Multilayer Convolutional Sparse Modeling: Pursuit and Dictionary Learning

Jeremias Sulam, *Member, IEEE*, Vardan Papyan, Yaniv Romano, and Michael Elad, *Fellow, IEEE*

***Abstract*—The recently proposed multilayer convolutional sparse coding (ML-CSC) model, consisting of a cascade of convolutional sparse layers, provides a new interpretation of convolutional neural networks (CNNs). Under this framework, the forward pass in a CNN is equivalent to a pursuit algorithm aiming to estimate the nested sparse representation vectors from a given input signal. Despite having served as a pivotal connection between CNNs and sparse modeling, a deeper understanding of the ML-CSC is still lacking. In this paper, we propose a sound pursuit algorithm for the ML-CSC model by adopting a projection approach. We provide new and improved bounds on the stability of the solution of such pursuit and we analyze different practical alternatives to implement this in practice. We show that the training of the filters is essential to allow for nontrivial signals in the model, and we derive an online algorithm to learn the dictionaries from real data, effectively resulting in cascaded sparse convolutional layers. Last, but not least, we demonstrate the applicability of the ML-CSC model for several applications in an unsupervised setting, providing competitive results. Our work represents a bridge between matrix factorization, sparse dictionary learning, and sparse autoencoders, and we analyze these connections in detail.**

***Index Terms*—Convolutional sparse coding, multilayer pursuit, convolutional neural networks, dictionary learning, sparse convolutional filters.**

I. Introduction

NEW ways of understanding real world signals, and proposing ways to model their intrinsic properties, have led to improvements in signal and image restoration, detection and classification, among other problems. Little over a decade ago, sparse representation modeling brought about the idea that natural signals can be (well) described as a linear combination of only a few building blocks or components, commonly known as atoms [1]. Backed by elegant theoretical results, this model led to a series of works dealing either with the problem of the pursuit of such decompositions, or with the design and learning of better atoms from real data [2]. The latter problem, termed dictionary learning, empowered sparse enforcing methods to achieve remarkable results in many different fields from signal and image processing [3]–[5] to machine learning [6]–[8].

Neural networks, on the other hand, were introduced around forty years ago and were shown to provide powerful classification algorithms through a series of function compositions [9], [10]. It was not until the last half-decade, however, that through a series of incremental modifications these methods were boosted to become the state-of-the-art machine learning tools for a wide range of problems, and across many different fields [11]. For the most part, the development of new variants of deep convolutional neural networks (CNNs) has been driven by trial-and-error strategies and a considerable amount of intuition.

Withal, a few research groups have begun providing theoretical justifications and analysis strategies for CNNs from very different perspectives. For instance, by employing wavelet filters instead of adaptive ones, the work by Bruna and Mallat [12] demonstrated how *scattering networks* represent shift invariant analysis operators that are robust to deformations (in a Lipschitz-continuous sense). The inspiring work of [13] proposed a generative Bayesian model, under which typical deep learning architectures perform an inference process. In [14], the authors proposed a hierarchical tensor factorization analysis model to analyze deep CNNs. Fascinating connections between sparse modeling and CNN have also been proposed. In [15], a neural network architecture was shown to be able to learn iterative shrinkage operators, essentially *unrolling* the iterations of a sparse pursuit. Building on this interpretation, the work in [16] further showed that CNNs can in fact improve the performance of sparse recovery algorithms.

A precise connection between sparse modeling and CNNs was recently presented in [17], and its contribution is centered in defining the Multi-Layer Convolutional Sparse Coding (ML-CSC) model. When deploying this model to real signals, compromises were made in way that each layer is only *approximately* explained by the following one. With this relaxation in the pursuit of the convolutional representations, the main observation of that work is that the inference stage of CNNs – nothing but the forward-pass – can be interpreted as a very crude pursuit algorithm seeking for unique sparse representations. This is a useful perspective as it provides a precise optimization objective which, it turns out, CNNs attempt to minimize.

Manuscript received January 10, 2018; revised April 24, 2018 and May 30, 2018; accepted May 31, 2018. Date of publication June 11, 2018; date of current version June 25, 2018. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Olivier Lezoray. This work was supported by the European Research Council under European Unions Seventh Framework Programme ERC Grant Agreement 320649. *(Corresponding author: Jeremias Sulam.)*

J. Sulam and M. Elad are with the Department of Computer Science, Technion-Israel Institute of Technology, Haifa 3200003, Israel (e-mail: jsulam@cs.technion.ac.il; elad@cs.technion.ac.il).

V. Papyan and Y. Romano are with the Department of Statistics, Stanford University, Stanford, CA 94305 USA (e-mail: vardanp91@gmail.com; yromano@tx.technion.ac.il).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The material includes complete proofs of some of the theoretical results, as well as further comments. The file is 357 KB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2018.2846226

The work in [17] further proposed improved pursuits for approximating the sparse representations of the network, or feature maps, such as the Layered Basis Pursuit algorithm. Nonetheless, as we will show later, neither this nor the forward pass serve the ML-CSC model exactly, as they do not provide signals that comply with the model assumptions. In addition, the theoretical guarantees accompanying these layered approaches suffer from bounds that become looser with the network's depth. The lack of a suitable pursuit, in turn, obscures how to properly sample from the ML-CSC model, and how to train the dictionaries from real data.

In this work we undertake a fresh study of the ML-CSC and of pursuit algorithms for signals in this model. Our contributions will be guided by addressing the following questions:

1) Given proper convolutional dictionaries, how can one project[1] signals onto the ML-CSC model?
2) When will the model allow for *any* signal to be expressed in terms of nested sparse representations? In other words, is the model empty?
3) What conditions should the convolutional dictionaries satisfy? and how can we adapt or learn them to represent real-world signals?
4) How is the learning of the ML-CSC model related to traditional CNN and dictionary learning algorithms?
5) What kind of performance can be expected from this model?

The model we analyze in this work is related to several recent contributions, both in the realm of sparse representations and deep-learning. On the one hand, the ML-CSC model is tightly connected to dictionary constrained learning techniques, such as Chasing Butterflies approach [18], fast transform learning [19], Trainlets [20], among several others. On the other hand, and because of the unsupervised flavor of the learning algorithm, our work shares connections to sparse auto-encoders [21], and in particular to the k-sparse [22] and winner-take-all versions [23].

In order to progressively answer the questions posed above, we will first review the ML-CSC model in detail in Section II. We will then study how signals can be projected onto the model in Section III, where we will analyze the stability of the projection problem and provide theoretical guarantees for practical algorithms. We will then propose a learning formulation in Section IV-B, which will allow, for the first time, to obtain a trained ML-CSC model from real data while being perfectly faithful to the model assumptions. In this work we restrict our study to the learning of the model in an unsupervised setting. This approach will be further demonstrated on signal approximation and unsupervised learning applications in Section V, before concluding in Section VI.

II. Background

A. Convolutional Sparse Coding

The Convolutional Sparse Coding (CSC) model assumes a signal $\mathbf{x} \in \mathbb{R}^N$ admits a decomposition as $\mathbf{D}_1\boldsymbol{\gamma}_1$, where

[1]By projection, we refer to the task of getting the closest signal to the one given that obeys the model assumptions.

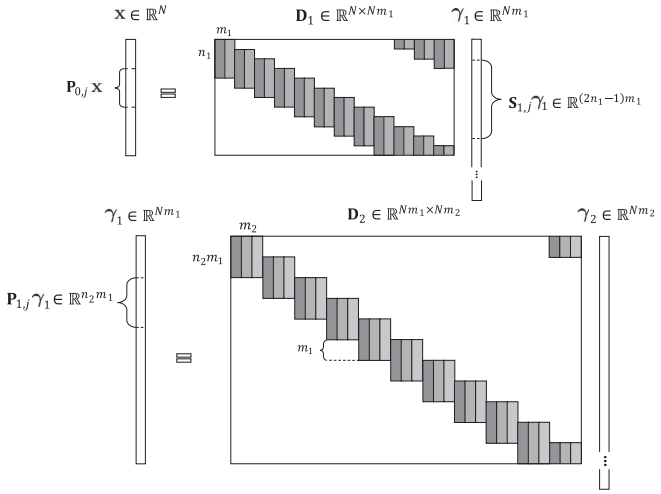


Fig. 1. The CSC model (top), and its ML-CSC extension by imposing a similar model on $\boldsymbol{\gamma}_1$ (bottom).

$\boldsymbol{\gamma}_1 \in \mathbb{R}^{Nm_1}$ is sparse and $\mathbf{D}_1 \in \mathbb{R}^{N \times Nm_1}$ has a convolutional structure. More precisely, this dictionary consists of m_1 local n_1-dimensional filters at every possible location (Fig. 1 top). An immediate consequence of this model assumption is the fact that each jth *patch* $\mathbf{P}_{0,j}\mathbf{x} \in \mathbb{R}^{n_1}$ from the signal $\mathbf{x}$ can be expressed in terms of a shift-invariant local model corresponding to a *stripe* from the global sparse vector, $\mathbf{S}_{1,j}\boldsymbol{\gamma}_1 \in \mathbb{R}^{(2n_1-1)m_1}$. From now on, and for the sake of simplicity, we will drop the first index on the stripe and patch extraction operators, simply denoting the jth stripe from $\boldsymbol{\gamma}_1$ as $\mathbf{S}_j\boldsymbol{\gamma}_1$.

In the context of CSC, the sparsity of the representation is better captured through the $\ell_{0,\infty}$ pseudo-norm [24]. This measure, as opposed to the traditional ℓ_0, provides a notion of local sparsity and it is defined by the maximal number of non-zeros in a stripe from $\boldsymbol{\gamma}$. Formally,

$$\|\boldsymbol{\gamma}\|_{0,\infty}^{s} = \max_i \|\mathbf{S}_i\boldsymbol{\gamma}\|_0. \tag{1}$$

We kindly refer the reader to [24] for a more detailed description of this model, as well as extensive theoretical guarantees associated with the model stability and the success of pursuit algorithms serving it.

This model presents several characteristics that make it relevant and interesting. On the one hand, CSC provides a systematic and formal way to develop and analyze very popular and successful patch-based algorithms in signal and image processing [24]. From a more practical perspective, on the other hand, the convolutional sparse model has recently received considerable attention in the computer vision and machine learning communities. Solutions based on the CSC have been proposed for detection [25], compressed sensing [26] texture-cartoon separation [27], inverse problems [28]–[30] and feature learning [31], [32], and different convolutional dictionary learning algorithms have been proposed and analyzed [28], [33], [34]. Interestingly, this model has also been employed in a hierarchical way [35]–[38] mostly following intuition and imitating successful CNNs' architectures. This connection between convolutional features and multi-layer constructions was recently made

precise in the form of the Multi-Layer CSC model, which we review next.

B. Multi Layer CSC

The Multi-Layer Convolutional Sparse Coding (ML-CSC) model is a natural extension of the CSC described above, as it assumes that a signal can be expressed by sparse representations at different layers in terms of nested convolutional filters. Suppose $\mathbf{x} = \mathbf{D}_1\boldsymbol{\gamma}_1$, for a convolutional dictionary $\mathbf{D}_1 \in \mathbb{R}^{N\times Nm_1}$ and an $\ell_{0,\infty}$-sparse representation $\boldsymbol{\gamma}_1 \in \mathbb{R}^{Nm_1}$. One can cascade this model by imposing a similar assumption on the representation $\boldsymbol{\gamma}_1$, i.e., $\boldsymbol{\gamma}_1 = \mathbf{D}_2\boldsymbol{\gamma}_2$, for a corresponding convolutional dictionary $\mathbf{D}_2 \in \mathbb{R}^{Nm_1\times Nm_2}$ with m_2 local filters and a $\ell_{0,\infty}$-sparse $\boldsymbol{\gamma}_2$, as depicted in Fig. 1. In this case, $\mathbf{D}_2$ is a also a convolutional dictionary with local filters skipping m_1 entries at a time[2] – as there are m_1 *channels* in the representation $\boldsymbol{\gamma}_1$.

Because of this multi-layer structure, vector $\boldsymbol{\gamma}_1$ can be viewed both as a sparse representation (in the context of $\mathbf{x} = \mathbf{D}_1\boldsymbol{\gamma}_1$) or as a signal (in the context of $\boldsymbol{\gamma}_1 = \mathbf{D}_2\boldsymbol{\gamma}_2$). Thus, one one can refer to both its stripes (looking backwards to patches from $\mathbf{x}$) or its patches (looking forward, corresponding to stripes of $\boldsymbol{\gamma}_2$). In this way, when analyzing the ML-CSC model we will not only employ the $\ell_{0,\infty}$ norm as defined above, but we will also leverage its *patch* counterpart, where the maximum is taken over all patches from the sparse vector by means of a patch extractor operator $\mathbf{P}_i$. In order to make their difference explicit, we will denote them as $\|\boldsymbol{\gamma}\|_{0,\infty}^s$ and $\|\boldsymbol{\gamma}\|_{0,\infty}^p$ for stripes and patches, respectively. In addition, we will employ the $\ell_{2,\infty}$ norm version, naturally defined as $\|\boldsymbol{\gamma}\|_{2,\infty}^s = \max_i \|\mathbf{S}_i\boldsymbol{\gamma}\|_2$, and analogously for patches.

We now formalize the model definition:

Definition 1: ML-CSC model:
Given a set of convolutional dictionaries $\{\mathbf{D}_i\}_{i=1}^L$ of appropriate dimensions, a signal $\mathbf{x}(\boldsymbol{\gamma}_i) \in \mathbb{R}^N$ admits a representation in terms of the ML-CSC model, i.e. $\mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{M}_{\boldsymbol{\lambda}}$, if

$$\begin{aligned}
\mathbf{x} &= \mathbf{D}_1\boldsymbol{\gamma}_1, \quad \|\boldsymbol{\gamma}_1\|_{0,\infty}^s \le \lambda_1,\\
\boldsymbol{\gamma}_1 &= \mathbf{D}_2\boldsymbol{\gamma}_2, \quad \|\boldsymbol{\gamma}_2\|_{0,\infty}^s \le \lambda_2,\\
&\vdots\\
\boldsymbol{\gamma}_{L-1} &= \mathbf{D}_L\boldsymbol{\gamma}_L, \quad \|\boldsymbol{\gamma}_L\|_{0,\infty}^s \le \lambda_L.
\end{aligned}$$

Note that $\mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{M}_{\boldsymbol{\lambda}}$ can also be expressed as $\mathbf{x} = \mathbf{D}_1\mathbf{D}_2 \ldots \mathbf{D}_L\boldsymbol{\gamma}_L$. We refer to $\mathbf{D}^{(i)}$ as the *effective* dictionary at the ith level, i.e., $\mathbf{D}^{(i)} = \mathbf{D}_1\mathbf{D}_2 \ldots \mathbf{D}_i$. This way, one can concisely write

$$\mathbf{x} = \mathbf{D}^{(i)}\boldsymbol{\gamma}_i,\ 1 \le i \le L. \tag{2}$$

Interestingly, the ML-CSC can be interpreted as a special case of a CSC model: one that enforces a very specific structure on the intermediate representations. We make this statement precise in the following Lemma:

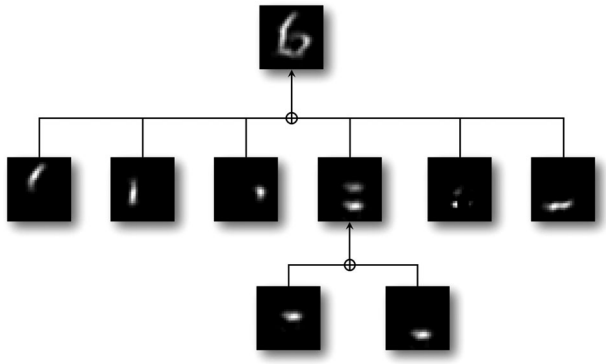


Fig. 2. From atoms to molecules: Illustration of the ML-CSC model for a number 6. Two local convolutional atoms (bottom row) are combined to create slightly more complex structures – molecules – at the second level, which are then combined to create the global atom representing, in this case, a digit.

Lemma 1: Given the ML-CSC model described by the set of convolutional dictionaries $\{\mathbf{D}_i\}_{i=1}^L$, with filters of spatial dimensions n_i and channels m_i, any dictionary $\mathbf{D}^{(i)} = \mathbf{D}_1\mathbf{D}_2 \ldots \mathbf{D}_i$ is a convolutional dictionary with m_i local atoms of dimension $n_i^{\text{eff}} = \sum_{j=1}^i n_j - (i-1)$. In other words, the ML-CSC model is a structured global convolutional model.
The proof of this lemma is rather straight forward, and we include it in the Supplementary Material I. Note that what was denoted as the effective dimension at the ith layer is nothing else than what is known in the deep learning community as the *receptive field* of a filter at layer i. Here, we have made this concept precise in the context of the ML-CSC model.

As it was presented, the convolutional model assumes that every n-dimensional atom is located at every possible location, which implies that the filter is shifted with strides of $s = 1$. An alternative, which effectively reduces the redundancy of the resulting dictionary, is to consider a stride greater than one. In such case, the resulting dictionary is of size $N \times Nm_1/s$ for one dimensional signals, and $N \times Nm_1/s^2$ for images. This construction, popular in the CNN community, does not alter the effective size of the filters but rather decreases the length of each stripe by a factor of s in each dimension. In the limit, when $s = n_1$, one effectively considers non-overlapping blocks and the stripe will be of length[3] m_1 – the number of local filters. Naturally, one can also employ $s > 1$ for any of the multiple layers of the ML-CSC model. We will consider $s = 1$ for all layers in our derivations for simplicity.

The ML-CSC imposes a unique structure on the global dictionary $\mathbf{D}^{(L)}$, as it provides a multi-layer linear composition of simpler structures. In other words, $\mathbf{D}_1$ contains (small) local n_1-dimensional atoms. The product $\mathbf{D}_1\mathbf{D}_2$ contains in each of its columns a linear combination of atoms from $\mathbf{D}_1$, merging them to create molecules. Further layers continue to create more complex constructions out of the simpler convolutional building blocks. We depict an example of such decomposition in Fig. 2 for a 3rd-layer convolutional atom of the digit "6". While the

[2]This construction provides operators that are convolutional in the space domain, but not in the channel domain – just as for CNNs.

[3]When $s = n_1$, the system is no longer shift-invariant, but rather invariant with a shift of n samples.

question of how to obtain such dictionaries will be addressed later on, let us make this illustration concrete: consider this atom to be given by $\mathbf{x}_0 = \mathbf{D}_1\mathbf{D}_2\mathbf{d}_3$, where $\mathbf{d}_3$ is sparse, producing the upper-most image $\mathbf{x}_0$. Denoting by $\mathcal{T}(\mathbf{d}_3) = Supp(\mathbf{d}_3)$, this atom can be equally expressed as

$$\mathbf{x}_0 = \mathbf{D}^{(2)}\mathbf{d}_3 = \sum_{j\in\mathcal{T}(\mathbf{d}_3)} \mathbf{d}_j^{(2)} d_3^j. \tag{3}$$

In words, the effective atom is composed of *a few* elements from the effective dictionary $\mathbf{D}^{(2)}$. These are the building blocks depicted in the middle of Fig. 2. Likewise, focusing on the fourth of such atoms, $\mathbf{d}_{j_4}^{(2)} = \mathbf{D}_1\mathbf{d}_{2,j_4}$. In this particular case, $\|\mathbf{d}_{2,j_4}\|_0 = 2$, so we can express $\mathbf{d}_{j_4}^{(2)} = \mathbf{d}_{i_1}^{(1)} d_{2,j_1}^{i_1} + \mathbf{d}_{i_2}^{(1)} d_{2,j_1}^{i_2}$. These two atoms from $\mathbf{D}_1$ are precisely those appearing in the bottom of the decomposition.

C. Pursuit in the Noisy Setting

Real signals might contain noise or deviations from the above idealistic model assumption, preventing us from enforcing the above model exactly. Consider the scenario of acquiring a signal $\mathbf{y} = \mathbf{x} + \mathbf{v}$, where $\mathbf{x} \in \mathcal{M}_{\boldsymbol{\lambda}}$ and $\mathbf{v}$ is a nuisance vector of bounded energy, $\|\mathbf{v}\|_2 \leq \mathcal{E}_0$. In this setting, the objective is to estimate all the representations $\boldsymbol{\gamma}_i$ which explain the measurements $\mathbf{y}$ up to an error of $\mathcal{E}_0$. In its most general form, this pursuit is represented by the Deep Coding Problem (DCP$_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$), as introduced in [17]:

Definition 2: DCP$_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ Problem:
For a global signal $\mathbf{y}$, a set of convolutional dictionaries $\{\mathbf{D}_i\}_{i=1}^L$, and vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\mathcal{E}}$:

$$\begin{aligned}
(\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}): \quad & \text{find} \quad \{\boldsymbol{\gamma}_i\}_{i=1}^L \quad \text{s.t.} \\
& \|\mathbf{y} - \mathbf{D}_1\boldsymbol{\gamma}_1\|_2 \leq \mathcal{E}_0, && \|\boldsymbol{\gamma}_1\|_{0,\infty}^s \leq \lambda_1 \\
& \|\boldsymbol{\gamma}_1 - \mathbf{D}_2\boldsymbol{\gamma}_2\|_2 \leq \mathcal{E}_1, && \|\boldsymbol{\gamma}_2\|_{0,\infty}^s \leq \lambda_2 \\
& \qquad\vdots && \qquad\vdots \\
& \|\boldsymbol{\gamma}_{L-1} - \mathbf{D}_L\boldsymbol{\gamma}_L\|_2 \leq \mathcal{E}_{L-1}, && \|\boldsymbol{\gamma}_L\|_{0,\infty}^s \leq \lambda_L
\end{aligned}$$

where λ_i and $\mathcal{E}_i$ are the ith entries of $\boldsymbol{\lambda}$ and $\boldsymbol{\mathcal{E}}$, respectively.

The solution to this problem was shown to be stable in terms of a bound on the ℓ_2-distance between the estimated representations $\hat{\boldsymbol{\gamma}}_i$ and the true ones, $\boldsymbol{\gamma}_i$. These results depend on the characterization of the dictionaries through their mutual coherence, $\mu(\mathbf{D})$, which measures the maximal normalized correlation between atoms in the dictionary. Formally, assuming the atoms are normalized as $\|\mathbf{d}_i\|_2 = 1\ \forall i$, this measure is defined as

$$\mu(\mathbf{D}) = \max_{i\neq j} |\mathbf{d}_i^T\mathbf{d}_j|. \tag{4}$$

Relying on this measure, Theorem 5 in [17] shows that given a signal $\mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ contaminated with noise of known energy $\mathcal{E}_0^2$, if the representations satisfy the sparsity constraint

$$\|\boldsymbol{\gamma}_i\|_{0,\infty}^s < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{D}_i)}\right), \tag{5}$$

then the solution to the DCP$_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ given by $\{\hat{\boldsymbol{\gamma}}_i\}_{i=1}^L$ satisfies

$$\|\boldsymbol{\gamma}_i - \hat{\boldsymbol{\gamma}}_i\|_2^2 \leq 4{\mathcal{E}_0}^2 \prod_{j=1}^{i} \frac{4^{i-1}}{1 - (2\|\boldsymbol{\gamma}_j\|_{0,\infty}^s - 1)\mu(\mathbf{D}_j)}. \tag{6}$$

In the particular instance of the DCP$_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ where $\mathcal{E}_i = 0$ for $1 \leq i \leq L-1$, the above bound can be made tighter by a factor of 4^{i-1} while preserving the same form.

These results are encouraging, as they show for the first time stability guarantees for a problem for which the forward pass provides an approximate solution. More precisely, if the above model deviations are considered to be greater than zero ($\mathcal{E}_i > 0$) several layer-wise algorithms, including the forward pass of CNNs, provide approximations to the solution of this problem [17]. We note two remarks about these stability results:

1) The bound increases with the number of layers or the depth of the network. This is a direct consequence of the layer-wise relaxation in the above pursuit, which causes these discrepancies to accumulate over the layers.
2) Given the underlying signal $\mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{M}_{\boldsymbol{\lambda}}$, with representations $\{\boldsymbol{\gamma}_i\}_{i=1}^L$, this problem searches for their corresponding estimates $\{\hat{\boldsymbol{\gamma}}_i\}_{i=1}^L$. However, because at each layer $\|\hat{\boldsymbol{\gamma}}_{i-1} - \mathbf{D}_i\hat{\boldsymbol{\gamma}}_i\|_2 > 0$, this problem *does not* provide representations for a signal in the model. In other words, $\hat{\mathbf{x}} \neq \mathbf{D}_1\hat{\boldsymbol{\gamma}}_1$, $\hat{\boldsymbol{\gamma}}_1 \neq \mathbf{D}_2\hat{\boldsymbol{\gamma}}_2$, and generally $\hat{\mathbf{x}} \notin \mathcal{M}_{\boldsymbol{\lambda}}$.

III. A Projection Alternative

In this section we provide an alternative approach to the problem of estimating the underlying representations $\boldsymbol{\gamma}_i$ under the same noisy scenario of $\mathbf{y} = \mathbf{x}(\boldsymbol{\gamma}_i) + \mathbf{v}$. In particular, we are interested in projecting the measurements $\mathbf{y}$ onto the set $\mathcal{M}_{\boldsymbol{\lambda}}$. Consider the following projection problem:

Definition 3: ML-CSC Projection $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$:
For a signal $\mathbf{y}$ and a set of convolutional dictionaries $\{\mathbf{D}_i\}_{i=1}^L$, define the Multi-Layer Convolutional Sparse Coding projection as:

$$(\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}): \quad \min_{\{\boldsymbol{\gamma}_i\}_{i=1}^L} \|\mathbf{y} - \mathbf{x}(\boldsymbol{\gamma}_i)\|_2 \quad \text{s.t.} \quad \mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{M}_{\boldsymbol{\lambda}}. \tag{7}$$

Note that this problem differs from the DCP$_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ counterpart in that we seek for a signal close to $\mathbf{y}$, whose representations $\boldsymbol{\gamma}_i$ give rise to $\mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{M}_{\boldsymbol{\lambda}}$. This is more demanding (less general) than the formulation in the DCP$_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$. Put differently, the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem can be considered as a special case of the DCP$_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ where model deviations are allowed only at the outer-most level. Recall that the theoretical analysis of the DCP$_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ problem indicated that the error thresholds should increase with the layers. Here, the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem suggests a completely different approach.

A. Stability of the Projection $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$

Given $\mathbf{y} = \mathbf{x}(\boldsymbol{\gamma}_i) + \mathbf{v}$, one can seek for the underlying representations $\boldsymbol{\gamma}_i$ through either the DCP$_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ or $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem. In light of the above discussion and the known stability result for the DCP$_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ problem, how close will the solution of the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem be from the true set of representations? The answer is provided through the following result.

Theorem 4: Stability of the solution to the $\mathcal{P}_{\mathcal{M}_\lambda}$ problem:
Suppose $\mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{M}_\lambda$ is observed through $\mathbf{y} = \mathbf{x} + \mathbf{v}$, where $\mathbf{v}$ is a bounded noise vector, $\|\mathbf{v}\|_2 \leq \mathcal{E}_0$, and $\|\boldsymbol{\gamma}_i\|_{0,\infty}^s = \lambda_i < \frac{1}{2}(1 + \frac{1}{\mu(\mathbf{D}^{(i)})})$, for $1 \leq i \leq L$. Consider the set $\{\hat{\boldsymbol{\gamma}}_i\}_{i=1}^L$ to be the solution of the $\mathcal{P}_{\mathcal{M}_\lambda}$ problem. Then,

$$\|\boldsymbol{\gamma}_i - \hat{\boldsymbol{\gamma}}_i\|_2^2 \leq \frac{4\mathcal{E}_0^2}{1 - (2\|\boldsymbol{\gamma}_i\|_{0,\infty}^s - 1)\mu(\mathbf{D}^{(i)})}. \tag{8}$$

For the sake of brevity, we include the proof of this claim in the Supplementary Material II. However, we note a few remarks:

1) The obtained bounds are not cumulative across the layers. In other words, they do not grow with the depth of the network.
2) Unlike the stability result for the DCP$_\lambda^\mathcal{E}$ problem, the assumptions on the sparse vectors $\boldsymbol{\gamma}_i$ are given in terms of the mutual coherence of the effective dictionaries $\mathbf{D}^{(i)}$. Interestingly enough, we will see in the experimental section that one can in fact have that $\mu(\mathbf{D}^{(i-1)}) > \mu(\mathbf{D}^{(i)})$ in practice; i.e., the effective dictionary becomes incoherent as it becomes deeper. Indeed, the deeper layers provide larger atoms with correlations that are expected to be lower than the inner products between two small local (and overlapping) filters.
3) While the conditions imposed on the sparse vectors $\boldsymbol{\gamma}_i$ might seem prohibitive, one should remember that this follows from a worst case analysis. Moreover, one can effectively construct analytic nested convolutional dictionaries with small coherence measures, as shown in [17].

Interestingly, one can also formulate bounds for the stability of the solution, i.e. $\|\boldsymbol{\gamma}_i - \hat{\boldsymbol{\gamma}}_i\|_2^2$, which are the tightest for the inner-most layer, and then increase as one moves to shallower layers – precisely the opposite behavior of the solution to the DCP$_\lambda^\mathcal{E}$ problem. This result, however, provides bounds that are generally looser than the one presented in the above theorem, and so we defer this to the Supplementary Material.

B. Pursuit Algorithms

We now focus on the question of how one can solve the above problems in pracice. As shown in [17], one can approximate the solution to the DCP$_\lambda^\mathcal{E}$ in a layer-wise manner, solving for the sparse representations $\hat{\boldsymbol{\gamma}}_i$ progressively from $i = 1, \ldots, L$. Surprisingly, the Forward Pass of a CNN *is* one such algorithm, yielding stable estimates. The Layered BP algorithm was also proposed, where each representation $\hat{\boldsymbol{\gamma}}_i$ is sparse coded (in a Basis Pursuit formulation) given the previous representation $\hat{\boldsymbol{\gamma}}_{i-1}$ and dictionary $\mathbf{D}_i$. As solutions to the DCP$_\lambda^\mathcal{E}$ problem, these algorithms inherit the layer-wise relaxation referred above, which causes the theoretical bounds to increase as a function of the layers or network depth.

Moving to the variation proposed in this work, how can one solve the $\mathcal{P}_{\mathcal{M}_\lambda}$ problem in practice? Applying the above layer-wise pursuit is clearly not an option, since after obtaining a necessarily distorted estimate $\hat{\boldsymbol{\gamma}}_1$ we cannot proceed with equalities for the next layers, as $\boldsymbol{\gamma}_1$ does not necessarily have a perfectly sparse representation with respect to $\mathbf{D}_2$. Herein we present a simple approach based on a global sparse coding solver which yields provable stable solutions.

Algorithm 1: ML-CSC Pursuit.

Input: $\mathbf{y}, \{\mathbf{D}_i\}, k$;

$\hat{\boldsymbol{\gamma}}_L \leftarrow \text{Pursuit}(\mathbf{y}, \mathbf{D}^{(L)}, k)$;

for $j = L, \ldots, 1$ **do**
  $\hat{\boldsymbol{\gamma}}_{j-1} \leftarrow \mathbf{D}_j \hat{\boldsymbol{\gamma}}_j$

return $\{\hat{\boldsymbol{\gamma}}_i\}$;

Consider Algorithm 1. This approach circumvents the problem of sparse coding the intermediate features while guaranteeing their exact expression in terms of the following layer. This is done by first running a Pursuit for the deepest representation through an algorithm which provides an approximate solution to the following problem:

$$\min_{\boldsymbol{\gamma}} \|\mathbf{y} - \mathbf{D}^{(L)}\boldsymbol{\gamma}\|_2^2 \;\text{ s.t. } \|\boldsymbol{\gamma}\|_{0,\infty}^s \leq k. \tag{9}$$

Once the deepest representation has been estimated, we proceed by obtaining the remaining ones by simply applying their definition, thus assuring that $\hat{\mathbf{x}} = \mathbf{D}^{(i)}\hat{\boldsymbol{\gamma}}_i \in \mathcal{M}_\lambda$. While this might seem like a dull strategy, we will see in the next section that, if the measurements $\mathbf{y}$ are close enough to a signal in the model, Algorithm 1 indeed provides stable estimates $\hat{\boldsymbol{\gamma}}_i$. In fact, the resulting stability bounds will be shown to be generally tighter than those existing for the layer-wise pursuit alternative. Moreover, as we will later see in the Results section, this approach can effectively be harnessed in practice in a real-data scenario.

C. Stability Guarantees for Pursuit Algorithms

Given a signal $\mathbf{y} = \mathbf{x}(\boldsymbol{\gamma}_i) + \mathbf{v}$, and the respective solution of the ML-CSC Pursuit in Algorithm 1, how close will the estimated $\hat{\boldsymbol{\gamma}}_i$ be to the original representations $\boldsymbol{\gamma}_i$? These bounds will clearly depend on the specific Pursuit algorithm employed to obtain $\hat{\boldsymbol{\gamma}}_L$. In what follows, we will present two stability guarantees that arise from solving this sparse coding problem under two different strategies: a greedy and a convex relaxation approach. Before diving in, however, we present two elements that will become necessary for our derivations.

The first one is a property that relates to the propagation of the support, or non-zeros, across the layers. Given the support of a sparse vector $\mathcal{T} = Supp(\boldsymbol{\gamma})$, consider dictionary $\mathbf{D}_\mathcal{T}$ as the matrix containing only the columns indicated by $\mathcal{T}$. Define $\|\mathbf{D}_\mathcal{T}\|_\infty^0 = \sum_{i=1}^n \|\mathcal{R}_i\mathbf{D}_\mathcal{T}\|_\infty^0$, where $\mathcal{R}_i$ extracts the ith *row* of the matrix on its right-hand side. In words, $\|\mathbf{D}_\mathcal{T}\|_\infty^0$ simply counts the number of non-zero rows of $\mathbf{D}_\mathcal{T}$. With it, we now define the following property:

Definition 5: Non Vanishing Support (N.V.S.):
A sparse vector $\boldsymbol{\gamma}$ with support $\mathcal{T}$ satisfies the N.V.S property for a given dictionary $\mathbf{D}$ if

$$\|\mathbf{D}\boldsymbol{\gamma}\|_0 = \|\mathbf{D}_\mathcal{T}\|_\infty^0. \tag{10}$$

Intuitively, the above property implies that the entries in $\boldsymbol{\gamma}$ will not cause two or more atoms to be combined in such a way that (any entry of) their supports cancel each other. Notice that this is a very natural assumption to make. Alternatively, one could assume the non-zero entries from $\boldsymbol{\gamma}$ to be Gaussian distributed, and in this case the N.V.S. property holds *a.s.*

A direct consequence of the above property is that of maximal cardinality of representations. If $\boldsymbol{\gamma}$ satisfies the N.V.S property for a dictionary $\mathbf{D}$, and $\bar{\boldsymbol{\gamma}}$ is another sparse vector with equal support (i.e., $Supp(\boldsymbol{\gamma}) = Supp(\bar{\boldsymbol{\gamma}})$), then necessarily $Supp(\mathbf{D}\bar{\boldsymbol{\gamma}}) \subseteq Supp(\mathbf{D}\boldsymbol{\gamma})$, and thus $\|\mathbf{D}\boldsymbol{\gamma}\|_0 \geq \|\mathbf{D}\bar{\boldsymbol{\gamma}}\|_0$. This follows from the fact that the number of non-zeros in $\mathbf{D}\bar{\boldsymbol{\gamma}}$ cannot be greater than the sum of non-zero rows from the set of atoms, $\mathbf{D}_{\mathcal{T}}$.

The second element concerns the local stability of the Stripe-RIP, the convolutional version of the Restricted Isometric Property [39]. As defined in [24], a convolutional dictionary $\mathbf{D}$ satisfies the Stripe-RIP condition with constant δ_k if, for every $\boldsymbol{\gamma}$ such that $\|\boldsymbol{\gamma}\|_{0,\infty}^{s} = k$,

$$(1-\delta_k)\|\boldsymbol{\gamma}\|_2^2 \leq \|\mathbf{D}\boldsymbol{\gamma}\|_2^2 \leq (1+\delta_k)\|\boldsymbol{\gamma}\|_2^2. \tag{11}$$

The S-RIP bounds the maximal change in (global) energy of a $\ell_{0,\infty}$-sparse vector when multiplied by a convolutional dictionary. We would like to establish an equivalent property but in a local sense. Recall the $\|\mathbf{x}\|_{2,\infty}^{p}$ norm, given by the maximal norm of a *patch* from $\mathbf{x}$, i.e. $\|\mathbf{x}\|_{2,\infty}^{p} = \max_i \|\mathbf{P}_i\mathbf{x}\|_2$. Analogously, one can consider $\|\boldsymbol{\gamma}\|_{2,\infty}^{s} = \max_i \|\mathbf{S}_i\boldsymbol{\gamma}\|_2$ to be the maximal norm of a *stripe* from $\boldsymbol{\gamma}$.

Now, is $\|\mathbf{D}\boldsymbol{\gamma}\|_{2,\infty}^{p}$ nearly isometric? The (partially affirmative) answer is given in the form of the following Lemma, which we prove in the Supplementary Material III.

Lemma 2: Local one-sided near isometry property:

If $\mathbf{D}$ is a convolutional dictionary satisfying the Stripe-RIP condition in (11) with constant δ_k, then

$$\|\mathbf{D}\boldsymbol{\gamma}\|_{2,\infty}^{2,p} \leq (1+\delta_k)\,\|\boldsymbol{\gamma}\|_{2,\infty}^{2,s}. \tag{12}$$

This result is worthy in its own right, as it shows for the first time that not only the CSC model is globally stable for $\ell_{0,\infty}$-sparse signals, but that one can also bound the change in energy in a local sense (in terms of the $\ell_{2,\infty}$ norm). While the above Lemma only refers to the upper bound of $\|\mathbf{D}\boldsymbol{\gamma}\|_{2,\infty}^{2,p}$, we conjecture that an analogous lower bound can be shown to hold as well.

With these elements, we can now move to the stability of the solutions provided by Algorithm 1:

Theorem 6: Stable recovery of the Multi-Layer Pursuit Algorithm in the convex relaxation case:

Suppose a signal $\mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{M}_{\boldsymbol{\lambda}}$ is contaminated with locally-bounded noise $\mathbf{v}$, resulting in $\mathbf{y} = \mathbf{x} + \mathbf{v}$, $\|\mathbf{v}\|_{2,\infty}^{p} \leq \epsilon_0$. Assume that all representations $\boldsymbol{\gamma}_i$ satisfy the N.V.S. property for the respective dictionaries $\mathbf{D}_i$, and that $\|\boldsymbol{\gamma}_i\|_{0,\infty}^{s} = \lambda_i < \frac{1}{2}(1 + \frac{1}{\mu(\mathbf{D}_i)})$, for $1 \leq i \leq L$ and $\|\boldsymbol{\gamma}_L\|_{0,\infty}^{s} = \lambda_L \leq \frac{1}{3}(1 + \frac{1}{\mu(\mathbf{D}^{(L)})})$. Let

$$\hat{\boldsymbol{\gamma}}_L = \arg\min_{\boldsymbol{\gamma}} \|\mathbf{y} - \mathbf{D}^{(L)}\boldsymbol{\gamma}\|_2^2 + \zeta_L\|\boldsymbol{\gamma}\|_1, \tag{13}$$

for $\zeta_L = 4\epsilon_0$, and set $\hat{\boldsymbol{\gamma}}_{i-1} = \mathbf{D}_i\hat{\boldsymbol{\gamma}}_i$, $i = L, \ldots, 1$. Then,

1) $Supp(\hat{\boldsymbol{\gamma}}_i) \subseteq Supp(\boldsymbol{\gamma}_i)$,
2) $\|\hat{\boldsymbol{\gamma}}_i - \boldsymbol{\gamma}_i\|_{2,\infty}^{p} \leq \epsilon_L \prod_{j=i+1}^{L} \sqrt{\frac{3c_j}{2}}$,

hold for every layer $1 \leq i \leq L$, where $\epsilon_L = \frac{15}{2}\,\epsilon_0\sqrt{\|\boldsymbol{\gamma}_L\|_{0,\infty}^{p}}$ is the error at the last layer, and c_j depends on the ratio between the local dimensions of the layers, $c_j = \lceil \frac{2n_{j-1}-1}{n_j} \rceil$.

Theorem 7: Stable recovery of the Multi-Layer Pursuit Algorithm in the greedy case:

Suppose a signal $\mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{M}_{\boldsymbol{\lambda}}$ is contaminated with energy-bounded noise $\mathbf{v}$, such that $\mathbf{y} = \mathbf{x} + \mathbf{v}$, $\|\mathbf{y} - \mathbf{x}\|_2 \leq \mathcal{E}_0$, and $\epsilon_0 = \|\mathbf{v}\|_{2,\infty}^{\mathrm{p}}$. Assume that all representations $\boldsymbol{\gamma}_i$ satisfy the N.V.S. property for the respective dictionaries $\mathbf{D}_i$, with $\|\boldsymbol{\gamma}_i\|_{0,\infty}^{s} = \lambda_i < \frac{1}{2}(1 + \frac{1}{\mu(\mathbf{D}_i)})$, for $1 \leq i \leq L$, and

$$\|\boldsymbol{\gamma}_L\|_{0,\infty}^{s} < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{D}^{(L)})}\right) - \frac{1}{\mu(\mathbf{D}^{(L)})} \cdot \frac{\epsilon_0}{|\gamma_L^{\min}|}, \tag{14}$$

where $\gamma_L^{\min}$ is the minimal entry in the support of $\boldsymbol{\gamma}_L$. Consider approximating the solution to the Pursuit step in Algorithm 1 by running Orthogonal Matching Pursuit for $\|\boldsymbol{\gamma}_L\|_0$ iterations. Then, for every ith layer,

1) $Supp(\hat{\boldsymbol{\gamma}}_i) \subseteq Supp(\boldsymbol{\gamma}_i)$,
2) $\|\hat{\boldsymbol{\gamma}}_i - \boldsymbol{\gamma}_i\|_2^2 \leq \frac{\mathcal{E}_0^2}{1-\mu(\mathbf{D}^{(L)})(\|\boldsymbol{\gamma}_L\|_{0,\infty}^{s}-1)} \left(\frac{3}{2}\right)^{L-i}$.

The proofs of both Theorems 6 and 7 are included in the Supplementary Material IV-A and IV-B, respectively. The coefficient c_j refers to the ratio between the filter dimensions at consecutive layers, and assuming $n_i \approx n_{i+1}$ (which indeed happens in practice), this coefficient is roughly 2. Importantly, and unlike the bounds provided for the layer-wise pursuit algorithm, the recovery guarantees are the tightest for the inner-most layer, and the bound increases slightly towards shallower representations. The relaxation to the ℓ_1 norm, in the case of the BP formulation, provides local error bounds, while the guarantees for the greedy version, in its OMP implementation, yield a global alternative.

Before proceeding, one might wonder if the above conditions imposed on the representations and dictionaries are too severe and whether the set of signals satisfying these is empty. This is, in fact, not the case. As shown in [17], multi-layer convolutional dictionaries can be constructed by means of certain wavelet functions, effectively achieving mutual coherence values in the order of 10^{-3}, leaving ample room for sampling sparse representations satisfying the theorems' assumptions. On the other hand, imposing a constraint on the number of non-zeros in a representation $\boldsymbol{\gamma}_{i-1} = \mathbf{D}_i\boldsymbol{\gamma}_i$ implies that part of the support of the atoms in $\mathbf{D}_i$ will be required to overlap. The N.V.S. property simply guarantees that whenever these overlaps occur, they will not cancel each other. Indeed, this happens with probability 1 if the non-zero coefficients are drawn from a Normal distribution. We further comment and exemplify this in the Supplementary Material IV-C.

D. Projecting General Signals

In the most general case, i.e. removing the assumption that $\mathbf{y}$ is close enough to a signal in the model, Algorithm 1 by itself might not solve $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$. Consider we are given a general signal $\mathbf{y}$

Algorithm 2: ML-CSC Projection Algorithm.

Init: $\mathbf{x}^* = \mathbf{0}$;
for $k = 1 : \lambda_L$ **do**
  $\hat{\boldsymbol{\gamma}}_L \leftarrow \text{OMP}(\mathbf{y}, \mathbf{D}^{(L)}, k)$;
  for $j = L : -1 : 1$ **do**
    $\hat{\boldsymbol{\gamma}}_{j-1} \leftarrow \mathbf{D}_j \hat{\boldsymbol{\gamma}}_j$;
  if $\|\hat{\boldsymbol{\gamma}}_i\|_{0,\infty}^s > \lambda_i$ *for any* $1 \le i < L$ **then**
    break;
  else
    $\mathbf{x}^* \leftarrow \mathbf{D}^{(i)} \hat{\boldsymbol{\gamma}}_i$;
return $\mathbf{x}^*$

and a model $\mathcal{M}_{\boldsymbol{\lambda}}$, and we run the ML-CSC Pursuit with $k = \lambda_L$ obtaining a set of representations $\{\hat{\boldsymbol{\gamma}}_j\}$. Clearly $\|\hat{\boldsymbol{\gamma}}_L\|_{0,\infty}^s \le \lambda_L$. Yet, nothing guarantees that $\|\hat{\boldsymbol{\gamma}}_i\|_{0,\infty}^s \le \lambda_i$ for $i < L$. In other words, in order to solve $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ one must guarantee that all sparsity constraints are satisfied.

Algorithm 2 progressively recovers sparse representations to provide a projection for any general signal $\mathbf{y}$. The solution is initialized with the zero vector, and then the OMP algorithm is applied with a progressively larger $\ell_{0,\infty}$ constraint on the deepest representation[4], from 1 to λ_L. The only modification required to run the OMP in this setting is to check at every iteration the value of $\|\hat{\boldsymbol{\gamma}}_L\|_{0,\infty}^s$, and to stop accordingly. At each step, given the estimated $\hat{\boldsymbol{\gamma}}_L$, the intermediate features and their $\ell_{0,\infty}$ norms, are computed. If all sparsity constraints are satisfied, then the algorithm proceeds. If, on the other hand, any of the constraints is violated, the previously computed $\mathbf{x}^*$ is reported as the solution. Note that this algorithm can be improved: if a constraint is violated, one might consider back-tracking the obtained deepest estimate and replacing the last obtained non-zero by an alternative solution, which might allow for the intermediate constraints to be satisfied. For simplicity, we present the completely greedy approach as in Algorithm 2.

This algorithm can be shown to be a greedy approximation to an optimal algorithm, under certain assumptions, and we provide a sketch of the proof of this claim in the Supplementary Material IV-D. Clearly, while Algorithms 1 and 2 were presented separately, they are indeed related and one can certainly combine them into a single method. The distinction between them was motivated by making the derivations of our theoretical analysis and guarantees easier to grasp. Nevertheless, stating further theoretical claims without the assumption of the signal $\mathbf{y}$ being close to an underlying $\mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{M}_\lambda$ is non-trivial, and we defer a further analysis of this case for future work.

E. Summary - Pursuit for the ML-CSC

Let us briefly summarize what we have introduced so far. We have defined a projection problem, $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$, seeking for the closest signal in the model $\mathcal{M}_{\boldsymbol{\lambda}}$ to the measurements $\mathbf{y}$. We have shown that if the measurements $\mathbf{y}$ are close enough to a signal in the model, i.e. $\mathbf{y} = \mathbf{x}(\boldsymbol{\gamma}_i) + \mathbf{v}$, with bounded noise $\mathbf{v}$, then the ML-CSC Pursuit in Algorithm 1 manages to obtain approximate solutions that are not far from these representations, by deploying either the OMP or the BP algorithms. In particular, the support of the estimated sparse vectors is guaranteed to be a subset of the correct support, and so all $\hat{\boldsymbol{\gamma}}_i$ satisfy the model constraints. In doing so we have introduced the N.V.S. property, and we have proven that the CSC and ML-CSC models are locally stable. Lastly, if no prior information is known about the signal $\mathbf{y}$, we have proposed an OMP-inspired algorithm that finds the closest signal $\mathbf{x}(\boldsymbol{\gamma}_i)$ to any measurements $\mathbf{y}$ by gradually increasing the support of all representations $\hat{\boldsymbol{\gamma}}_i$ while guaranteeing that the model constraints are satisfied.

IV. Learning the Model

The entire analysis presented so far relies on the assumption of the existence of proper dictionaries $\mathbf{D}_i$ allowing for corresponding *nested sparse features* $\boldsymbol{\gamma}_i$. Clearly, the ability to obtain such representations greatly depends on the design and properties of these dictionaries.

While in the traditional sparse modeling scenario certain analytically-defined dictionaries (such as the Discrete Cosine Transform) often perform well in practice, in the ML-CSC case it is hard to propose an off-the-shelf construction which would allow for any meaningful decompositions. To see this more clearly, consider obtaining $\hat{\boldsymbol{\gamma}}_L$ with Algorithm 1 removing all other assumptions on the dictionaries $\mathbf{D}_i$. In this case, nothing will prevent $\hat{\boldsymbol{\gamma}}_{L-1} = \mathbf{D}_L \hat{\boldsymbol{\gamma}}_L$ from being dense. More generally, we have no guarantees that *any* collection of dictionaries would allow for any signal with nested sparse components $\boldsymbol{\gamma}_i$. In other words, how do we know if the model represented by $\{\mathbf{D}_i\}_{i=1}^L$ is not empty?

To illustrate this important point, consider the case where $\mathbf{D}_i$ are random – a popular construction in other sparsity-related applications. In this case, every atom from the dictionary $\mathbf{D}_L$ will be a random variable $\mathbf{d}_L^j \sim \mathcal{N}(\mathbf{0}, \sigma_L^2 \mathbf{I})$. In this case, one can indeed construct $\boldsymbol{\gamma}_L$, with $\|\boldsymbol{\gamma}_L\|_{0,\infty}^s \le 2$, such that *every entry* from $\boldsymbol{\gamma}_{L-1} = \mathbf{D}_L \boldsymbol{\gamma}_L$ will be a random variable $\gamma_{L-1}^j \sim \mathcal{N}(0, \sigma_L^2)$, $\forall\ j$. Thus, $\Pr(\gamma_{L-1}^j = 0) = 0$. As we see, there will not exist any sparse (or dense, for that matter) $\boldsymbol{\gamma}_L$ which will create a sparse $\boldsymbol{\gamma}_{L-1}$. In other words, for this choice of dictionaries, the ML-CSC model is empty.

A. Sparse Dictionaries

From the discussion above one can conclude that one of the key components of the ML-CSC model is sparse dictionaries: if both $\boldsymbol{\gamma}_L$ and $\boldsymbol{\gamma}_{L-1} = \mathbf{D}_L \boldsymbol{\gamma}_L$ are sparse, then atoms in $\mathbf{D}$ must indeed contain only a few non-zeros. We make this observation concrete in the following lemma.

Lemma 3: Dictionary Sparsity Condition
Consider the ML-CSC model $\mathcal{M}_{\boldsymbol{\lambda}}$ described by the dictionaries $\{\mathbf{D}_i\}_{i=1}^L$ and the layer-wise $\ell_{0,\infty}$-sparsity levels $\lambda_1, \lambda_2, \ldots, \lambda_L$. Given $\boldsymbol{\gamma}_L : \|\boldsymbol{\gamma}_L\|_{0,\infty}^s \le \lambda_L$ and constants $c_i =$

[4]Instead of repeating the pursuit from scratch at every iteration, one might warm start the OMP algorithm by employing current estimate, $\hat{\boldsymbol{\gamma}}_L$, as initial condition so that only new non-zeros are added.

$\lceil \frac{2n_{i-1}-1}{n_i} \rceil$, the signal $\mathbf{x} = \mathbf{D}^{(L)}\boldsymbol{\gamma}_L \in \mathcal{M}_{\boldsymbol{\lambda}}$ if

$$\|\mathbf{D}_i\|_0 \leq \frac{\lambda_{i-1}}{\lambda_i c_i}, \quad \forall\, 1 < i \leq L. \tag{15}$$

The simple proof of this Lemma is included in the Supplementary Material V. Notably, while this claim does not tell us if a certain model is empty, it does guarantee that if the dictionaries satisfy a given sparsity constraint, one can simply sample from the model by drawing the inner-most representations such that $\|\boldsymbol{\gamma}_L\|_{0,\infty}^s \leq \lambda_L$. One question remains: how do we train such dictionaries from real data?

B. Learning Formulation

One can understand from the previous discussion that there is no hope in solving the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem for real signals without also addressing the learning of dictionaries $\mathbf{D}_i$ that would allow for the respective representations. To this end, considering the scenario where one is given a collection of K training signals, $\{\mathbf{y}^k\}_{k=1}^K$, we upgrade the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem to a learning setting in the following way:

$$\min_{\{\boldsymbol{\gamma}_i^k\},\{\mathbf{D}_i\}} \sum_{k=1}^K \|\mathbf{y}^k - \mathbf{x}^k(\boldsymbol{\gamma}_i^k, \mathbf{D}_i)\|_2^2 \quad \text{s.t.} \begin{cases} \mathbf{x}^k \in \mathcal{M}_{\boldsymbol{\lambda}}, \\ \|\mathbf{d}_i^j\|_2 = 1, \forall\, i, j. \end{cases} \tag{16}$$

We have included the constraint of every dictionary atom to have a unit norm to prevent arbitrarily small coefficients in the representations $\boldsymbol{\gamma}_i^k$. This formulation, while complete, is difficult to address directly: The constraints on the representations $\boldsymbol{\gamma}_i$ are coupled, just as in the pursuit problem discussed in the previous section. In addition, the sparse representations now also depend on the variables $\mathbf{D}_i$. In what follows, we provide a relaxation of this cost function that will result in a simple learning algorithm.

The problem above can also be understood from the perspective of minimizing the number of non-zeros in the representations at every layer, subject to an error threshold – a typical reformulation of sparse coding problems. Our main observation arises from the fact that, since $\boldsymbol{\gamma}_{L-1}$ is function of both $\mathbf{D}_L$ and $\boldsymbol{\gamma}_L$, one can upper-bound the number of non-zeros in $\boldsymbol{\gamma}_{L-1}$ by that of $\boldsymbol{\gamma}_L$. More precisely,

$$\|\boldsymbol{\gamma}_{L-1}\|_{0,\infty}^s \leq c_L \|\mathbf{D}_L\|_0 \|\boldsymbol{\gamma}_L\|_{0,\infty}^s, \tag{17}$$

where c_L is a constant[5]. Therefore, instead of minimizing the number of non-zeros in $\boldsymbol{\gamma}_{L-1}$, we can address the minimization of its upper bound by minimizing both $\|\boldsymbol{\gamma}_L\|_{0,\infty}^s$ and $\|\mathbf{D}_L\|_0$. This argument can be extended to any layer, and we can generally write

$$\|\boldsymbol{\gamma}_i\|_{0,\infty}^s \;\leq\; c \prod_{j=i+1}^{L} \|\mathbf{D}_j\|_0 \|\boldsymbol{\gamma}_L\|_{0,\infty}^s. \tag{18}$$

In this way, minimizing the sparsity of any ith representation can be done implicitly by minimizing the sparsity of the last layer and the number of non-zeros in the dictionaries from layer $(i+1)$ to L. Put differently, the sparsity of the intermediate convolutional dictionaries serve as proxies for the sparsity of the respective representation vectors. Following this observation, we now recast the problem in Equation (16) into the following Multi-Layer Convolutional Dictionary Learning Problem:

$$\min_{\{\boldsymbol{\gamma}_L^k\},\{\mathbf{D}_i\}} \sum_{k=1}^K \|\mathbf{y}^k - \mathbf{D}_1\mathbf{D}_2\ldots\mathbf{D}_L\boldsymbol{\gamma}_L^k\|_2^2 + \sum_{i=2}^L \zeta_i \|\mathbf{D}_i\|_0 \quad \text{s.t.} \begin{cases} \|\boldsymbol{\gamma}_L^k\|_{0,\infty}^s \leq \lambda_L, \\ \|\mathbf{d}_i^j\|_2 = 1, \forall\, i, j. \end{cases} \tag{19}$$

Under this formulation, this problem seeks for sparse representations $\boldsymbol{\gamma}_L^k$ for each example $\mathbf{y}^k$, while forcing the intermediate convolutional dictionaries (from layer 2 to L) to be sparse. The reconstructed signal, $\mathbf{x} = \mathbf{D}_1\boldsymbol{\gamma}_1$, is not expected to be sparse, and so there is no reason to enforce this property on $\mathbf{D}_1$. Note that there is now only one sparse coding process involved – that of $\boldsymbol{\gamma}_L^k$ – while the intermediate representations are never computed explicitly. Recalling the theoretical results from the previous section, this is in fact convenient as one only has to estimate the representation for which the recovery bound is the tightest.

Following the theoretical guarantees presented in Section III, one can alternatively replace the $\ell_{0,\infty}$ constraint on the deepest representation by a convex ℓ_1 alternative. The resulting formulation resembles the lasso formulation of the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem, for which we have presented theoretical guarantees in Theorem 6. In addition, we replace the constraint on the ℓ_2 of the dictionary atoms by an appropriate penalty term, recasting the above problem into a simpler (unconstrained) form:

$$\min_{\{\boldsymbol{\gamma}_L^k\},\{\mathbf{D}_i\}} \sum_{k=1}^K \|\mathbf{y}^k - \mathbf{D}_1\mathbf{D}_2\ldots\mathbf{D}_L\boldsymbol{\gamma}_L^k\|_2^2 + \iota \sum_{i=1}^L \|\mathbf{D}_i\|_F^2 + \sum_{i=2}^L \zeta_i \|\mathbf{D}_i\|_0 + \lambda \|\boldsymbol{\gamma}_L^k\|_1, \tag{20}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The problem in Equation (20) is highly non-convex, due to the ℓ_0 terms and the product of the factors. In what follows, we present an online alternating minimization algorithm, based on stochastic gradient descent, which seeks for the deepest representation $\boldsymbol{\gamma}_L$ and then progressively updates the layer-wise convolutional dictionaries.

For each incoming sample $\mathbf{y}^k$ (or potentially, a mini-batch), we will first seek for its deepest representation $\boldsymbol{\gamma}_L^k$ considering the dictionaries fixed. This is nothing but the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem in (7), which was analyzed in detail in the previous sections, and its solution will be approximated through iterative shrinkage algorithms. Also, one should keep in mind that while representing each dictionary by $\mathbf{D}_i$ is convenient in terms of notation, these matrices are never computed explicitly – which would be prohibitive. Instead, these dictionaries (or their transpose) are applied effectively through convolution operators. In turn, this implies that images are not vectorized but processed as 2 dimensional matrices (or 3-dimensional tensors for multi-channel

[5]From [17], we have that $\|\boldsymbol{\gamma}_{L-1}\|_{0,\infty}^p \leq \|\mathbf{D}_L\|_0 \|\boldsymbol{\gamma}_L\|_{0,\infty}^s$. From here, and denoting by c_L the upper-bound on the number of patches in a stripe from $\boldsymbol{\gamma}_{L-1}$ given by $c_L = \lceil \frac{2n_{L-1}-1}{n_L} \rceil$, we can obtain a bound to $\|\boldsymbol{\gamma}_{L-1}\|_{0,\infty}^s$.

Algorithm 3: Multi-Layer Convolutional Dictionary Learning.

Data: Training samples $\{\mathbf{y}_k\}_{k=1}^K$, initial convolutional dictionaries $\{\mathbf{D}_i\}_{i=1}^L$
for $k = 1, \ldots, K$ **do**
  Draw $\mathbf{y}_k$ at random;
  Sparse Coding:
  $\boldsymbol{\gamma}_L \leftarrow \arg\min_{\boldsymbol{\gamma}} \|\mathbf{y}_k - \mathbf{D}^{(L)}\boldsymbol{\gamma}\|_2 + \lambda\|\boldsymbol{\gamma}\|_1$;
  Update Dictonaries:
  for $i = L, \ldots, 2$ **do**
    for t = *1,...,T* **do**
      $\mathbf{D}_i^{t+1} \leftarrow \mathcal{H}_{\zeta_i}\left[\mathbf{D}_i^t - \eta\nabla f(\mathbf{D}_i^t)\right]$;
  for t = *1,...,T* **do**
    $\mathbf{D}_1^{t+1} \leftarrow \mathbf{D}_1^t - \eta\nabla f(\mathbf{D}_1^t)$;

images). In addition, these operators are very efficient due to their high sparsity, and one could in principle benefit from specific libraries to boost performance in this case, such as the one in [40].

Given the obtained $\boldsymbol{\gamma}_L^k$, we then seek to update the respective dictionaries. As it is posed – with a global ℓ_0 norm over each dictionary – this is nothing but a generalized pursuit as well. Therefore, for each dictionary $\mathbf{D}_i$, we minimize the function in Problem (20) by applying T iterations of projected gradient descent. This is done by computing the gradient of the ℓ_2 terms in Problem (20) (call it $f(\mathbf{D}_i)$) with respect to a each dictionary $\mathbf{D}_i$ (i.e., $\nabla f(\mathbf{D}_i)$), making a gradient step and then applying a hard-thresholding operation, $\mathcal{H}_{\zeta_i}(\cdot)$, depending on the parameter ζ_i. This is simply an instance of the Iterative Hard Thresholding algorithm [41]. In addition, the computation of $\nabla f(\mathbf{D}_i)$ involves only multiplications the convolutional dictionaries for the different layers. The overall algorithm is depicted in Algorithm 3, and we will expand on further implementation details in the results section.

The parameters of the models involve the ℓ_1 penalty of the deepest representation, i.e. λ, and the parameter for each dictionary, ζ_i. The first parameter can be set manually or determined so as to obtain a given given representation error. On the other hand, the dictionary-wise ζ_i parameters are less intuitive to establish, and the question of how to set these values for a given learning scenario remains a subject of current research. Nevertheless, we will show in the experimental section that setting these manually results in effective constructions.

Note this approach can also be employed to minimize Problem (19) by introducing minor modifications: In the sparse coding stage, the Lasso is replaced by a $\ell_{0,\infty}$ pursuit, which can be tackled with a greedy alternative as the OMP (as described in Theorem 7) or by an Iterative Hard Thresholding alternative [41]. In addition, one could consider employing the ℓ_1 norm as a surrogate for the ℓ_0 penalty imposed on the dictionaries. In this case, their update can still be performed by the same projected gradient descent approach, though replacing the hard thresholding with its soft counterpart.

Before moving on, and even though an exhaustive computational complexity analysis is out of the scope of this paper, we want to briefly comment on the general aspects of the algorithm's complexity. For a particular network architecture (number of layers, number of filters per layer, filter sizes, etc) let us denote by $\mathcal{C}$ the complexity of applying the forward pass – or in other words, multiplying by $\mathbf{D}^{(L)^T}$ – on an input image, or a minibach (i.e., for each kth iteration). The sparse coding step in our algorithm is carried with iterative shrinkage methods, and assuming these algorithms are run for τ iterations, the complexity incurred in each sparse coding step is[6] $\mathcal{O}(\tau\mathcal{C})$. The update of the dictionaries, on the other hand, requires computing the gradient for each set of filters. Each of these gradients involves, roughly speaking, the computation of yet another forward and backward pass[7]. In this way, the dictionary update stage is $\mathcal{O}(LT\mathcal{C})$. Note that we are disregarding the shrinkage operators both on the representations and on the filters, which are entry-wise operations that are negligible when compared to applying $\mathbf{D}^{(L)}$ or its transpose. As can be seen, the complexity of our algorithm is approximately $(\tau + TL)$ times that of a similar CNNs architectures. Finally, note that we are not considering the (important) fact that, in our case, the convolutional kernels are sparse, and as such they may incur in significantly cheaper computations. This precise analysis, and how to maximize the related computational benefit, is left for future work.

C. Connection to Related Works

Naturally, the proposed algorithm has tight connections to several recent dictionary learning approaches. For instance, our learning formulation is closely related to the Chasing Butterflies approach in [18], and our resulting algorithm can be interpreted as a particular case of the FAUST method, proposed in the inspiring work from [42]. FAUST decomposes linear operators into sparse factors in a hierarchical way in the framework of a batch learning algorithm, resulting in improved complexity. Unlike that work, our multi-layer decompositions are not only sparse but also convolutional, and they are updated within a stochastic optimization framework. The work in [19], on the other hand, proposed a learning approach where the dictionary is expressed as a cascade of convolutional filters with sparse kernels, and they effectively showed how this approach can be used to approximate large-dimensional analytic atoms such as those from wavelets and curvelets. As our proposed approach effectively learns a sparse dictionary, our work also shares similarities with the double-sparsity work from [43]. In particular, in its Trainlets version [20], the authors proposed to learn a dictionary as a sparse combination of cropped wavelets atoms. From the previous comment on the work from [19], this could also potentially be expressed as a product of sparse convolutional atoms. All these works, as well as our approach, essentially enforce extra regularization into the dictionary learning problem.

[6]Each such iteration actually involves the application of a forward and backward pass, resulting from the fact that one needs to apply $\mathbf{D}^{(L)}$ and $\mathbf{D}^{(L)^T}$.

[7]The dictionary gradients can actually be computed more efficiently if intermediate computations are saved (and stored), incurring in $\mathcal{O}(L\log_2(L))$ convolution operators. Thus, in this case the dictionary update stage is $\mathcal{O}(\log_2(L)T\mathcal{C})$. We defer the implementation of this more efficient algorithm for future work.

As a result, these methods perform better in cases with corrupted measurements, in high dimensional settings, and in cases with limited amount of training data (see [20], [43]).

What is the connection between this learning formulation and that of deep convolutional networks? Recalling the analysis presented in [17], the Forward Pass is nothing but a layered non-negative thresholding algorithm, the simplest form of a pursuit for the ML-CSC model with layer-wise deviations. Therefore, if the pursuit for $\hat{\boldsymbol{\gamma}}_L$ in our setting is solved with such an algorithm, then the problem in (20) *implements a convolutional neural network with only one RELU operator at the last layer, with sparse-enforcing penalties on the filters.* Moreover, due the data-fidelity term in our formulation, the proposed optimization problem provides nothing but a convolutional sparse autoencoder. As such, our work is related to the extensive literature in this topic. For instance, in [21], sparsity is enforced in the hidden activation layer by employing a penalty term proportional to the KL divergence between the hidden unit marginals and a target sparsity probability.

Other related works include the k-sparse autoencoders [22], where the hidden layer is constrained to having exactly k non-zeros. In practice, this boils down to a hard thresholding step of the hidden activation, and the weights are updated with gradient descent. In this respect, our work can be thought of a generalization of this work, where the pursuit algorithm is more sophisticated than a simple thresholding operation, and where the filters are composed by a cascade of sparse convolutional filters. More recently, the work in [23] proposed the *winner-take-all* autoencoders. In a nutshell, these are non-symmetric autoencoders having a few convolutional layers (with ReLu non-linearities) as the encoder, and a simple linear decoder. Sparsity is enforced in what the authors refer to as "spatial" and a "lifetime" sparsity.

Finally, and due to the fact that our formulation effectively provides a convolutional network with sparse kernels, our approach is reminiscent of works attempting to sparsify the filters in deep learning models. For instance, the work in [40] showed that the weights of learned deep convolutional networks can be sparsified without considerable degradation of classification accuracy. Nevertheless, one should perpend the fact that these works are motivated merely by cheaper and faster implementations, whereas our model is intrinsically built by theoretically justified sparse kernels. We do not attempt to compare our approach to such sparsifying methods at this stage, and we defer this to future work.

In light of all these previous works, the practical contribution of the learning algorithm presented here is to demonstrate, as we will see in the following Experiments section, that our online block-coordinate descent method can be effectively deployed in an unsupervised setting competing favorably with state of the art dictionary learning and convolutional network auto-encoders approaches.

V. Experiments

We now provide experimental results to demonstrate several aspects of the ML-CSC model. As a case-study, we consider the MNIST dataset [44]. We define our model as consisting of 3 convolutional layers: the first one contains 32 local filters of size 7×7 (with a stride of 2), the second one consists of 128 filters of dimensions $5 \times 5 \times 32$ (with a stride of 1), and the last one contains 1024 filters of dimensions $7 \times 7 \times 128$. At the third layer, the effective size of the atoms is 28 – representing an entire digit.

Training is performed with Algorithm 3, using a mini-batch of 100 samples per iteration. For the Sparse Coding stage, we leverage an efficient implementation of FISTA [45], and we adjust the penalty parameter λ to obtain roughly 15 non-zeros in the deepest representation $\boldsymbol{\gamma}_3$. The ζ_i parameters, the penalty parameters for the dictionaries sparsity levels, are set manually for simplicity. In addition, and as it is commonly done in various Gradient Descent methods, we employ a momentum term for the update of the dictionaries $\mathbf{D}_i$ within the projected gradient descent step in Algorithm 3, and set its memory parameter to 0.9. The step size is set to 1, the update dictionary iterations is set as $T = 1$, $\iota = 0.001$, and we run the algorithm for 20 epochs, which takes approximately 30 minutes. Our implementation uses the Matconvnet library, which leverages efficient functions for GPU[8]. No pre-processing was performed, with the exception of the subtraction of the mean image (computed on the training set).

We depict the evolution of the Loss function during training in Fig. 4, as well as the sparsity of the second and third dictionaries (i.e., 1 minus the number of non-zero coefficients in the filters relative to the filters dimension) and the average residual norm. The resulting model is depicted in Fig. 3. One can see how the first layer is composed of very simple small-dimensional edges or blobs. The second dictionary, $\mathbf{D}_2$, is effectively 99% sparse, and its non-zeros combine a few atoms from $\mathbf{D}_1$ in order to create slightly more complex edges, as the ones in the effective dictionary $\mathbf{D}^{(2)}$. Lastly, $\mathbf{D}_3$ is 99.8% sparse, and it combines atoms from $\mathbf{D}^{(2)}$ in order to provide atoms that resemble different kinds (or parts) of digits. These final global atoms are nothing but a linear combination of local small edges by means of convolutional sparse kernels.

Interestingly, we have observed that the mutual coherence of the effective dictionaries do not necessarily increase with the layers, and they often decrease with the depth. While this measure relates to worst-case analysis conditions and do not mean much in the context of practical performance, one can see that the effective dictionary indeed becomes less correlated as the depth increases. This is intuitive, as very simple edges – and at every location – are expected to show large inner products, larger than the correlation of two more complex number-like structures. This effect can be partially explained by the dictionary redundancy: having 32 local filters in $\mathbf{D}_1$ (even while using a stride of 2) implies a 8-fold redundancy in the effective dictionary at this level. This redundancy decreases with the depth (at this least for the current construction), and at the third layer one has *merely* 1024 atoms (redundancy of about 1.3, since the signal dimension is 28^2).

We can also find the multi-layer representation for real images – essentially solving the projection problem $\mathcal{P}_{\mathcal{M}_\lambda}$. In Fig. 5, we depict the multi-layer features $\boldsymbol{\gamma}_i$, $i = 1, 2, 3$, obtained with the

[8]All experiments are run on a 16 i7 cores Windows station with a NVIDIA GTX 1080 Ti.

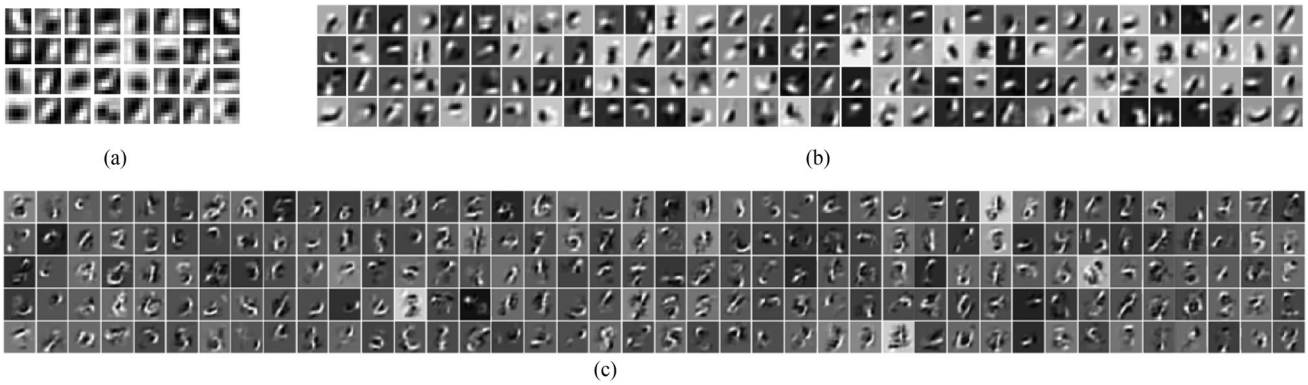


Fig. 3. ML-CSC model trained on the MNIST dataset. (a) The local filters of the dictionary $\mathbf{D}_1$. (b) The local filters of the effective dictionary $\mathbf{D}^{(2)} = \mathbf{D}_1\mathbf{D}_2$. (c) Some of the 1024 local atoms of the effective dictionary $\mathbf{D}^{(3)}$ which, because of the dimensions of the filters and the strides, are global atoms of size 28×28.

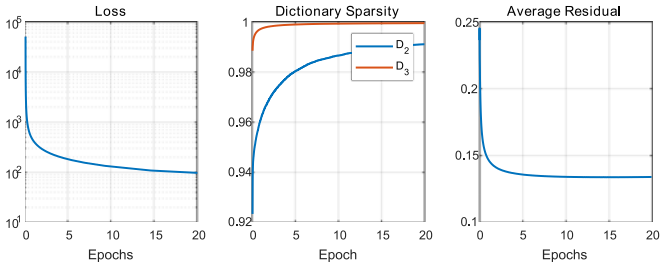


Fig. 4. Evolution of the Loss function, sparsity of the convolutional dictionaries and average residual norm during training on the MNIST dataset.

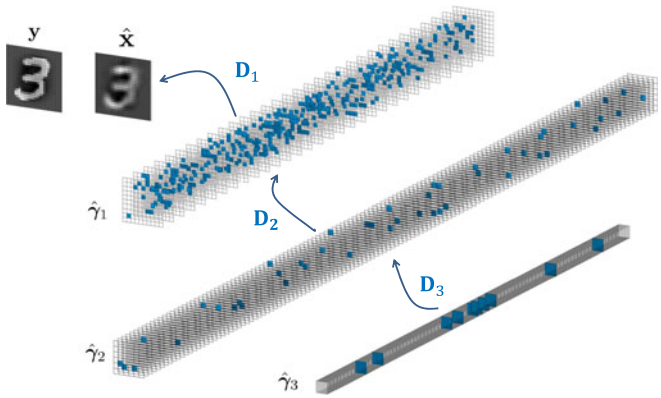


Fig. 5. Decompositions of an image from MNIST in terms of its nested sparse features $\boldsymbol{\gamma}_i$ and multi-layer convolutional dictionaries $\mathbf{D}_i$.

Algorithm 1, that approximate an image $\mathbf{y}$ (not included in the training set). Note that all the representations are notably sparse thanks to the very high sparsity of the dictionaries $\mathbf{D}_2$ and $\mathbf{D}_3$. These decompositions (any of them) provide a sparse decomposition of the number 3 at different scales, resulting in an approximation $\hat{\mathbf{x}}$. Naturally, the quality of the approximation can be improved by increasing the cardinality of the representations.

A. Sparse Recovery

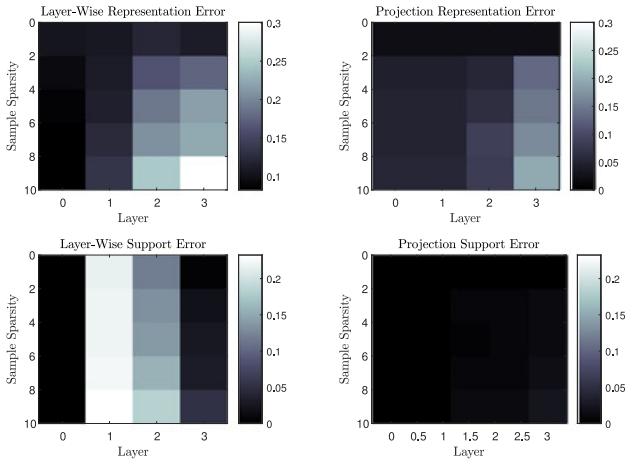
The first experiment we explore is that of recovering sparse vectors from corrupted measurements, in which we will compare the presented ML-CSC Pursuit with the Layered approach from [17]. For the sake of completion and understanding, we will first carry this experiment in a synthetic setting and then on projected real digits, leveraging the dictionaries obtained in the beginning of this section.

We begin by constructing a 3 layers "non-convolutional"[9] model for signals of length 200, with the dictionaries having 250, 300, and 350 atoms, respectively. The first dictionary is constructed as a random matrix, whereas the remaining ones are composed of sparse atoms with random supports and a sparsity of 99%. Finally, 500 representations are sampled by drawing sparse vectors $\boldsymbol{\gamma}_L$, with a target sample sparsity k and normally distributed coefficients. We generate the signals as $\mathbf{x} = \mathbf{D}^{(i)}\boldsymbol{\gamma}_i$, and then corrupt them with Gaussian noise ($\sigma = 0.02$) obtaining the measurements $\mathbf{y} = \mathbf{x}(\boldsymbol{\gamma}_i) + \mathbf{v}$.

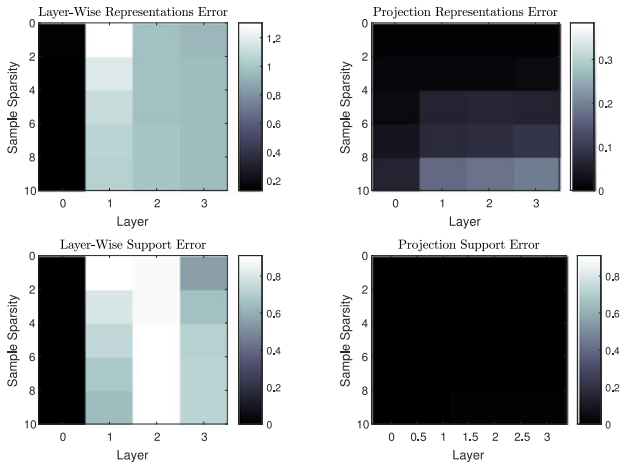
In order to evaluate our projection approach, we run Algorithm 1 employing the Subspace Pursuit algorithm [46] for the sparse coding step, with the oracle target cardinality k. Recall that once the deepest representations $\hat{\boldsymbol{\gamma}}_L$ have been obtained, the inner ones are simply computed as $\hat{\boldsymbol{\gamma}}_{i-1} = \mathbf{D}_i\hat{\boldsymbol{\gamma}}_i$. In the layered approach from [17], on the other hand, the pursuit of the representations progresses sequentially: first running a pursuit for $\hat{\boldsymbol{\gamma}}_1$, then employing this estimate to run another pursuit for $\hat{\boldsymbol{\gamma}}_2$, etc. In the same spirit, we employ Subspace Pursuit layer by layer, employing the oracle cardinality of the representation at each stage. The results are presented in Fig. 6(a): at the top we depict the relative ℓ_2 error of the recovered representations ($\|\hat{\boldsymbol{\gamma}}_i - \boldsymbol{\gamma}_i\|_2 / \|\boldsymbol{\gamma}_i\|_2$) and, at the bottom, the normalized intersection of the supports [47], both as a function of the sample cardinality k and the layer depth.

The projection algorithm manages to retrieve the representations $\hat{\boldsymbol{\gamma}}_i$ more accurately than the layered pursuit, as evidenced by the ℓ_2 error and the support recovery. The main reason behind the difficulty of the layer-by-layer approach is that the entire process relies on the correct recovery of the first layer representations, $\hat{\boldsymbol{\gamma}}_1$. If these are not properly estimated (as evidenced by the bottom-left graph), there is little hope for the

[9]The non-convolutional case is still a ML-CSC model, in which the signal dimension is the same as the length of the atoms n, and with a stride of the same magnitude n. We choose this setting for the synthetic experiment to somewhat favor the results of the layered pursuit approach.



(a) Synthetic signals.



(b) MNIST signals.

Fig. 6. Recovery of representations from noisy MNIST digits. Top: normalized ℓ_2 error between the estimated and the true representations. Bottom: normalized intersection between the estimated and the true support of the representations.

recovery of the deeper ones. In addition, these representations $\boldsymbol{\gamma}_1$ are the least sparse ones, and so they are expected to be the most challenging ones to recover. The projection alternative, on the other hand, relies on the estimation of the deepest $\hat{\boldsymbol{\gamma}}_L$, which are very sparse. Once these are estimated, the remaining ones are simply computed by propagating them to the shallower layers. Following our analysis in the Section III-C, if the support of $\hat{\boldsymbol{\gamma}}_L$ is estimated correctly, so will be the support of the remaining representations $\hat{\boldsymbol{\gamma}}_i$.

We now turn to deploy the 3 layer convolutional dictionaries for real digits obtained previously. To this end we take 500 test digits from the MNIST dataset and project them on the trained model, essentially running Algorithm 1 and obtaining the representations $\boldsymbol{\gamma}_i$. We then create the noisy measurements as $\mathbf{y} = \mathbf{D}^{(i)}\boldsymbol{\gamma}_i + \mathbf{v}$, where $\mathbf{v}$ is Gaussian noise with $\sigma = 0.02$. We then repeat both pursuit approaches to estimate the underlying representations, obtaining the results reported in Fig. 6(b).

Clearly, this represents a significantly more challenging scenario for the layered approach, which recovers only a small

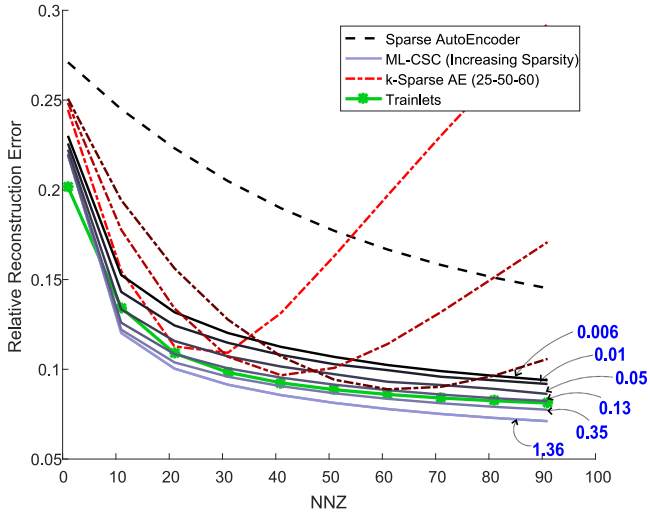


Fig. 7. M-term approximation for MNIST digits, comparing sparse autoencoders [21], k-sparse autoencoders [22], trainlets (OSDL) [20], and the proposed ML-CSC for models with different filter sparsity levels. The relative number of parameters is depicted in blue.

fraction of the correct support of the sparse vectors. The projection algorithm, on the other hand, provides accurate estimations with negligible mistakes in the estimated supports, and very low ℓ_2 error. Note that the ℓ_2 error has little significance for the Layered approach, as this algorithm does not manage to find the true supports. The reason for the significant deterioration in the performance of the Layered algorithm is that this method actually finds alternative representations $\hat{\boldsymbol{\gamma}}_1$, of the same sparsity, providing a lower fidelity term than the projection counterpart for the first layer. However, these estimates $\hat{\boldsymbol{\gamma}}_1$ do not necessarily provide a signal in the model, which causes further errors when estimating $\hat{\boldsymbol{\gamma}}_2$.

B. Sparse Approximation

A straight forward application for unsupervised learned model is that of approximation: how well can one approximate or reconstruct a signal given only a few k non-zero values from some representation? In this subsection, we study the performance of the ML-CSC model for this task while comparing with related methods, and we present the results in Fig. 7. The model is trained on $60K$ training examples, and the M-term approximation is measured on the remaining $10K$ testing samples. All of the models are designed with 1K hidden units (or atoms).

Given the close connection of the ML-CSC model to sparse auto-encoders, we present the results obtained by approximating the signals with sparse autoencoders [21] and k-sparse autoencoders [22]. In particular, the work in [21] trains sparse auto-encoders by penalizing the KL divergence between the activation distribution of the hidden neurons and that of a binomial distribution with a certain target activation rate. As such, the resulting activations are never truly sparse. For this reason, since the M-term approximation is computed by picking the highest entries in the hidden neurons and setting the remaining ones to zero, this method exhibits a considerable representation error.

K-sparse auto-encoders perform significantly better, though they are sensitive to the number of non-zeros used during training. Indeed, if the model is trained with 25 non-zeros per sample, the model performs well for a similar range of cardinalities. Despite this sensitivity on training, their performance is remarkable considering the simplicity of the pursuit involved: the reconstruction is done by computing $\hat{\mathbf{x}} = \mathbf{W}\hat{\boldsymbol{\gamma}}_k + \mathbf{b}'$, where $\hat{\boldsymbol{\gamma}}_k$ is a k-sparse activation (or feature) obtained by hard thresholding as $\hat{\boldsymbol{\gamma}}_k = H_k\left[\mathbf{W}^T\mathbf{y} + \mathbf{b}\right]$, and where $\mathbf{b}$ and $\mathbf{b}'$ are biases vectors. Note that while a convolutional multi-layer version of this family of autoencoders was proposed in [23], these constructions are trained in stacked manner – i.e., training the first layer independently, then training the second one to represent the features of the first layer while introducing pooling operations, and so forth. In this manner, each layer is trained to represent the (pooled) features from the previous layer, but the entire architecture cannot be directly employed for comparison in this problem.

Regarding the ML-CSC, we trained 6 different models by enforcing 6 different levels of sparsity in the convolutional filters (i.e., different values of the parameters ζ_i in Algorithm 3), with a fixed target sparsity of $k = 10$ non-zeros. The sparse coding of the inner-most $\hat{\boldsymbol{\gamma}}_3$ was done with the Iterative Hard Thresholding algorithm, in order to guarantee an exact number of non-zeros. The numbers pointing at the different models indicate the relative amount of parameters in the model, where 1 corresponds to $28^2 \times 1K$ parameters required in a standard autoencoder (this is also the number of parameters in the sparse-autoencoders and k-sparse autoencoders, without counting the biases). As one can see, the larger the number of parameters, the lower the representation error the model is able to provide. In particular, the ML-CSC yields slightly better representation error than that of k-sparse autoencoders, for a wide range of non-zero values (without the need to train different models for each one) and *with 1 and 2 orders of magnitude less parameters*.

Since the training of the ML-CSC model can also be understood as a dictionary learning algorithm, we compare here with the state-of-the-art method of [20]. For this case, we trained 1K trainlet atoms with the OSDL algorithm. Note that this comparison is interesting, as OSDL also provides sparse atoms with reduced number of parameters. For the sake of comparison, we employed an atom-sparsity that results in 13% of parameters relative to the total model size (just as one of the trained ML-CSC models), and the sparse coding was done also with the IHT algorithm. Notably, the performance of this relatively sophisticated dictionary learning method, which leverages the representation power of a cropped wavelets base dictionary, is only slightly superior to the proposed ML-CSC.

C. Unsupervised Classification

Unsupervised trained models are usually employed as feature extractors, and a popular way to assess the quality of such features is to train a linear classifier on them for a certain classification task. While the intention of this paper is not to provide a state-of-the-art unsupervised learning algorithm, we simply intent to demonstrate that the learned model generalizes to unseen examples, providing meaningful representations. To this end, we train a model with 3 layers, each containing: 16 (5×5) atoms, 64 ($5 \times 5 \times 16$) atoms and 1024 atoms of dimension $5 \times 5 \times 64$ (stride of 2) on 60K training samples from MNIST. Just as for the previous model, the global sparse coding is performed with FISTA and a target (average) sparsity of 25 non-zeros. Once trained, we compute the representations $\hat{\gamma}_i$ with an elastic net formulation and non-negativity constraints, before fitting a simple linear classifier on the obtained features. Employing an elastic-net formulation (by including an ℓ_2 regularization parameter, in addition to the ℓ_1 norm) results in slightly denser representations, with improved classification performance. Similarly, the non-negativity constraint significantly facilitates the classification by linear classifiers. We compare our results with similar methods under the same experimental setup, and we depict the results in Table I, reporting the classification error on the 10K testing samples.

TABLE I
UNSUPERVISED CLASSIFICATION RESULTS ON MNIST

Method	Test Error
Stacked Denoising Autoencoder (3 layers) [49]	1.28%
k-Sparse Autoencoder (1K units) [22]	1.35%
Shallow WTA Autoencoder (2K units) [23]	1.20%
Stacked WTA Autoencoder (2K units)[23]	1.11%
ML-CSC (1K units) - 2nd Layer Rep.	1.30%
ML-CSC (2K units) - 2nd&3rd Layer Rep.	1.15%

Recall that within the ML-CSC model, all features $\boldsymbol{\gamma}_i$ have a very clear meaning: they provide a sparse representation at a different layer and scale. We can leverage this multi-layer decomposition in a very natural way within this unsupervised classification framework. We detail the classification performance achieved by our model in two different scenarios: on the first one we employ the 1K-dimensional features corresponding to the second layer of the ML-CSC model, obtaining better performance than the equivalent k-sparse autoencoder. In the second case, we add to the previous features the 1K-dimensional features from the third layer, resulting in a classification error of 1.15%, comparable to the Stacked Winner Take All (WTA) autoencoder (with the same number of neurons).

Lastly, it is worth mentioning that a stacked version of convolutional WTA autoencoder [23] achieve a classification error of 0.48, providing significantly better results. However, note that this model is trained with a 2-stage process (training the layers separately) involving significant pooling operations between the features at different layers. More importantly, the features computed by this model are 51,200-dimensional (more than an order of magnitude larger than in the other models) and thus cannot be directly compared to the results reporter by our method. In principle, similar stacked-constructions that employ pooling could be built for our model as well, and this remains as part of ongoing work.

VI. CONCLUSION

We have carefully revisited the ML-CSC model and explored the problem of projecting a signal onto it. In doing so, we have provided new theoretical bounds for the solution of this problem

as well as stability results for practical algorithms, both greedy and convex. The search for signals within the model led us to propose a simple, yet effective, learning formulation adapting the dictionaries across the different layers to represent natural images. We demonstrated the proposed approach on a number of practical applications, showing that the ML-CSC can indeed provide significant expressiveness with a very small number of model parameters.

Several question remain open: how should the model be modified to incorporate pooling operations between the layers? what consequences, both theoretical and practical, would this have? How should one recast the learning problem in order to address supervised and semi-supervised learning scenarios? Lastly, we envisage that the analysis provided in this work will empower the development of better practical and theoretical tools not only for structured dictionary learning approaches, but to the field of deep learning and machine learning in general.

ACKNOWLEDGMENT

J. Sulam would like to thank J. Turek for fruitful discussions.

REFERENCES

[1] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Rev.*, vol. 51, pp. 34–81, Feb. 2009.

[2] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *IEEE Proc.*, vol. 98, no. 6, pp. 1045–1057, Jun. 2010.

[3] J. Sulam, B. Ophir, and M. Elad, "image denoising through multi-scale learnt dictionaries," in *Proc. IEEE Int. Conf. Image Process.*, 2014, pp. 808–812.

[4] Y. Romano, M. Protter, and M. Elad, "Single image interpolation via adaptive nonlocal sparsity-based modeling," *IEEE Trans. Image Process.*, vol. 23, no. 7, pp. 3085–3098, Jul. 2014.

[5] J. Mairal, F. Bach, and G. Sapiro, "Non-local sparse models for image restoration," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, 2009, vol. 2, pp. 2272–2279.

[6] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent K-SVD: Learning a discriminative dictionary for recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2651–2664, Nov. 2013.

[7] V. M. Patel, Y.-C. Chen, R. Chellappa, and P. J. Phillips, "Dictionaries for image and video-based face recognition," *J. Opt. Soc. Amer. A*, vol. 31, no. 5, pp. 1090–1103, 2014.

[8] A. Shrivastava, V. M. Patel, and R. Chellappa, "Multiple kernel learning for sparse representation-based classification," *IEEE Trans. Image Process.*, vol. 23, no. 7, pp. 3013–3024, Jul. 2014.

[9] Y. LeCun *et al.*, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*. San Mateo, CA, USA: Morgan Kaufmann, 1990, pp. 396–404.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[12] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, Aug. 2013.

[13] A. Patel, T. Nguyen, and R. Baraniuk, "A Probabilistic framework for deep learning," Proc. NIPS 2016, Barcelona, Spain.

[14] N. Cohen, O. Sharir, and A. Shashua, "On the expressive power of deep learning: A tensor analysis," in *Proc. 29th Annu. Conf. Learn. Theory*, Jun. 23–26, 2016, pp. 698–728.

[15] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 399–406.

[16] B. Xin, Y. Wang, W. Gao, D. Wipf, and B. Wang, "Maximal sparsity with deep networks?" in *Proc. Conf. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4340–4348.

[17] V. Papyan, Y. Romano, and M. Elad, "Convolutional neural networks analyzed via convolutional sparse coding," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 2887–2938, 2017.

[18] L. Le Magoarou and R. Gribonval, "Chasing butterflies: In search of efficient dictionaries," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2015, pp. 3287–3291.

[19] O. Chabiron, F. Malgouyres, J. Tourneret, and N. Dobigeon, "Toward fast transform learning," *Int. J. Comput. Vis.*, vol. 114, pp. 195–216, 2015.

[20] J. Sulam, B. Ophir, M. Zibulevsky, and M. Elad, "Trainlets: Dictionary learning in high dimensions," *IEEE Trans. Signal Process.*, vol. 64, no. 12, pp. 3180–3193, Jun. 2016.

[21] A. Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[22] A. Makhzani and B. Frey, "k-sparse autoencoders," *Proc. Int. Conf. Learn. Represent.*, Banff, Canada, 2014.

[23] A. Makhzani and B. J. Frey, "Winner-take-all autoencoders," in *Proc. Conf. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2791–2799.

[24] V. Papyan, J. Sulam, and M. Elad, "Working locally thinking globally: Theoretical guarantees for convolutional sparse coding," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5687–5701, Nov. 2017.

[25] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 3626–3633.

[26] K. Li, L. Gan, and C. Ling, "Convolutional compressed sensing using deterministic sequences," *IEEE Trans. Signal Process.*, vol. 61, no. 3, pp. 740–752, Feb. 2013.

[27] H. Zhang and V. M. Patel, "Convolutional sparse and low-rank coding-based image decomposition," *IEEE Trans. Image Process.*, vol. 27, no. 5, pp. 2121–2133, May 2018.

[28] V. Papyan, Y. Romano, J. Sulam, and M. Elad, "Convolutional dictionary learning via local processing," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 5306–5314.

[29] F. Heide, W. Heidrich, and G. Wetzstein, "Fast and flexible convolutional sparse coding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5135–5143.

[30] B. Choudhury, R. Swanson, F. Heide, G. Wetzstein, and W. Heidrich, "Consensus convolutional sparse coding," in *Proc. IEEE Int. Conf. Comput. Vis*, 2017, pp. 4290–4298.

[31] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCun, "Unsupervised learning of sparse features for scalable audio classification," in *Proc. 12th Int. Soc. Music Inf. Retrieval Conf.*, 2011, pp. 681–686.

[32] A. D. Szlam, K. Gregor, and Y. L. Cun, "Structured sparse coding via lateral inhibition," in *Proc. Conf. Adv. Neural Inf. Process. Syst.*, 2011, pp. 1116–1124.

[33] B. Wohlberg, "Efficient algorithms for convolutional sparse representations," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 301–315, Jan. 2016.

[34] J. Liu, C. Garcia-Cardona, B. Wohlberg, and W. Yin, "Online convolutional dictionary learning," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, (Beijing, China), pp. 1707–1711, Sep. 2017, doi: 10.1109/ICIP.2017.8296573.

[35] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 2528–2535.

[36] A. Szlam, K. Kavukcuoglu, and Y. LeCun, "Convolutional matching pursuit and dictionary training," 2010, arXiv:1010.0422, to be published.

[37] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. L. Cun, "Learning convolutional feature hierarchies for visual recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1090–1098.

[38] Y. He, K. Kavukcuoglu, Y. Wang, A. Szlam, and Y. Qi, "Unsupervised feature learning by deep sparse coding," in *Proc. SIAM Int. Conf. Data Mining*, 2014, pp. 902–910.

[39] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[40] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 806–814.

[41] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *J. Fourier Anal. Appl.*, vol. 14, pp. 629–654, Sep. 2008.

[42] L. Le Magoarou and R. Gribonval, "Flexible multilayer sparse approximations of matrices and applications," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 4, pp. 688–700, Jun. 2016.

[43] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1553–1564, Mar. 2010.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[45] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
[46] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2230–2249, May 2009.
[47] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, 1st ed. New York, NY, USA: Springer, 2010.
[48] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.

Jeremias Sulam (M'14) received the Bioengineering degree from the Concepcin del Uruguay, Argentina, in 2013, and the Ph.D. degree from the Department of Computer Science, Technion—Israel Institute of Technology, Haifa, Israel, in 2018, where he is currently a Postdoctoral Fellow. His research interests include signal and image processing, sparse representation modeling, inverse problems and machine learning, and their application to biomedical problems. He is the recipient of the Best Graduates award of the the Argentinean National Academy of Engineering.

Vardan Papyan received the B.Sc. and Ph.D. degrees from the Department of Computer Science, Technion—Israel Institute of Technology, Haifa, Israel, in 2013 and 2017, respectively. He is currently a Postdoctoral Fellow with the Department of Statistics Department, Stanford University, Stanford, CA, USA. His research interests include signal and image processing, sparsity-based modeling of signals, and in particular deep learning and its relation to sparsity.

Yaniv Romano received the B.Sc., M.Sc., and Ph.D. degrees from the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa, Israel, in 2012, 2015, and 2017, respectively. He is currently a Postdoctoral Fellow with the Department of Statistics, Stanford University, Stanford, CA, USA. His research interests include signal and image modeling, inverse problems, sparse & redundant representations and machine learning. He was a recipient of the 2015 Zeff fellowship, the 2017 Andrew and Erna Finci Viterbi fellowship, the 2017 Irwin and Joan Jacobs fellowship, and the 2018-2019 Zuckerman Postdoctoral scholarship.

Michael Elad (F'12) received the B.Sc., M.Sc., and D.Sc. degrees from the Department of Electrical engineering, Technion—Israel Institute of Technology, Haifa, Israel, in 1986, 1988, and 1997, respectively. Since 2003, he has been a faculty member in the Department of Computer Science, Technion—Israel Institute of Technology, where since 2010, he is a Full Professor. He works in the field of signal and image processing, specializing in inverse problems, and sparse representations. He was the recipient of numerous teaching awards, the 2008 and 2015 Henri Taub Prizes for Academic Excellence, and the 2010 Hershel-Rich prize for innovation. He is a SIAM Fellow (2018). Since January 2016, He has been the Editor-in-Chief for *SIAM Journal on Imaging Sciences* since January 2016.