observe that the 2L-SPC second-layer reconstructions exhibit more details that the corresponding Hi-La reconstructions. In particular the 2L-SPC tends to better outline the contours of the object.

Next we compute the difference of prediction errors between the Hi-La and the 2L-SPC:

$$\begin{aligned}\delta_{L1} &= \left(\boldsymbol{x} - \mathbf{D}_{1_H}^T \boldsymbol{\gamma}_{1_H}\right)^2 - \left(\boldsymbol{x} - \mathbf{D}_{1_S}^T \boldsymbol{\gamma}_{1_S}\right)^2 \\ \delta_{L2} &= \left(\mathbf{D}_{1_H}^T (\boldsymbol{\gamma}_{1_H} - \mathbf{D}_{2_H}^T \boldsymbol{\gamma}_{2_H})\right)^2 - \left(\mathbf{D}_{1_S}^T (\boldsymbol{\gamma}_{1_S} - \mathbf{D}_{2_S}^T \boldsymbol{\gamma}_{2_S})\right)^2. \end{aligned} \tag{4.1}$$

In this equation, δ_{L1} denotes the difference of the first-layer prediction error between both models, and δ_{L2} is the difference of the back-projected prediction errors in input space. Variables indexed by the letters H and S are the Hi-La and the 2L-SPC variables, respectively. Figure 5b shows the first- and second-layer differences of prediction error for the different states A, B, and C. In these images, we choose a color scale such that blue points out the areas where the 2L-SPC has a lower prediction error than the Hi-La, and inversely for red. In state A, we observe that the 2L-SPC presents a lower prediction error than the Hi-La inside the represented object (a bird in the example in Figure 5b). Note that at the first inference step, both Hi-La and 2L-SPC have the same inference process (see section 2.5). Therefore, this difference in prediction error in state A is exclusively coming from the learning of their respective dictionaries. In particular, the 2L-SPC dictionaries tend to better represent finer textures (such as wood or the feathers of the bird example shown in Figure 5b). On the other hand, the first layer of the Hi-La is better at reconstructing the input when the inference process is in state B or C. This last observation is in line with the curves presented in Figure 5a. Concerning the second-layer prediction error, we observe that the Hi-La is better performing than the 2L-SPC when it comes to filling in the details of the object in state B. Nevertheless, the more the inference process is advanced, the more the 2L-SPC gets better at filling in the object (see state C, layer 2 in the third row of Figure 5b). Finally, when the inference process has converged, the 2L-SPC tends to better represent contrasted contours as the beak or the breast.

These observations suggest that the 2L-SPC is beneficial for the second layer to better represent both fine textural details and contrasted contours.

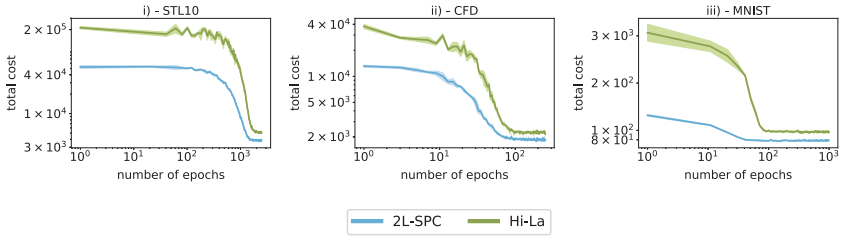

Figure 6: Evolution of the total cost during the training evaluated on the STL-10, CFD, and MNIST testing sets. Shaded areas correspond to mean absolute deviation on seven runs. All graphs have a logarithmic scale on both the x- and y-axis.

4.4 2L-SPC Learns Faster. Figure 6 shows the evolution of the total cost during the dictionary learning stage and evaluated on the testing set (see section 5 for the AT&T database). For all databases, the 2L-SPC model reaches its minimal total cost before the Hi-La model. The convergence rate of both models is comparable, but the 2L-SPC has a much lower cost in the very first epochs. The interlayer feedback connection of the 2L-SPC pushes the network toward lower prediction errors from the very beginning of the learning.

4.5 2L-SPC Features are Larger and More Generic. Another way to grasp the impact of the interlayer feedback connection is to visualize its effect on the dictionaries. To generate a human-readable visualization of the learned dictionaries, we back-project them into the image space using a cascade of transposed convolutions (see Figure 12). Using the analogy with neuroscience, these back-projections are called receptive fields (RFs). Figure 7 shows some of the RFs for the two layers and the second-layer activation probability histogram for both models when they are trained on the CFD database. In general, first-layer RFs are oriented Gabor-like filters, and second-layer RFs are more specific and represent more abstract concepts (e.g., curvatures, eyes, mouth, nose). In some extreme cases, RFs in the second layer of the Hi-La seem to overfit some specific faces and do not encompass all generality in the concept of a face. The red-framed RFs highlight one of these typical cases: the corresponding activation probabilities are 0.25% and 0.92% for Hi-La and 2L-SPC, respectively. This overfitting of features is supported by the lowest activation probability of the second layer's atoms of the Hi-La compared to the one of the 2L-SPC (0.16% versus 0.30%). This phenomenon is even more striking when we sort all the features by activation probabilities in descending order (see Figure 14). We filter out the highest activation probability (corresponding to the low-frequency filters highlighted by the black squares) of both Hi-La and 2L-SPC for scaling

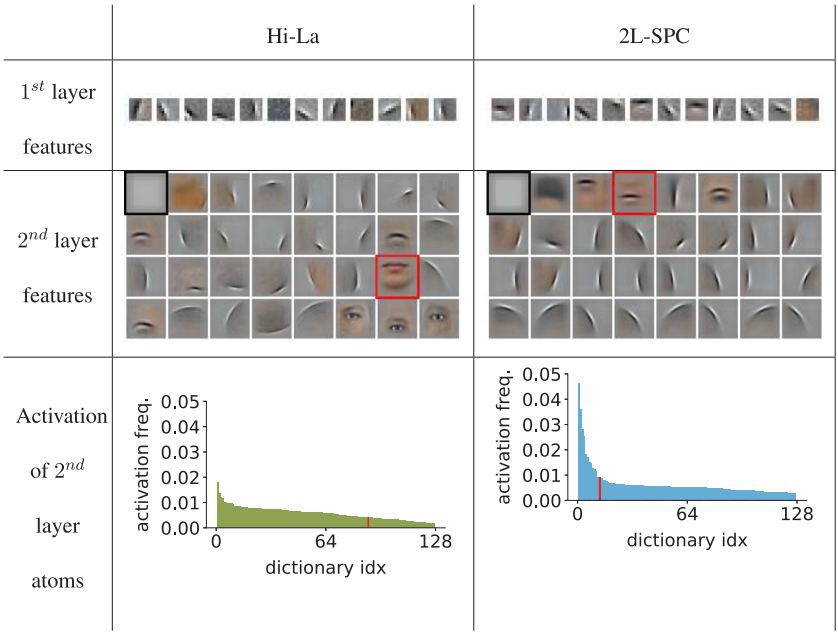

Figure 7: Hi-La and 2L-SPC RFs obtained on the CFD database (with $\lambda_1 = 0.3$, $\lambda_2 = 1.8$) and their associated second-layer activation probability histogram. The first- and second-layer RFs are 9×9 px and 33×33, px respectively. For the first-layer RFs, we randomly selected 12 of 64 atoms. For the second-layer RFs, we sampled 32 of 128 atoms and ranked them by their activation probability in descending order. For readability, we removed the most activated filter (RF framed in black) in 2L-SPC and the Hi-La second-layer activation histogram. The activation probabilities of the RFs framed in red are shown as a red bar in the corresponding histogram.

reasons. All the filters are displayed in Figures 13 through 16 for STL-10, CFD, MNIST, and AT&T RFs, respectively. The atoms' activation probability confirms the qualitative analysis of the RFs: the features learned by that the 2L-SPC learned are more generic and informative as they describe a wider range of images.

We also observe that the second-layer RFs present longer curvatures and oriented lines in the 2L-SPC than in the Hi-La model. We quantitatively confirm this statement by computing the mean surface coverage (MSC) of the first- and second-layer features (see Table 2). To perform such an analysis, we first filter out the low-frequency and face-specific features (e.g., eyes, nose) to keep only the oriented lines and the curvatures. Next, we normalize dictionaries for both models to make sure that the maximum pixel's intensity is equal to one. We next binarize each atom using the standard deviation

Table 2: Mean Surface Coverage for the First- and Second-Layer Features Obtained on Four Databases.

	Mean Surface Coverage					
	Layer 1			Layer 2		
Database	Hi-La	2L-SPC	Variation	Hi-La	2L-SPC	Variation
CFD	8.4%	9.4%	+12%	3.2%	4.8%	+50%
STL-10	7.9%	9.5%	+20%	2.5%	3.9%	+56%
MNIST	12.1%	14.6%	+21%	3.8%	5.8%	+52%
AT&T	11.1%	12.3%	+11%	3.0%	4.3%	+43%

as a threshold. Consequently, the only remaining pixels are those presenting a pixel's intensity higher than the standard deviation of the atom. Finally, we compute the MSC by summing the number of active pixels and dividing it by the total surface of the RF. For the four databases, both the first- and the second-layer dictionaries of the 2L-SPC are covering a larger space in the RFs (see Table 2). In particular, the first-layer dictionary of the 2L-SPC has an extra MSC varying from +11% to +21% compared to the one of the Hi-La. The increase of spatial extension is even more pronounced in the case of the second-layer features. The 2L-SPC second-layer atom have an extra MSC ranging from +43% to +56% compared to those of the Hi-La.

The analysis of the features reveals that the second-layer features of the 2L-SPC are more frequently activated than those of the Hi-La. The 2L-SPC second-layer atoms are then more generic as they encode for more diverse situations. In addition, these features are presenting a larger spatial extension and then include more contextual information compared to those of the Hi-La. Interestingly, these results might serve as an explanation for the lower global residual error observed in Figure 2.

5 Conclusion

What are the computational advantages of interlayer feedback connections in hierarchical sparse coding algorithms? We answered this question by comparing the hierarchical lasso (Hi-La) and the 2-Layer sparse predictive coding (2L-SPC) models. Both are identical in every respect, except that the 2L-SPC adds interlayer feedback connections. These extra connections force the internal state variables of the 2L-SPC to converge toward a trade-off between an accurate prediction passed by the lower layer and a better predictability by the upper layer. Experimentally, we demonstrated for these 2-layered networks on four different databases that the interlayer feedback connection (1) mitigates the overall prediction error by distributing it among layers, (2) accelerates the convergence toward a stable internal state, and (3) allows the second layer to refine its representation of the input,

(4) accelerates the learning process, and (5) enables the learned features to be more generic and spatially extended.

The 2L-SPC is novel in its way to consider hierarchical sparse coding (HSC) as a combination of local SC subproblems linked with the PC theory. This is a crucial difference with CNNs that are trained by backpropagating gradients from a global loss. To the best of our knowledge, the 2L-SPC model is the first one that leverages local sparse coding into a hierarchical and unsupervised algorithm. The ML-CSC from Sulam et al. (2018) is equivalent to a one-layer sparse coding algorithm (Aberdam et al., 2019), and the ML-ISTA from Sulam et al. (2019) is trained using supervised learning. However, deconvolutional networks (Zeiler & Fergus, 2012; Zeiler et al., 2011) are not leveraging local sparse coding as each layer aims at reconstructing the input image. Interestingly, other approaches based on hierarchical probabilistic inference have successfully generated sparse decomposition of images even though these models are not explicitly solving HSC problems (Lee, Grosse, Ranganath, & Ng, 2009).

Therefore, the 2L-SPC is a proof of concept demonstrating the beneficial impact of feedback connection in HSC models. Further work needs to be conducted to generalize our results to deeper networks. In particular, one needs to find an efficient normalization mechanism, compatible with recurrent networks, to mitigate the strong, vanishing activity phenomenon we have observed with deeper latent variables. Neuroscience might bring an elegant solution to this problem through divisive normalization. Another crucial extension of the work presented here would consist in including a channel-wise activation function (e.g., one sparsity parameter per atoms) in which the optimal biases would be assessed in the inference process. Such an improvement would allow the network to adapt the sparsity of the layers with the specificity of the input image. For example, highly textural images would result in a very high sparsity for oriented line features and a low sparsity for textural features.

If one succeeds in applying all of these improvement to such a sparse predictive coding framework, such networks should exhibit promising results to model the brain and tackle practical applications like image inpainting, denoising, and image super-resolution.

Appendix

A.1 2L-SPC Pseudo-Code.

Algorithm 2: 2L-SPC Inference Algorithm.

input : image: $\boldsymbol{x}$, dictionaries: $\{\boldsymbol{D}_i\}_{i=1}^L$, penalty param: $\{\lambda_i\}_{i=1}^L$, stability threshold: T_{stab}

$\boldsymbol{\gamma}_0^t = \boldsymbol{x}$

$\{\boldsymbol{\gamma}_i^0\}_{i=1}^L = \mathbf{0}, \quad \{\boldsymbol{\gamma}_{m_i}^1\}_{i=1}^L = \mathbf{0}$ # Initializing state variables

$\alpha^1 = 1$ # Initializing momentum strength

$\eta_{c_i} = \dfrac{1}{\max\left(eigen\_value(\boldsymbol{D}_i^T \boldsymbol{D}_i)\right)}$

$Stable = False$ # Initializing the stability criterion

$t = 0$

while $Stable == False$ **do**

  $t \mathrel{+}= 1$

  $\alpha^{t+1} = \dfrac{1 + \sqrt{1 + 4(\alpha^t)^2}}{2}$

  for $i = 1$ **to** L **do**

    $\boldsymbol{\epsilon}_{LL} = \boldsymbol{\gamma}_{m_{i-1}}^t - \boldsymbol{D}_i^T \boldsymbol{\gamma}_{m_i}^t$ # Update lower-layer error

    if $i \neq L$ **then**

      $\boldsymbol{\epsilon}_{UL} = \boldsymbol{\gamma}_{m_i}^t - \boldsymbol{D}_{i+1}^T \boldsymbol{\gamma}_{m_{i+1}}^t$ # Update the upper-layer error

    else

      $\boldsymbol{\epsilon}_{UL} = \mathbf{0}$

    $\boldsymbol{\gamma}_i^t = \mathcal{T}_{\eta_{c_i}\lambda_i}\left(\boldsymbol{\gamma}_{m_i}^t + \eta_{c_i} \boldsymbol{D}_i \boldsymbol{\epsilon}_{LL} - \eta_{c_i} \boldsymbol{\epsilon}_{UL}\right)$ # Update layer state variables

    $\boldsymbol{\gamma}_{m_i}^{t+1} = \mathcal{T}_0\left(\boldsymbol{\gamma}_i^t + \left(\dfrac{\alpha^t - 1}{\alpha^{t+1}}\right)\left(\boldsymbol{\gamma}_i^t - \boldsymbol{\gamma}_i^{t-1}\right)\right)$ # Update momentum

  if $\bigwedge_{i=1}^{L} \left(\dfrac{\|\boldsymbol{\gamma}_i^t - \boldsymbol{\gamma}_i^{t-1}\|_2}{\|\boldsymbol{\gamma}_i^t\|_2} < T_{stab}\right)$ **then**

    $Stable = True$ # Update the stability creterion

return $\{\boldsymbol{\gamma}_i^t\}_{i=1}^L$

Note: $\mathcal{T}_\alpha(\cdot)$ denotes the element-wise, non-negative soft-thresholding operator. A fortiori, $\mathcal{T}_0(\cdot)$ is a rectified linear unit operator. # comments are comments.

A.2 Evolution of the Global Prediction Error with Error Bar.

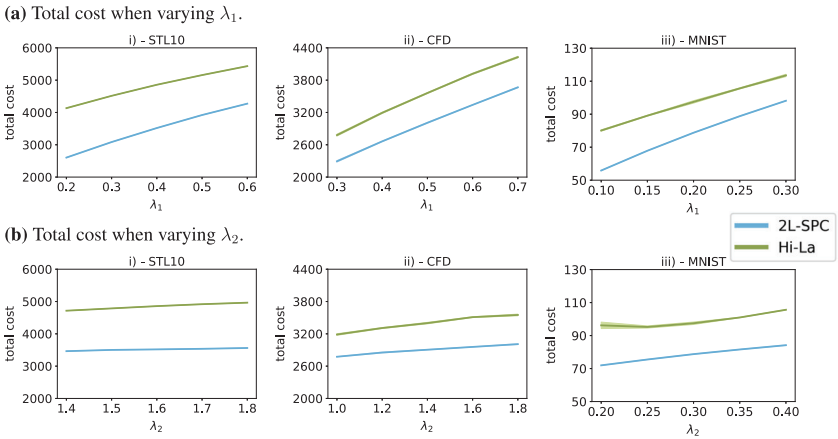

Figure 8: Evolution of the total cost evaluated on the testing set for both 2L-SPC and Hi-La networks. We vary the first-layer sparsity in the top row and the second-layer sparsity in the bottom row. Experiments have been conducted on STL-10, CFD, and MNIST databases. Shaded areas correspond to mean absolute deviation on seven runs. Sometimes the dispersion is so small that it looks as if there is no shade.

A.3 2L-SPC Parameters on ATT.

Table 3: Network Architecture, Training, and Simulation Parameters on the AT&T Database.

		ATT Database
Network parameters	$\boldsymbol{D}_1$ size	[64, 1, 9, 9] (3)
	$\boldsymbol{D}_2$ size	[128, 64, 9, 9] (1)
	T_{stab}	5e-4
Training parameters	Number of epochs	1000
	η_{L_1}	1e-4
	η_{L_2}	5e-3
Simulation parameters	λ_1 range	[0.3 : 0.7 :: 0.1, 1]
	λ_2 range	[0.5, 0.6 : 1.6 :: 0.2]

Notes: The sizes of the convolutional kernels are shown in the format: [# features, # channels, width, height] (stride). To describe the range of explored parameters during simulations, we use the format [0.3 : 0.7 :: 0.1, 0.5],which means that we vary λ_1 from 0.3 to 0.7 by steps of 0.1 while λ_2 is fixed to 0.5.

A.4 Prediction Error Distribution on AT&T.

(a) Distribution of the total lcost among layers when varying λ_1.

(b) Distribution of the total cost among layers when varying λ_2.

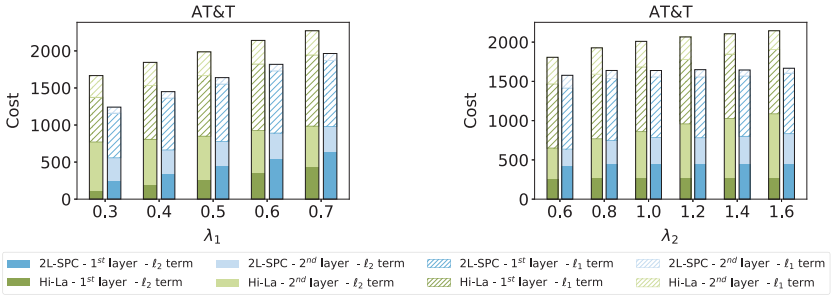

Figure 9: Evolution of the costs, evaluated on the testing set, for both 2L-SPC and Hi-La networks trained on the AT&T database. We vary the first-layer sparsity (a) and the second-layer sparsity (b). For each layer, the cost is decomposed into a quadratic cost (i.e., ℓ_2 term) represented with solid bars and a sparsity cost (i.e., ℓ_1 term) represented with hashed bars. First-layer costs are represented with darker colors and second-layer costs with lighter colors.

A.5 Number of Iterations of the Inference on AT&T.

(a) Number of iterations of the inference when varying λ_1.

(b) Number of iterations of the inference when varying λ_2.

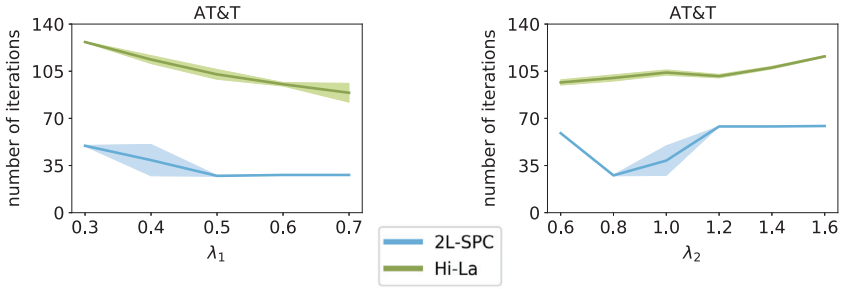

Figure 10: Evolution of the number of iterations needed to reach a stability criterion for both 2L-SPC and Hi-La networks on the AT&T testing set. We vary first-layer sparsity (a) and second-layer sparsity (b). Shaded areas correspond to mean absolute deviation on seven runs. Sometimes the dispersion is so small that it looks like there is no shade.

A.6 Evolution of Prediction Error during Training.

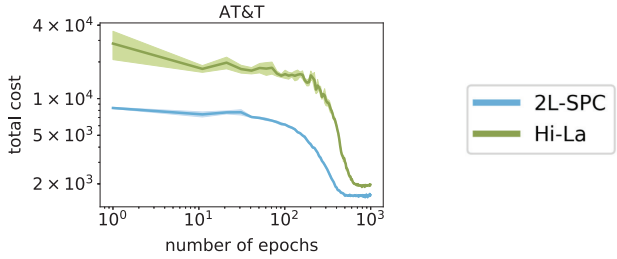

Figure 11: Evolution of the total cost during the training for the ATT testing set. Shaded areas correspond to mean absolute deviation on seven runs. The graph has a logarithmic scale in both the x- and y-axis.

A.7 Illustration of the Back-Projection Mechanism.

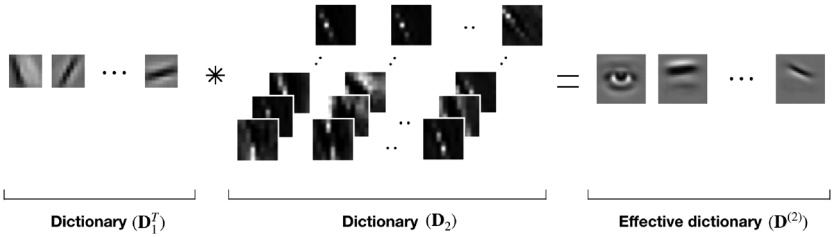

Figure 12: Generation of the second-layer effective dictionary. The result of this back-projection is called effective dictionary and could be assimilated to the notion of preferred stimulus in neuroscience. In a general case, the effective dictionary at layer i is computed as follow: $\mathbf{D}_i^{\text{eff},T} = \mathbf{D}_0^T..\mathbf{D}_{i-1}^T\mathbf{D}_i^T$ (Sulam et al., 2018).

A.8 Full Map of RFs for 2L-SPC and Hi-La on STL10.

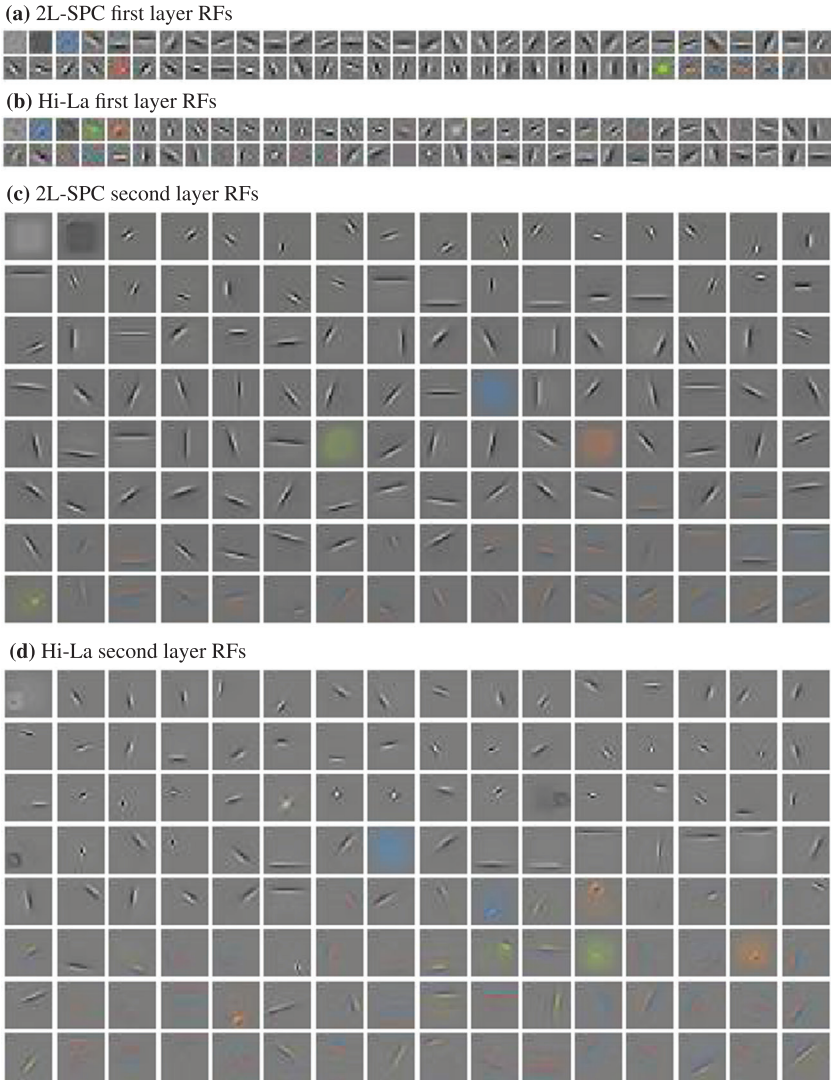

Figure 13: 2L-SPC (a,c) and Hi-La (b,d) effective dictionaries obtained on the STL-10 database, with sparsity parameter: ($\lambda_1 = 0.5$, $\lambda_2 = 1$). All other parameters are those described in Table 1. Atoms are sorted by activation probabilities in descending order. First-layer effective dictionaries have a size of 8×8 px (a,b) and second-layer RFs have a size of 22×22 (c,d) px, respectively.

A.9 Full Map of RFs for 2L-SPC and Hi-La on CFD.

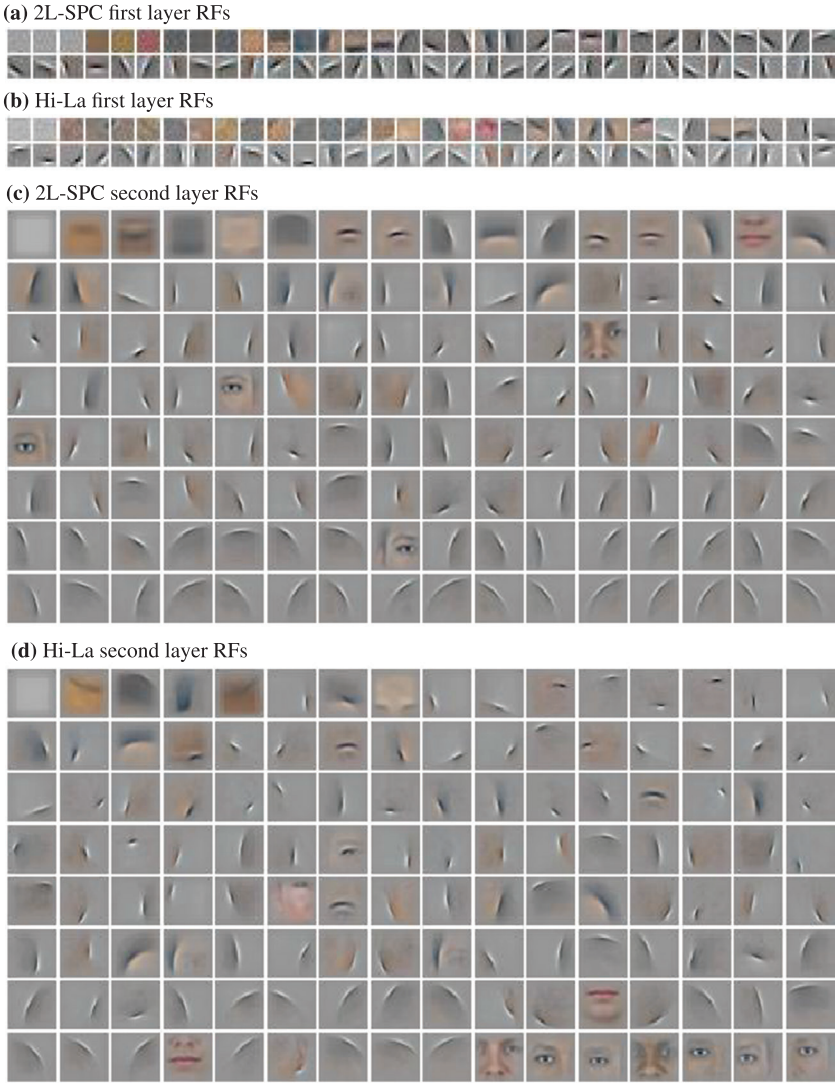

Figure 14: 2L-SPC (a,c) and Hi-La (b,d) effective dictionaries obtained on the CFD database, with sparsity parameter: ($\lambda_1 = 0.3$, $\lambda_2 = 1.8$). All other parameters are those described in Table 1. Atoms are sorted by activation probabilities in descending order. First-layer effective dictionaries have a size of 9×9 px (a,b) and second-layer RFs have a size of 33×33 (c,d) px, respectively.

A.10 Full Map of RFs for 2L-SPC and Hi-La on MNIST.

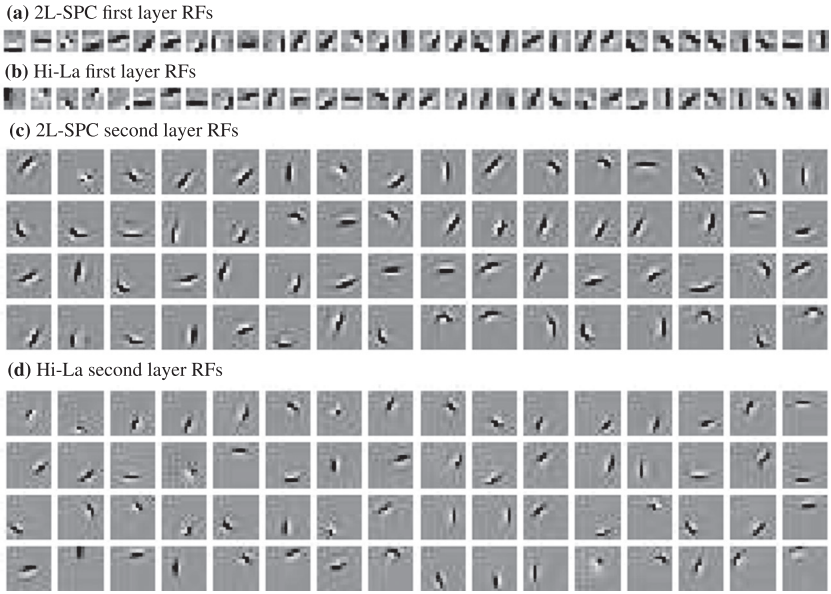

Figure 15: 2L-SPC (a,c) and Hi-La (b,d) effective dictionaries obtained on the MNIST database, with sparsity parameter: ($\lambda_1 = 0.2, \lambda_2 = 0.3$). Atoms are sorted by activation probabilities in descending order. All other parameters are those described in Table 1 for the MNIST database. The visualization shown here is the projection of the dictionaries into the input space. First-layer effective dictionaries have a size of 5×5 px (a,b) and second-layer RFs have a size of 14×14 (c,d) px, respectively.

A.11 Full Map of RFs for 2L-SPC and Hi-La on AT&T.

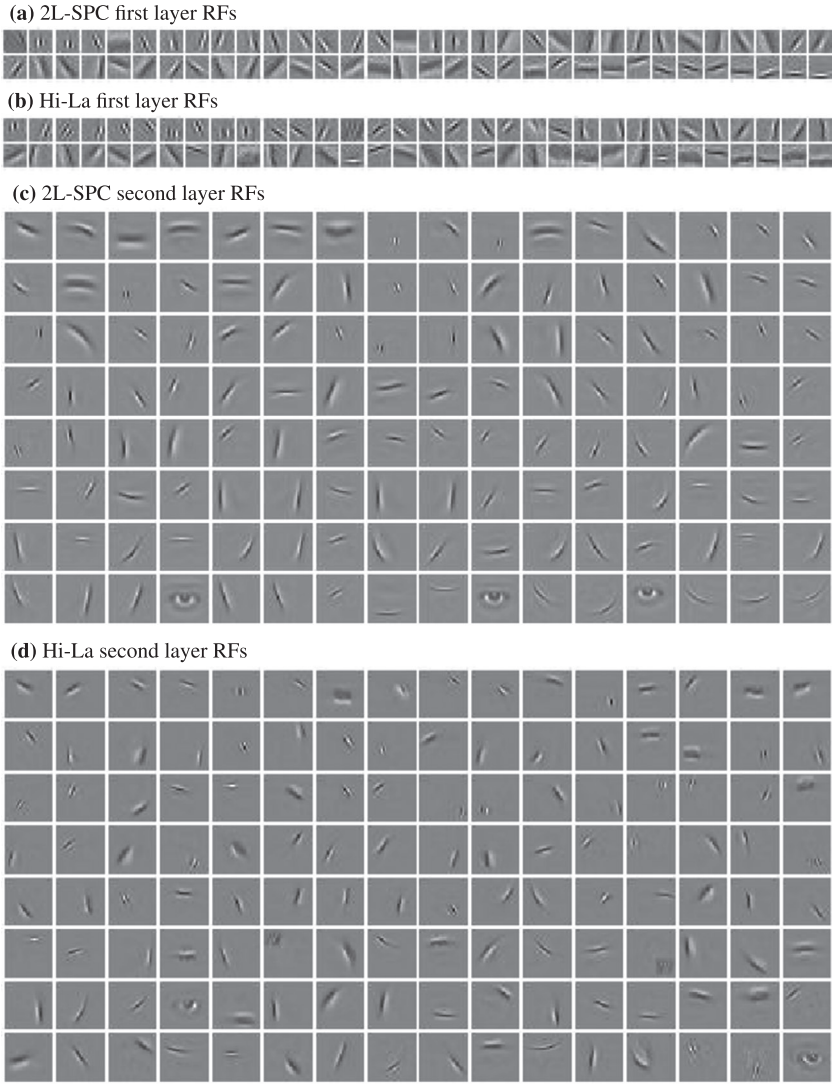

Figure 16: 2L-SPC (a,c) and Hi-La (b,d) effective dictionaries obtained on the AT&T database, with sparsity parameter: ($\lambda_1 = 0.5$, $\lambda_2 = 1$). All other parameters are those described in Table 3. Atoms are sorted by activation probabilities in descending order. First-layer effective dictionaries have a size of 9×9 px (a,b) and second-layer RFs have a size of 26×26 (c,d) px, respectively.

A.12 Preprocessed Samples.

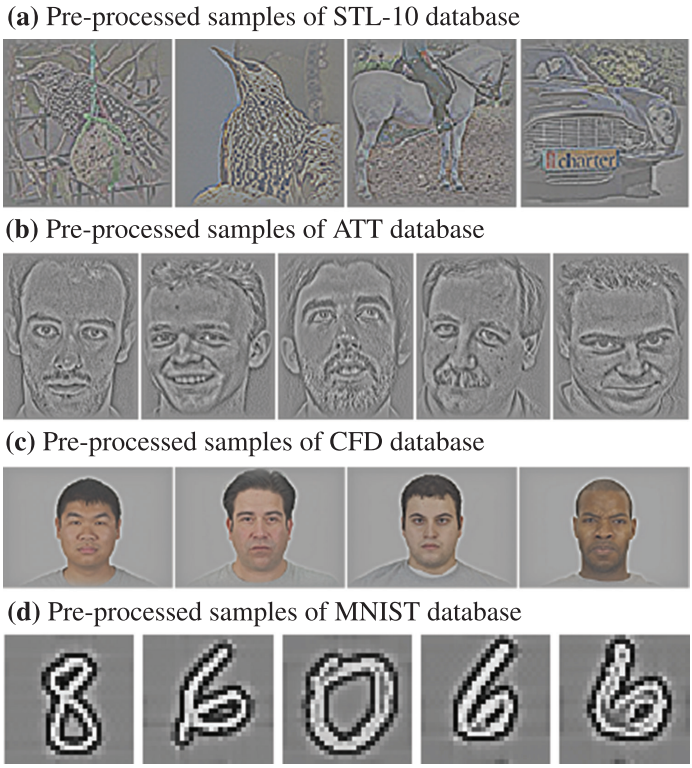

Figure 17: Preprocessed samples from STL-10 (a), AT&T (b), CFD (c), and MNIST (d) databases. All of these databases are preprocessed using local-contrast normalization (LCN) and whitening.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement 713750. It has also been carried out with financial support from the Regional Council of Provence-Alpes-Côte d'Azur and with the financial support of the A*MIDEX (ANR-11-IDEX-0001-02). This work was granted access to the HPC resources of Aix-Marseille Université financed by the project EquipMeso (ANR-10-EQPX-29-01) of the Investissements d'Avenir program.

References

Aberdam, A., Sulam, J., & Elad, M. (2019). Multi-layer sparse coding: The holistic way. *SIAM Journal on Mathematics of Data Science*, *1*(1), 46–77.

AT&T. (1994). *Face database.* Cambridge, UK: AT&T Laboratories.

Beck, A., & Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, *2*(1), 183–202.

Chalasani, R., Principe, J. C., & Ramakrishnan, N. (2013). A fast proximal method for convolutional sparse coding. In *Proceedings of the 2013 International Joint Conference on Neural Networks* (pp. 1–5). Piscataway, NJ: IEEE.

Coates, A., Ng, A., & Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (pp. 215–223).

Elad, M. (2010). *Sparse and redundant representations: From theory to applications in signal and image processing*. New York: Springer.

Friston, K. (2010). The free-energy principle: A unified brain theory? *Nature Reviews Neuroscience*, *11*(2), 127.

Gregor, K., & LeCun, Y. (2010). Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning* (pp. 399–406). ICML.

Han, K., Wen, H., Zhang, Y., Fu, D., Culurciello, E., & Liu, Z. (2018). Deep predictive coding network with local recurrent processing for object recognition. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems*, *31* (pp. 9221–9233). Red Hook, NY: Curran.

Heide, F., Heidrich, W., & Wetzstein, G. (2015). Fast and flexible convolutional sparse coding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5135–5143). Piscataway, NJ: IEEE.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, *160*(1), 106–154.

Jarrett, K., Kavukcuoglu, K., & LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *Proceedings of the 2009 IEEE 12th International Conference on Computer Vision*, (pp. 2146–2153). Piscataway, NJ: IEEE.

Kreutz-Delgado, K., Murray, J. F., Rao, B. D., Engan, K., Lee, T.-W., & Sejnowski, T. J. (2003). Dictionary learning algorithms for sparse representation. *Neural Computation*, *15*(2), 349–396.

LeCun, Y. (1998). *The MNIST database of handwritten digits.* http://yann.lecun.com/exdb/mnist/

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 609–616).

Li, Y., & Osher, S. (2009). Coordinate descent optimization for l1 minimization with application to compressed sensing: A greedy algorithm. *Inverse Problems and Imaging*, *3*(3), 487–503.

Lotter, W., Kreiman, G., & Cox, D. (2016). *Deep predictive coding networks for video prediction and unsupervised learning.* arXiv:1605.08104.

Ma, D. S., Correll, J., & Wittenbrink, B. (2015). The Chicago face database: A free stimulus set of faces and norming data. *Behavior Research Methods*, *47*(4), 1122–1135.

Mairal, J., Bach, F., Ponce, J., & Sapiro, G. (2009). Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 689–696).

Mairal, J., Bach, F., Ponce, J., Sapiro, G., & Zisserman, A. (2009). Non-local sparse models for image restoration. In *Proceedings of the 2009 IEEE 12th International Conference on Computer Vision* (pp. 2272–2279). Piscataway, NJ: IEEE.

Makhzani, A., & Frey, B. (2013). *K-sparse autoencoders.* arXiv:1312.5663.

Makhzani, A., & Frey, B. J. (2015). Winner-take-all autoencoders. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems, 28* (pp. 2791–2799). Red Hook, NY: Curran.

Mallat, S., & Zhang, Z. (1993). *Matching pursuit with time-frequency dictionaries* (Technical Report). New York: Courant Institute of Mathematical Sciences.

Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, *37*(23), 3311–3325.

Papyan, V., Romano, Y., & Elad, M. (2017). Convolutional neural networks analyzed via convolutional sparse coding. *Journal of Machine Learning Research*, *18*(1), 2887–2938.

Perrinet, L. U., & Bednar, J. A. (2015). Edge co-occurrences can account for rapid categorization of natural versus animal images. *Scientific Reports*, *5*, 11400. http://www.nature.com/articles/srep11400. doi: 10.1038/srep11400

Pham-Gia, T., & Hung, T. (2001). The mean and median absolute deviations. *Mathematical and Computer Modelling*, *34*(7–8), 921–936.

Rao, R. P., & 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.

Rubinstein, R., Bruckstein, A. M., & Elad, M. (2010). Dictionaries for sparse representation modeling. In *Proceedings of the IEEE*, *98*(6), 1045–1057.

Spratling, M. W. (2017). A hierarchical predictive coding model of object recognition in natural images. *Cognitive Computation*, *9*(2), 151–167.

Sulam, J., Aberdam, A., Beck, A., & Elad, M. (2019). On multi-layer basis pursuit, efficient algorithms and convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*, 1968–1980.

Sulam, J., Papyan, V., Romano, Y., & Elad, M. (2018). Multilayer convolutional sparse modeling: Pursuit and dictionary learning. *IEEE Transactions on Signal Processing*, *66*(15), 4090–4104.

Szlam, A., Kavukcuoglu, K., & LeCun, Y. (2010). *Convolutional matching pursuit and dictionary training.* arXiv:1010.0422.

Wen, H., Han, K., Shi, J., Zhang, Y., Culurciello, E., & Liu, Z. (2018). *Deep predictive coding network for object recognition*. arXiv:1802.04762.

Yang, M., Zhang, L., Yang, J., & Zhang, D. (2011). Robust sparse coding for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 625–632). Piscataway, NJ: IEEE.

Zeiler, M. D., & Fergus, R. (2012). *Differentiable pooling for hierarchical feature learning.* arXiv:1207.0151.

Zeiler, M. D., Taylor, G. W., & Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *Proceedings of the 2011 International Conference on Computer Vision* (pp. 2018–2025). Washington, DC: IEEE Computer Society.

Received January 24, 2020; accepted July 6, 2020.