Multi-Layer Convolutional Sparse Modeling: Pursuit and Dictionary Learning

Jeremias Sulam[†], Vardan Papyan[†], Yaniv Romano[‡] and Michael Elad[†]

[†]Department of Computer Science, [‡]Department of Electrical Engineering
Technion – Israel Institute of Technology

Abstract

The recently proposed Multi-Layer 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 computation of the forward pass in a CNN is equivalent to a pursuit algorithm aiming to estimate the nested sparse representation vectors – or feature maps – 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: there are no pursuit algorithms that can serve this model exactly, nor are there conditions to guarantee a non-empty model. While one can easily obtain signals that *approximately* satisfy the ML-CSC constraints, it remains unclear how to simply sample from the model and, more importantly, how one can train the convolutional filters from real data.

In this work, 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 non-trivial 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 auto-encoders, and we analyze these connections in detail.

Keywords — Convolutional Sparse Coding, Multilayer Pursuit, Convolutional Neural Networks, Dictionary Learning, Sparse Convolutional Filters.

1 Introduction

Signal models, in a broad sense, have always been central to the development of new algorithms. 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, 4] to machine learning [5, 6, 7].

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 [8, 9]. 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 [10]. 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 [11] demonstrated how *scattering networks* represent shift invariant analysis operators that are robust to deformations (in a Lipschitz-continuous sense). The work in [12] showed that deep neural networks preserve the metric structure of the data, under Gaussian weights assumption. In [13], 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 [14], 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 [15] 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 [16], 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 this 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.

The work in [16] 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 model's 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?

Before proceeding, it is worth noting that 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 learning approaches, in particular to those leveraging different structures or constraints in the construction of such dictionary. A very partial list of these works include the Chasing Butterflies approach [17], fast transform learning [18], Trainlets [19], 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 [20], and in particular to the k-sparse [21] and winner-take-all versions [22].

In order to progressively answer the questions posed above, we will first review the ML-CSC model in detail in Section 2. We will then study how signals can be projected onto the model in

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

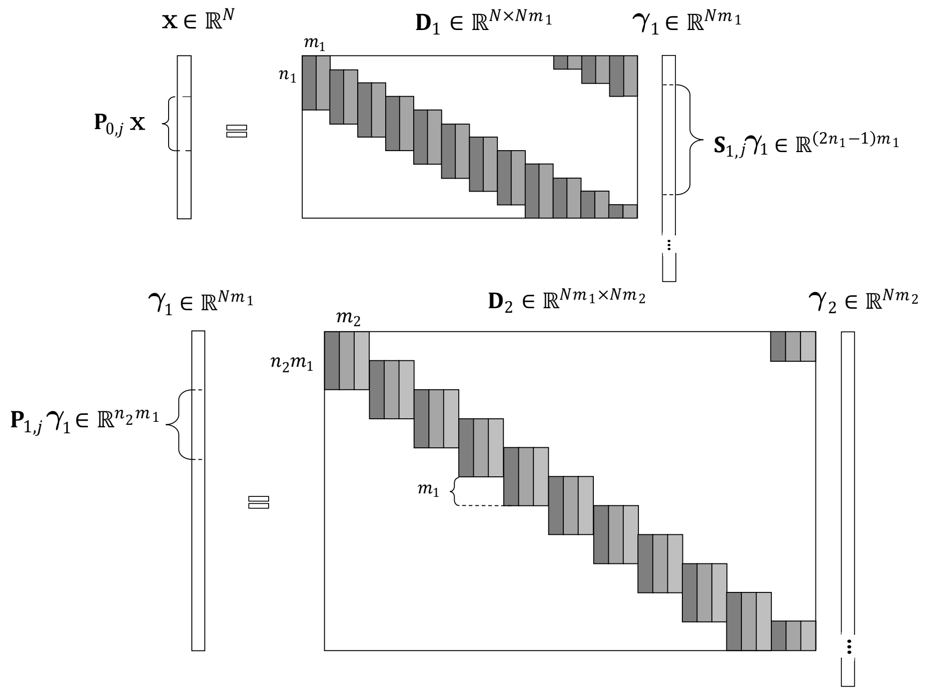
$\mathbf{x} \in \mathbb{R}^N$
$\mathbf{D}_1 \in \mathbb{R}^{N \times N m_1}$
$\boldsymbol{\gamma}_1 \in \mathbb{R}^{N m_1}$
m_1
n_1
$\mathbf{P}_{0,j}\,\mathbf{x}$
$=$
$\mathbf{S}_{1,j}\boldsymbol{\gamma}_1 \in \mathbb{R}^{(2n_1-1)m_1}$
$\boldsymbol{\gamma}_1 \in \mathbb{R}^{N m_1}$
$\mathbf{D}_2 \in \mathbb{R}^{N m_1 \times N m_2}$
$\boldsymbol{\gamma}_2 \in \mathbb{R}^{N m_2}$
m_2
$n_2 m_1$
$\mathbf{P}_{1,j}\boldsymbol{\gamma}_1 \in \mathbb{R}^{n_2 m_1}$
$=$
m_1

Figure 1: The CSC model (top), and its ML-CSC extension by imposing a similar model on $\boldsymbol{\gamma}_1$ (bottom). From a local perspective, a patch from the signal, $\mathbf{P}_{0,j}\mathbf{x}$ has a corresponding sparse stripe given by $\mathbf{S}_{1,j}\boldsymbol{\gamma}_1$. An analogous decomposition can be stated for a patch from the signal $\boldsymbol{\gamma}_1$, represented by $\mathbf{P}_{1,j}\boldsymbol{\gamma}_1$.

Section 3, 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 4, 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 5, before concluding in Section 6.

2 Background

2.1 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 $\boldsymbol{\gamma}_1 \in \mathbb{R}^{Nm_1}$ 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 (Figure 1 top). An immediate consequence of this model assumption is the fact that each j^{th} *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 j^{th} 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 [23]. 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 [23] 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.

2.2 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 Figure 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 $\|\boldsymbol{\gamma}\|^s_{0,\infty}$ and $\|\boldsymbol{\gamma}\|^p_{0,\infty}$ for stripes and patches, respectively. In addition, we will employ the $\ell_{2,\infty}$ norm version, naturally defined as $\|\boldsymbol{\gamma}\|^s_{2,\infty} = \max_i \|\mathbf{S}_i\boldsymbol{\gamma}\|_2$, 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 $\mathbf{x} \in \mathbb{R}^N$ admits a representation in terms of the ML-CSC model if

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

We will refer to the set of signals satisfying the ML-CSC model assumptions with parameter $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_L]$, as the set $\mathcal{M}_{\boldsymbol{\lambda}}$. In addition, when referring to a signal $\mathbf{x} \in \mathcal{M}_{\boldsymbol{\lambda}}$, we will often denote it by $\mathbf{x}(\boldsymbol{\gamma}_i)$ to emphasize its decomposition in terms of the nested representations $\{\boldsymbol{\gamma}_i\}_{i=1}^L$.

Note that $\mathbf{x} \in \mathcal{M}_{\boldsymbol{\lambda}}$ can also be expressed as $\mathbf{x} = \mathbf{D}_1\mathbf{D}_2 \dots \mathbf{D}_L\boldsymbol{\gamma}_L$. For the purpose of the following derivations, define $\mathbf{D}^{(i)}$ to be the *effective* dictionary at the i^{th} level, i.e., $\mathbf{D}^{(i)} = \mathbf{D}_1\mathbf{D}_2 \dots \mathbf{D}_i$. This way, one can concisely write

$$\mathbf{x} = \mathbf{D}^{(L)}\boldsymbol{\gamma}_L, \tag{2}$$

where $\mathbf{D}^{(L)}$ is the L-layers Convolutional Dictionary. Generally, we have that $\mathbf{x} = \mathbf{D}^{(i)}\boldsymbol{\gamma}_i,\ 1 \le i \le L$.

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:

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 $\mathbf{D}^{(i)} = \mathbf{D}_1\mathbf{D}_2 \dots \mathbf{D}_i$ 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.

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

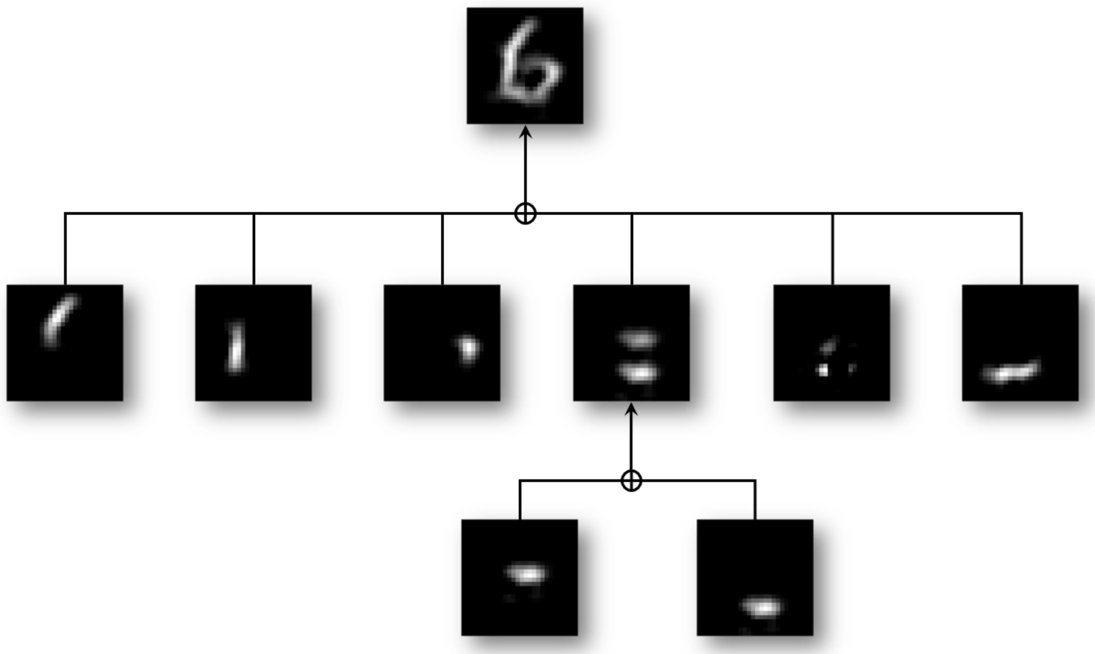

Figure 2: From atoms to molecules: Illustration of the ML-CSC model for a number 6. Two local convolutional atoms (bottom row) are combined to create slightly more complex structures – molecules – at the second level, which are then combined to create the global atom representing, in this case, a digit. Note that even though the atoms are local (with small support) and convolutional, we depict them in their respective locations within the global structure. Refer to the main body (Section 2.2) for a detailed description of this decomposition.

The proof of this lemma is rather straight forward, and we include it in Appendix A. Note that what was denoted as the effective dimension at the i^{th} 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 Figure 2 for a 3^{rd}-layer convolutional atom of the digit "6". While the 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

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

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 Figure 2. Likewise, we now focus on the fourth of such atoms, $\mathbf{d}_{j_4}^{(2)} = \mathbf{D}_1\mathbf{d}_{2,j_4}$. In this particular case, $\|\mathbf{d}_{2,j_4}\|_0 = 2$. Indicating these two non-zeros elements by i_1 and i_2, we can express:

$$\mathbf{d}_{j_4}^{(2)} = \mathbf{d}_{i_1}^{(1)} d_{2,j_1}^{i_1} + \mathbf{d}_{i_2}^{(1)} d_{2,j_1}^{i_2}. \tag{4}$$

These two atoms from $\mathbf{D}_1$ are precisely those appearing in the bottom of the decomposition.

2.3 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 $\mathcal{E}_0$. This pursuit problem – searching for sparse convolutional features under the ML-CSC model – can be formulated in a number of different ways depending on the model deviations assumed at each layer. In its most general form, this pursuit is represented by the Deep Coding Problem (DCP$_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$), as introduced in [16]:

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 $\boldsymbol{\mathcal{E}}$, the deep coding problem DCP$_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ is defined as:

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

where the scalars λ_i and $\mathcal{E}_i$ are the i^{th} 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

$$\mu(\mathbf{D}) = \max_{i\neq j} |\mathbf{d}_i^T\mathbf{d}_j|. \tag{5}$$

Relying on this measure, Theorem 5 in [16] 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

$$\|\boldsymbol{\gamma}_i\|_{0,\infty}^s < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{D}_i)}\right), \tag{6}$$

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

$$\|\boldsymbol{\gamma}_i - \hat{\boldsymbol{\gamma}}_i\|_2^2 \leq 4{\mathcal{E}_0}^2 \prod_{j=1}^{i} \frac{4^{i-1}}{1 - (2\|\boldsymbol{\gamma}_j\|_{0,\infty}^s - 1)\mu(\mathbf{D}_j)}. \tag{7}$$

In the particular instance of the $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ where $\mathcal{E}_i = 0$ for $1 \le i \le L-1$, 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 [16].

Two remarks should be noted about the above stability result:

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 $\hat{\mathbf{x}} \notin \mathcal{M}_{\boldsymbol{\lambda}}$.

3 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:

$$(\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}): \quad \min_{\{\boldsymbol{\gamma}_i\}_{i=1}^L} \quad \|\mathbf{y} - \mathbf{x}(\boldsymbol{\gamma}_i)\|_2 \quad \text{s.t.} \quad \mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{M}_{\boldsymbol{\lambda}}. \tag{8}$$

Note that this problem differs from the $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ 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 $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$. Put differently, the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem can be considered as a special case of the $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ where model deviations are allowed only at the outer-most level. From this perspective, the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ is an instance of the $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ for which $\mathcal{E}_i = 0$ for $i \ge 1$. Recall that the theoretical analysis of the $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ 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.

3.1 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 $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ or $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem. In light of the above discussion and the known stability result for the $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ 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 $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem:*
Suppose $\mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{M}_{\boldsymbol{\lambda}}$ is observed through $\mathbf{y} = \mathbf{x} + \mathbf{v}$, where $\mathbf{v}$ is a bounded noise vector, $\|\mathbf{v}\|_2 \le \mathcal{E}_0$, and $\|\boldsymbol{\gamma}_i\|_{0,\infty}^s = \lambda_i < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{D}^{(i)})}\right)$, for $1 \le i \le L$. Consider the set $\{\hat{\boldsymbol{\gamma}}_i\}_{i=1}^L$ to be the solution of the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem. Then,

$$\|\boldsymbol{\gamma}_i - \hat{\boldsymbol{\gamma}}_i\|_2^2 \le \frac{4\mathcal{E}_0^2}{1 - (2\|\boldsymbol{\gamma}_i\|_{0,\infty}^s - 1)\mu(\mathbf{D}^{(i)})} \tag{9}$$

Before presenting the proof of this claim, we note a few remarks on this result:

1. The obtained bound for every layer depends only on the sparsity of the representation and on the mutual coherence of the effective dictionary for that layer, $\mathbf{D}^{(i)}$. This allows us to provide a bound which is not cumulative across the layers – it does not grow with the depth of the network.

2. Unlike the stability result for the $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ 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. On the other hand, the deeper layers correspond to higher abstractions levels, and the corresponding representations are indeed expected to be sparser.

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 [16].

We now prove the stability result.

Proof. Denote the solution to the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem by $\hat{\mathbf{x}}$; i.e., $\hat{\mathbf{x}} = \mathbf{D}^{(i)}\hat{\boldsymbol{\gamma}}_i$. Given that the original signal $\mathbf{x}$ satisfies $\|\mathbf{y}-\mathbf{x}\|_2 \le \mathcal{E}_0$, the solution to the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem, $\hat{\mathbf{x}}$ must satisfy

$$\|\mathbf{y}-\hat{\mathbf{x}}\|_2 \le \|\mathbf{y}-\mathbf{x}\|_2 \le \mathcal{E}_0, \tag{10}$$

as this is the signal which provides the shortest ℓ_2 (data-fidelity) distance from $\mathbf{y}$. Note that because $\hat{\mathbf{x}}(\boldsymbol{\gamma}_i) \in \mathcal{M}_{\boldsymbol{\lambda}}$, we can have that $\hat{\mathbf{x}} = \mathbf{D}^{(i)}\hat{\boldsymbol{\gamma}}_i,\ \forall\ 1 \le i \le L$. Recalling Lemma 1, the product $\mathbf{D}_1\mathbf{D}_2\ldots\mathbf{D}_i$ is a convolutional dictionary. In addition, we have required that $\|\hat{\boldsymbol{\gamma}}_i\|_{0,\infty}^{s} \le \lambda_i < \frac{1}{2}\left(1+\frac{1}{\mu(\mathbf{D}^{(i)})}\right)$. Therefore, from the same arguments presented in [23], it follows that

$$\|\boldsymbol{\gamma}_i-\hat{\boldsymbol{\gamma}}_i\|_2^2 \le \frac{4\mathcal{E}_0^2}{1-(2\|\boldsymbol{\gamma}_i\|_{0,\infty}^{s}-1)\mu(\mathbf{D}^{(i)})}. \tag{11}$$

□

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 $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ problem. This result, however, provides bounds that are generally looser than the one presented in the above theorem, and so we defer this to Appendix B.

3.2 Pursuit Algorithms

Both the $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ and $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problems seek for underlying representations $\hat{\boldsymbol{\gamma}}_i$ which explain – under different assumptions – the measurements $\mathbf{y}$. The next natural question is how and to what accuracy one can retrieve the solutions of those respective problems. In other words, how does one solve these problems *in practice*?

As shown in [16], one can approximate the solution to the $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ 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, and it provides an approximate solution of this problem. Better alternatives were also proposed, such as the Layered BP algorithm, 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 $\text{DCP}_{\boldsymbol{\lambda}}^{\boldsymbol{\mathcal{E}}}$ problem, naturally, 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 $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ 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

Algorithm 1: ML-CSC Pursuit

```
Input: $\mathbf{y}, \{\mathbf{D}_i\}, k$;
$\hat{\gamma}_L \leftarrow$ Pursuit$(\mathbf{y}, \mathbf{D}^{(L)}, k)$;
for $j = L, \ldots, 1$ do
    $\hat{\gamma}_{j-1} \leftarrow \mathbf{D}_j \hat{\gamma}_j$
return $\{\hat{\gamma}_i\}$;
```

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 a stable solution.

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:

$$\min_{\gamma} \|\mathbf{y} - \mathbf{D}^{(L)}\gamma\|_2^2 \text{ s.t. } \|\gamma\|_{0,\infty}^s \leq k. \tag{12}$$

In a setting where a signal $\mathbf{x}(\gamma_i) \in \mathcal{P}_{\mathcal{M}_\lambda}$ has been corrupted with noise of known energy $\mathcal{E}_0^2$, one could reformulate this problem by analogously minimizing over the $\ell_{0,\infty}$ norm of γ subject to the constraint $\|\mathbf{y} - \mathbf{D}^{(L)}\gamma\|_2^2 \leq \mathcal{E}_0^2$. We employ the formulation in (12), however, as this preserves the structure of our projection formulation.

Once the deepest representation has been estimated, we proceed by obtaining the remaining ones by simply applying their definition, thus assuring that $\hat{\mathbf{x}} = \mathbf{D}^{(i)}\hat{\gamma}_i \in \mathcal{M}_\lambda$. 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 $\hat{\gamma}_i$. 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.

3.3 Stability Guarantees for Pursuit Algorithms

Given a signal $\mathbf{y} = \mathbf{x}(\gamma_i) + \mathbf{v}$, and the respective solution of the ML-CSC Pursuit in Algorithm 1, how close will the estimated $\hat{\gamma}_i$ be to the original representations γ_i? These bounds will clearly depend on the specific Pursuit algorithm employed to obtain $\hat{\gamma}_L$. 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 presenting these results, however, we shall need to state 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 $\mathcal{T} = Supp(\gamma)$, 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 i^{th} *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 γ with support $\mathcal{T}$ satisfies the the N.V.S property for a given dictionary $\mathbf{D}$ if

$$\|\mathbf{D}\gamma\|_0 = \|\mathbf{D}_\mathcal{T}\|_\infty^0. \tag{13}$$

Intuitively, the above property implies that the entries in γ 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. Because our derivations will follow a worse-case and deterministic analysis, however, we will need this property to formulate recovery guarantees for pursuit algorithms. 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 almost surely.

A direct consequence of the above property is that of maximal cardinality of representations. If $\boldsymbol{\gamma}$ satisfies the 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 [24]. As defined in [23], a convolutional dictionary $\mathbf{D}$ satisfies the Stripe-RIP condition with constant δ_k if, for every $\boldsymbol{\gamma}$ such that $\|\boldsymbol{\gamma}\|_{0,\infty}^s = k$,

$$(1-\delta_k)\|\boldsymbol{\gamma}\|_2^2 \leq \|\mathbf{D}\boldsymbol{\gamma}\|_2^2 \leq (1+\delta_k)\|\boldsymbol{\gamma}\|_2^2. \tag{14}$$

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 Appendix C.

Lemma 2. Local one-sided near isometry property:
If $\mathbf{D}$ is a convolutional dictionary satisfying the Stripe-RIP condition in (14) with constant δ_k, then

$$\|\mathbf{D}\boldsymbol{\gamma}\|_{2,\infty}^{2,p} \leq (1+\delta_k)\ \|\boldsymbol{\gamma}\|_{2,\infty}^{2,s}. \tag{15}$$

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). On the other hand, this property states nothing unexpected: if the CSC model is fully described by a shift-invariant local model, then its properties should be able to be characterized in a local manner as well. Lastly, 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. The respective proof is more involved, however, and a matter of current work.

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 $\|\boldsymbol{\gamma}_i\|_{0,\infty}^s = \lambda_i < \frac{1}{2}\left(1+\frac{1}{\mu(\mathbf{D}_i)}\right)$, for $1 \leq i \leq L$ and $\|\boldsymbol{\gamma}_L\|_{0,\infty}^s = \lambda_L \leq \frac{1}{3}\left(1+\frac{1}{\mu(\mathbf{D}^{(L)})}\right)$. Consider solving the Pursuit stage in Algorithm 1 as*

$$\hat{\boldsymbol{\gamma}}_L = \arg\min_{\boldsymbol{\gamma}} \|\mathbf{y} + \mathbf{D}^{(L)}\boldsymbol{\gamma}\|\|_2^2 + \zeta_L\|\boldsymbol{\gamma}\|_1, \tag{16}$$

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, for every $1 \leq i \leq L$ layer,

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

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 is a coefficient that depends on the ratio between the local dimensions of the layers, $c_j = \left\lceil \frac{2n_{j-1}-1}{n_j} \right\rceil$.

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 $\epsilon_0 = \|\mathbf{v}\|^{\mathrm{P}}_{2,\infty}$. Assume that all representations $\boldsymbol{\gamma}_i$ satisfy the N.V.S. property for the respective dictionaries $\mathbf{D}_i$, with $\|\boldsymbol{\gamma}_i\|^s_{0,\infty} = \lambda_i < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{D}_i)}\right)$, for $1 \leq i \leq L$, and*

$$\|\boldsymbol{\gamma}_L\|^s_{0,\infty} < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{D}^{(L)})}\right) - \frac{1}{\mu(\mathbf{D}^{(L)})} \cdot \frac{\epsilon_0}{|\gamma_L^{min}|}, \tag{17}$$

where γ_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

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

The proofs of both Theorems 6 and 7 are included in Appendix D.1 and D.2, 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 bound on the representation error.

These results show the flavor of theoretical claims that can be obtained for the proposed ML-CSC Pursuit Algorithm. By employing similar derivations to those detailed in the respective proofs, one could – in principle – provide recovery claims for other versions of this method, by employing other sparse coding strategies.

3.4 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 $\mathbf{y}$ 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 $\|\hat{\boldsymbol{\gamma}}_L\|^s_{0,\infty} \leq \lambda_L$. Yet, nothing guarantees that $\|\hat{\boldsymbol{\gamma}}_i\|^s_{0,\infty} \leq \lambda_i$ 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 $\|\hat{\boldsymbol{\gamma}}_L\|^s_{0,\infty}$, 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.

This algorithm can be shown to be optimal under certain assumptions, and we now provide a sketch of the proof of this claim. Consider the first iteration of the above method, where $k = 1$. If OMP succeeds in providing the closest $\hat{\boldsymbol{\gamma}}_L$ subject to the respective constraint, i.e. providing the solution to

$$\min_{\boldsymbol{\gamma}} \|\mathbf{y} - \mathbf{D}^{(L)}\boldsymbol{\gamma}\|^2_2 \text{ s.t. } \|\boldsymbol{\gamma}\|^s_{0,\infty} \leq 1, \tag{18}$$

and if $\|\hat{\boldsymbol{\gamma}}_i\|^s_{0,\infty} \leq \lambda_i$ for every i, then this solution effectively provides the closest signal to $\mathbf{y}$ in the model defined by $\boldsymbol{\lambda} = [\lambda_1, \dots, 1]$. If $\lambda_L = 1$, we are done. Otherwise, if $\lambda_L > 1$, we might increase the number of non-zeros in $\hat{\boldsymbol{\gamma}}_L$, while decreasing the ℓ_2 distance to $\mathbf{y}$. This is done by continuing

[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.

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 $\|\hat{\boldsymbol{\gamma}}_i\|^s_{0,\infty} > \lambda_i$ for any $1 \leq i < L$ then
        break;
    else
        $\mathbf{x}^* \leftarrow \mathbf{D}^{(i)} \hat{\boldsymbol{\gamma}}_i$;
return $\mathbf{x}^*$
```

to the next iteration: running again OMP with the constraint $\|\hat{\boldsymbol{\gamma}}_L\|^s_{0,\infty} \leq 2$, and obtaining the respective $\hat{\boldsymbol{\gamma}}_i$.

At any k^{th} iteration, due to the nature of the OMP algorithm, $Supp(\hat{\boldsymbol{\gamma}}_L^{k-1}) \subseteq Supp(\hat{\boldsymbol{\gamma}}_L^k)$. If all estimates $\hat{\boldsymbol{\gamma}}_i$ satisfy the N.V.S. property for the respective dictionaries $\mathbf{D}_i$, then the sparsity of each $\hat{\boldsymbol{\gamma}}_i$ is non-decreasing through the iterations, $\|\hat{\boldsymbol{\gamma}}_i^{k-1}\|^s_{0,\infty} \leq \|\hat{\boldsymbol{\gamma}}_i^k\|^s_{0,\infty}$. For this reason, if an estimate $\hat{\boldsymbol{\gamma}}_L^k$ is obtained such that any of the corresponding $\ell_{0,\infty}$ constraints is violated, then necessarily one constraint will be violated at the next (or any future) iteration. Therefore, the closest signal in the model will be the one corresponding to the iteration before one of the constraints was violated.

As a final comment on this subject, while Algorithms 1 and 2 were presented separately, they are indeed related and one could envision combining them into a single method. The distinction between them was motivated by making the derivations of our theoretical analysis 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.

3.5 Summary - Pursuit for the ML-CSC

Before proceeding, 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.

We now move to the next major difficulty with the ML-CSC model: studying the convolutional filters and the need to learn its parameters.

4 Learning the model

4.1 Preliminaries

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 $\hat{\gamma}_L$ with Algorithm 1 removing all other assumptions on the dictionaries $\mathbf{D}_i$. In this case, nothing will prevent $\hat{\gamma}_{L-1} = \mathbf{D}_L\hat{\gamma}_L$ from being dense. While one could argue that this is an artifact of the presented algorithm (for instance, for not explicitly enforcing both representations to be sparse), nothing 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 $\|\boldsymbol{\gamma}_L\|_{0,\infty}^s \leq 2$, 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, $\Pr\left(\gamma_{L-1}^j = 0\right) = 0$. 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.

4.2 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 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 $\boldsymbol{\gamma}_L : \|\boldsymbol{\gamma}_L\|_{0,\infty}^s \leq \lambda_L$ and constants $c_i = \left\lceil \frac{2n_{i-1}-1}{n_i} \right\rceil$, the signal $\mathbf{x} = \mathbf{D}^{(L)}\boldsymbol{\gamma}_L \in \mathcal{M}_{\boldsymbol{\lambda}}$ if

$$\|\mathbf{D}_i\|_0 \leq \frac{\lambda_{i-1}}{\lambda_i c_i}, \quad \forall\ 1 < i \leq L. \tag{19}$$

The simple proof of this Lemma is included in Appendix E. 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?

4.3 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:

$$\min_{\{\boldsymbol{\gamma}_i^k\},\{\mathbf{D}_i\}} \quad \sum_{k=1}^K \|\mathbf{y}^k - \mathbf{x}^k(\boldsymbol{\gamma}_i^k, \mathbf{D}_i)\|_2^2 \quad \text{s.t.} \begin{cases} \mathbf{x}^k \in \mathcal{M}_{\boldsymbol{\lambda}}, \\ \|\mathbf{d}_i^j\|_2 = 1, \forall\ i, j \end{cases} \tag{20}$$

We have included the constraint of every dictionary atom, of every level, 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. To begin with, 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 arise 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,

$$\|\boldsymbol{\gamma}_{L-1}\|_{0,\infty}^s \leq c_L \|\mathbf{D}_L\|_0 \|\boldsymbol{\gamma}_L\|_{0,\infty}^s, \tag{21}$$

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

$$\|\boldsymbol{\gamma}_i\|_{0,\infty}^s \;\leq\; c \prod_{j=i+1}^{L} \|\mathbf{D}_j\|_0 \|\boldsymbol{\gamma}_L\|_{0,\infty}^s. \tag{22}$$

In this way, minimizing the sparsity of any i^{th} 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 (20) into the following Multi-Layer Convolutional Dictionary Learning Problem:

$$\min_{\{\boldsymbol{\gamma}_L^k\},\{\mathbf{D}_i\}} \sum_{k=1}^{K} \|\mathbf{y}^k - \mathbf{D}_1\mathbf{D}_2\dots\mathbf{D}_L\boldsymbol{\gamma}_L^k\|_2^2 + \sum_{i=2}^{L} \zeta_i \|\mathbf{D}_i\|_0 \quad \text{s.t.} \begin{cases} \|\boldsymbol{\gamma}_L^k\|_{0,\infty}^s \leq \lambda_L, \\ \|\mathbf{d}_i^j\|_2 = 1, \forall\, i,j \end{cases} \tag{23}$$

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 3, 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:

$$\min_{\{\boldsymbol{\gamma}_L^k\},\{\mathbf{D}_i\}} \sum_{k=1}^{K} \|\mathbf{y}^k - \mathbf{D}_1\mathbf{D}_2\dots\mathbf{D}_L\boldsymbol{\gamma}_L^k\|_2^2 + \iota \sum_{i=1}^{L} \|\mathbf{D}_i\|_F^2 + \sum_{i=2}^{L} \zeta_i \|\mathbf{D}_i\|_0 + \lambda \|\boldsymbol{\gamma}_L^k\|_1. \tag{24}$$

The problem in Equation (24) 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 (8), which was analyzed in detail in the previous sections, and its solution will be approximated through iterative shrinkage algorithms. In particular, we employ an efficient implementation of the FISTA algorithm [25]. 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, common in the deep learning community. In addition, these operators are expected to be very efficient to apply 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 [26].

[5] From [16], 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 $c_L = \left\lceil \frac{2n_{L-1}-1}{n_L} \right\rceil$, we can obtain a bound to $\|\boldsymbol{\gamma}_{L-1}\|_{0,\infty}^s$.

Given the obtained γ_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 (24) by applying T iterations of projected gradient descent. This is done by computing the gradient of the ℓ_2 terms in Problem (24) (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 [27]. 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.

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: $\gamma_L \leftarrow \arg\min_{\gamma} \|\mathbf{y}_k - \mathbf{D}^{(L)}\gamma\|_2 + \lambda\|\gamma\|_1$ ;
    Update Dictonaries:
    for $i = L, \ldots, 1$ do
        for $t = 1, \ldots, T$ 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 $t = 1, \ldots, T$ do
        $\mathbf{D}_1^{t+1} \leftarrow \mathbf{D}_1^t - \eta\nabla f(\mathbf{D}_1^t)$ ;
```

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.

As a final comment, note this approach can also be employed to minimize Problem (23) 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 [27]. 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.

4.4 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 work [17], and our resulting algorithm is very similar of the PALM method employed by the authors, initially proposed in [28]. However, their approach is designed for general (not convolutional) multi-level dictionaries, and the algorithm is particularly targeted to lower semicontinuous functions. The inspiring work of in [18], 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. Finally, as our proposed approach effectively learns a sparse dictionary, we share some similarities with the double-sparsity work from [29]. In particular, in its Trainlets version [19], the authors proposed to learn a dictionary as a sparse combination of cropped wavelets atoms. From the previous comment on the work from [18], this could also potentially be expressed as a product of sparse convolutional atoms.

What is the connection between this learning formulation and that of deep convolutional networks? Recalling the analysis presented in [16], 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 $\hat{\gamma}_L$ in our setting is solved with such an algorithm, then the problem in (24) *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 [20], 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 [21], where the hidden layer is constrained to having exactly k non-zeros. In practice, this boils down to a simple $k-$hard thresholding step of the hidden activation, and the neuron 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 [22] 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 [26] 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.

5 Experiments

We now provide experimental results to demonstrate several aspects of the ML-CSC model. As a case-study, we consider the MNIST dataset [30]. 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.

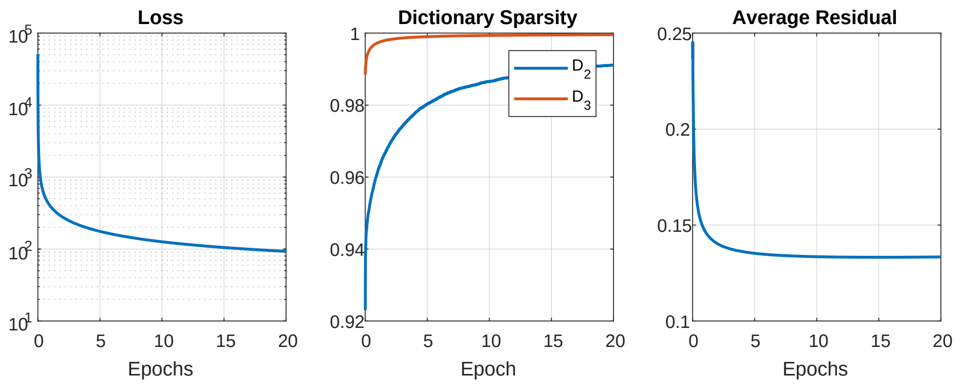

Figure 3: Evolution of the Loss function, sparsity of the convolutional dictionaries and average residual norm during training on the MNIST dataset.

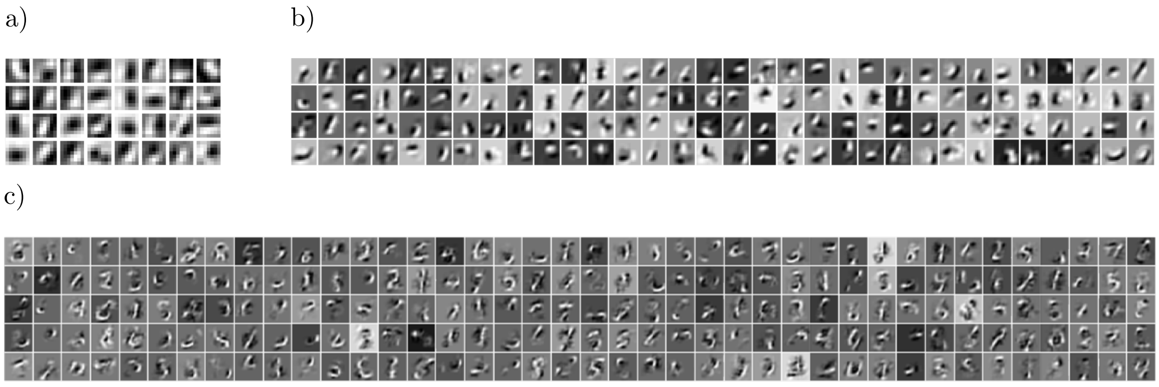

Figure 4: ML-CSC model trained on the MNIST dataset. a) The local filters of the dictionary $\mathbf{D}_1$. b) The local filters of the effective dictionary $\mathbf{D}^{(2)} = \mathbf{D}_1\mathbf{D}_2$. c) Some of the 1024 local atoms of the effective dictionary $\mathbf{D}^{(3)}$ which, because of the dimensions of the filters and the strides, are global atoms of size 28×28.

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 the FISTA [25] algorithm, 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[6].

We depict the evolution of the Loss function during training in Figure 3, as well as the sparsity of the second and third dictionaries and the average residual norm. The resulting model is depicted in Figure 4. 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 $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$. In Figure 5, we depict the multi-layer features $\boldsymbol{\gamma}_i$, $i = 1, 2, 3$, obtained with the 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

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

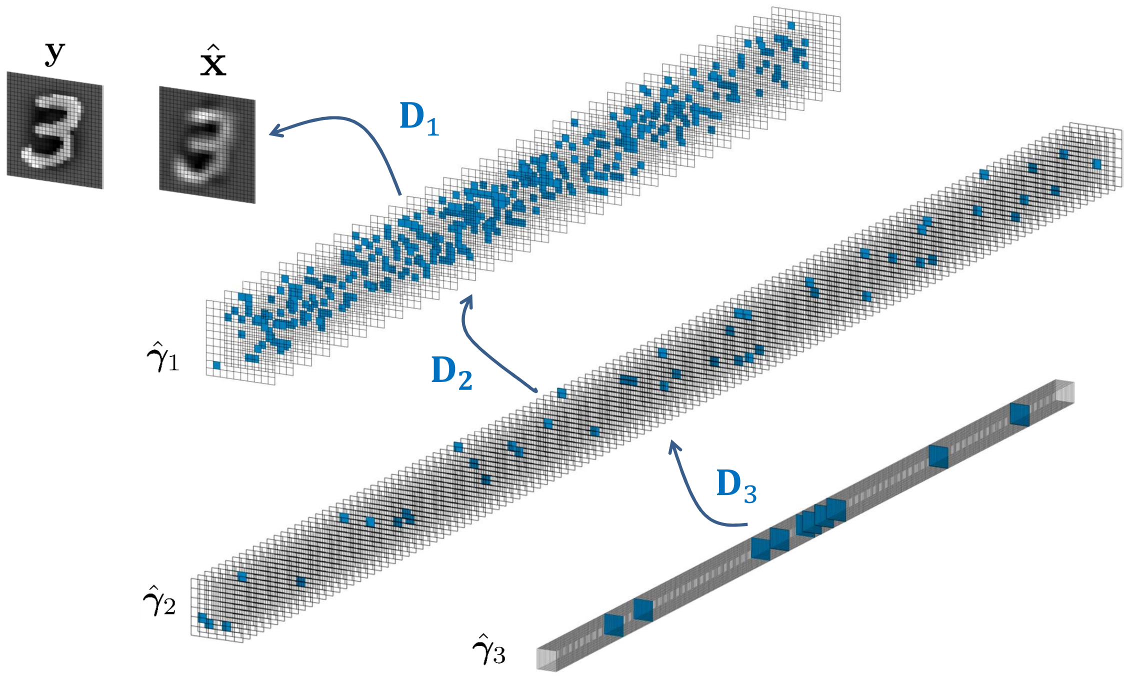

Figure 5: Decompositions of an image from MNIST in terms of its nested sparse features γ_i and multi-layer convolutional dictionaries $\mathbf{D}_i$.

scales, resulting in an approximation $\hat{\mathbf{x}}$. Naturally, the quality of the approximation can be improved by increasing the cardinality of the representations.

5.1 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 [16]. 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" [7] 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 γ_L, with a target sample sparsity k and normally distributed coefficients. We generate the signals as $\mathbf{x} = \mathbf{D}^{(i)}\gamma_i$, and then corrupt them with Gaussian noise ($\sigma = 0.02$) obtaining the measurements $\mathbf{y} = \mathbf{x}(\gamma_i) + \mathbf{v}$.

In order to evaluate our projection approach, we run Algorithm 1 employing the Subspace Pursuit algorithm [31] for the sparse coding step, with the oracle target cardinality k. Recall that once the deepest representations $\hat{\gamma}_L$ have been obtained, the inner ones are simply computed as $\hat{\gamma}_{i-1} = \mathbf{D}_i\hat{\gamma}_i$. In the layered approach from [16], on the other hand, the pursuit of the representations progresses sequentially: first running a pursuit for $\hat{\gamma}_1$, then employing this estimate to run another pursuit for $\hat{\gamma}_2$, 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 Figure 6: at the top we depict the relative ℓ_2 error of the recovered representations ($\|\hat{\gamma}_i - \gamma_i\|_2/\|\gamma_i\|_2$) and, at the bottom,

[7]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.

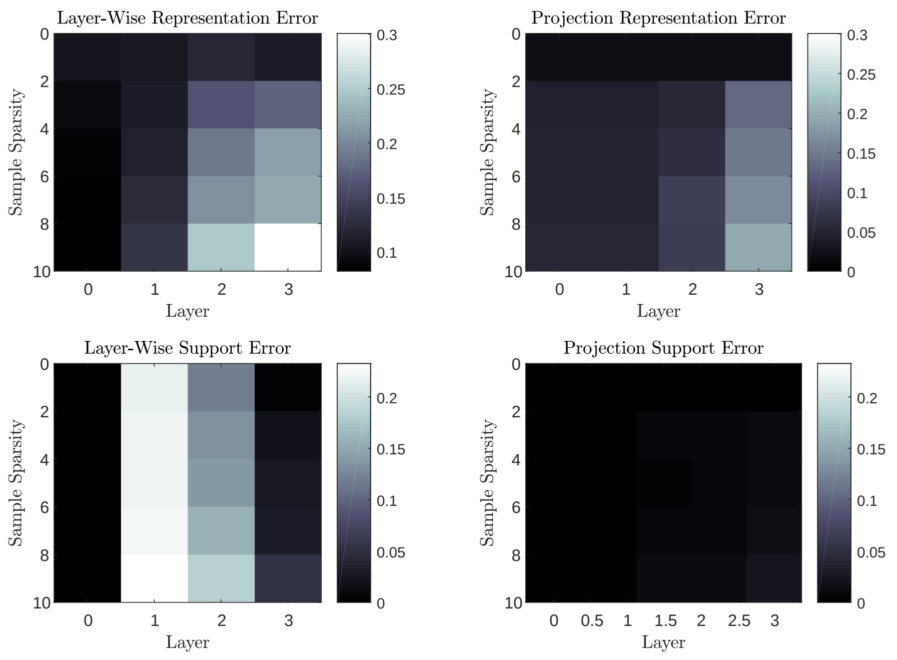

Figure 6: Recovery of representations from noisy synthetic signals. Top: normalized ℓ_2 error between the estimated and the true representations. Bottom: normalized intersection between the estimated and the true support of the representations.

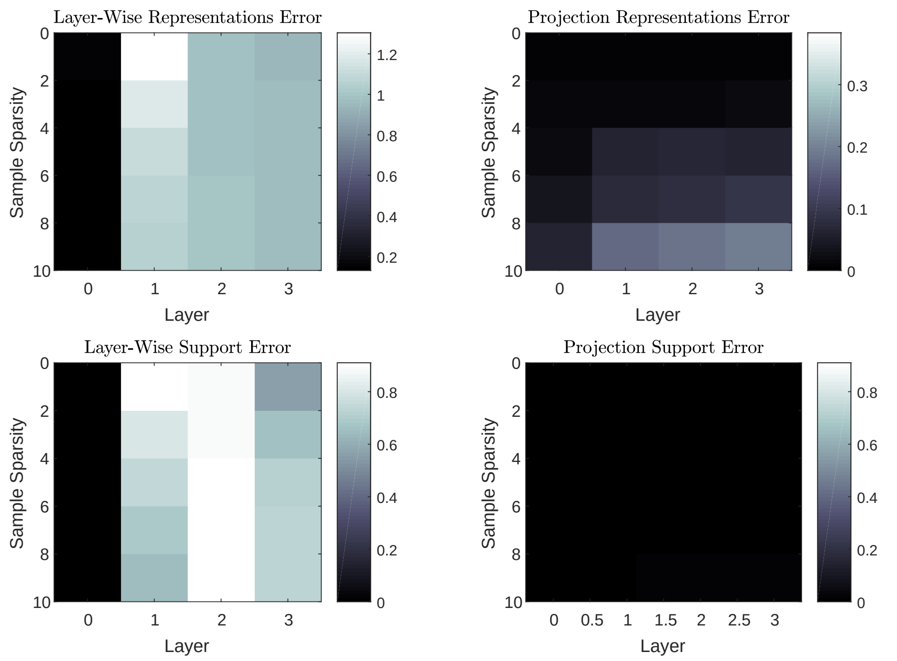

Figure 7: Recovery of representations from noisy MNIST digits. Top: normalized ℓ_2 error between the estimated and the true representations. Bottom: normalized intersection between the estimated and the true support of the representations.

the normalized intersection of the supports [32], 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 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 3.3, 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 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 $\sigma = 0.02$, providing nothing but noisy digits. We then repeat both pursuit approaches seeking to estimate the underlying representations, obtaining the results reported in Figure 7.

Clearly, this represents a significantly more challenging scenario for the layered approach, which recovers only a small 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 $\hat{\boldsymbol{\gamma}}_2$.

5.2 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 Figure 8. 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 [20] and k-sparse autoencoders [21]. In particular, the work in [20] 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 [22], 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 trivially employed for comparison in this problem.

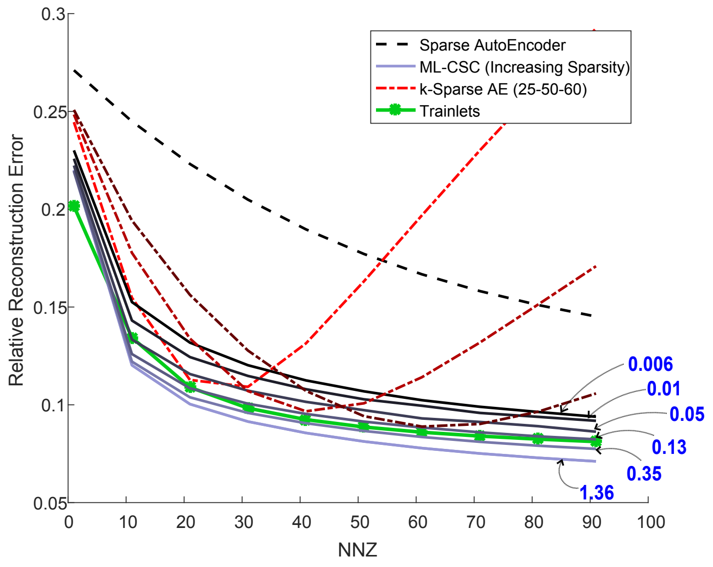

Figure 8: M-term approximation as a function of non-zero coefficients (NNZ) for MNIST digits, comparing sparse autoencoders [20], k-sparse autoencoders [21], trainlets (OSDL) [19], and the proposed ML-CSC for models with different filter sparsity levels. The relative number of parameters is depicted in NavyBlue.

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 $\hat{\gamma}_3$ 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 [19]. 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.

5.3 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. 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

Method	**Classification Error**
Stacked Denoising Autoencoder (3 layers) [33]	1.28%
k-Sparse Autoencoder (1K units) [21]	1.35%
Shallow WTA Autoencoder (2K units) [22]	1.20%
Stacked WTA Autoencoder (2K units)[22]	1.11%
ML-CSC (1K units) - 2nd Layer Rep.	1.30%
ML-CSC (2K units) - 2nd&3rd Layer Rep.	1.15%

Table 1: Unsupervised classification results on MNIST.

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 1, reporting the classification error on the 10K testing samples.

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. 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 [22] achieve a classification error of 0.48, which provide 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.

6 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. In particular, we employed the dictionary sparsity as a proxi for the sparsity of the inner representations, which effectively yields a model consisting of cascade of sparse convolutional filters. We demonstrated the proposed approach by learning the model on the MNIST dataset, and studied several practical applications. The experimental results show 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.

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 Review.*, vol. 51, pp. 34–81, Feb. 2009.

[2] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *IEEE Proceedings - Special Issue on Applications of Sparse Representation & Compressive Sensing*, vol. 98, no. 6, pp. 1045–1057, 2010.

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

[4] J. Mairal, F. Bach, and G. Sapiro, "Non-local Sparse Models for Image Restoration," *IEEE International Conference on Computer Vision.*, vol. 2, pp. 2272–2279, 2009.

[5] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent k-svd: Learning a discriminative dictionary for recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2651–2664, 2013.

[6] V. M. Patel, Y.-C. Chen, R. Chellappa, and P. J. Phillips, "Dictionaries for image and video-based face recognition," *Journal of the Optimal Society of America*, vol. 31, no. 5, pp. 1090–1103, 2014.

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

[8] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*, pp. 396–404, 1990.

[9] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, "Learning representations by back-propagating errors," *Cognitive Modeling*, vol. 5, no. 3, p. 1, 1988.

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

[11] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.

[12] R. Giryes, G. Sapiro, and A. M. Bronstein, "Deep neural networks with random gaussian weights: A universal classification strategy?" *IEEE Transactions on Signal Processing*, vol. 64, no. 13, pp. 3444–3457, 2016.

[13] N. Cohen, O. Sharir, and A. Shashua, "On the expressive power of deep learning: A tensor analysis," in *29th Annual Conference on Learning Theory* (V. Feldman, A. Rakhlin, and O. Shamir, eds.), vol. 49 of *Proceedings of Machine Learning Research*, (Columbia University, New York, New York, USA), pp. 698–728, PMLR, 23–26 Jun 2016.

[14] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 399–406, 2010.

[15] B. Xin, Y. Wang, W. Gao, D. Wipf, and B. Wang, "Maximal sparsity with deep networks?," in *Advances in Neural Information Processing Systems*, pp. 4340–4348, 2016.

[16] V. Papyan, Y. Romano, and M. Elad, "Convolutional neural networks analyzed via convolutional sparse coding," *To appear in JMLR. arXiv preprint arXiv:1607.08194*, 2016.

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

[18] O. Chabiron, F. Malgouyres, J. Tourneret, and N. Dobigeon, “Toward Fast Transform Learning,” *International Journal of Computer Vision*, pp. 1–28, 2015.

[19] J. Sulam, B. Ophir, M. Zibulevsky, and M. Elad, “Trainlets: Dictionary learning in high dimensions,” *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3180–3193, 2016.

[20] A. Ng, “Sparse autoencoder,” *CS294A Lecture Notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[21] A. Makhzani and B. Frey, “K-sparse autoencoders,” *arXiv preprint arXiv:1312.5663*, 2013.

[22] A. Makhzani and B. J. Frey, “Winner-take-all autoencoders,” in *Advances in Neural Information Processing Systems*, pp. 2791–2799, 2015.

[23] V. Papyan, J. Sulam, and M. Elad, “Working locally thinking globally: Theoretical guarantees for convolutional sparse coding,” *To appear in IEEE Transactions of Signal Processing. Preprint arXiv:1607.02009*, 2017.

[24] E. J. Candes and T. Tao, “Decoding by linear programming,” *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[25] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[26] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, “Sparse convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 806–814, 2015.

[27] T. Blumensath and M. E. Davies, “Iterative Thresholding for Sparse Approximations,” *Journal of Fourier Analysis and Applications*, vol. 14, pp. 629–654, Sept. 2008.

[28] J. Bolte, S. Sabach, and M. Teboulle, “Proximal alternating linearized minimization for nonconvex and nonsmooth problems,” *Mathematical Programming*, vol. 146, no. 1-2, pp. 459–494, 2014.

[29] R. Rubinstein, M. Zibulevsky, and M. Elad, “Double Sparsity: Learning Sparse Dictionaries for Sparse Signal Approximation,” *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1553–1564, 2010.

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[31] W. Dai and O. Milenkovic, “Subspace pursuit for compressive sensing signal reconstruction,” *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

[32] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing.* Springer Publishing Company, Incorporated, 1st ed., 2010.

[33] 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,” *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

A Properties of the ML-CSC model

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.

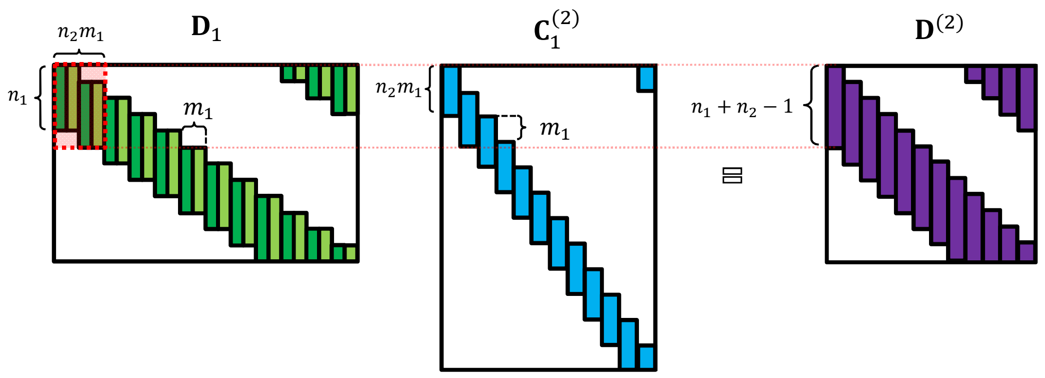

Figure 9: Illustration of a convolutional dictionary $\mathbf{D}_1$ multiplied by one of the circulat matrices from $\mathbf{D}_2$, in this case $\mathbf{C}_1^{(2)}$.

Proof. A convolutional dictionary is formally defined as the concatenation of banded circulant matrices. Consider $\mathbf{D}_1 = \left[\mathbf{C}_1^{(1)}, \mathbf{C}_2^{(1)}, \dots, \mathbf{C}_{m_1}^{(1)}\right]$, where each circulant $\mathbf{C}_i^{(1)} \in \mathbb{R}^{N \times N}$. Likewise, one can express $\mathbf{D}_2 = \left[\mathbf{C}_1^{(2)}, \mathbf{C}_2^{(2)}, \dots, \mathbf{C}_{m_2}^{(2)}\right]$, where $\mathbf{C}_i^{(2)} \in \mathbb{R}^{Nm_1 \times N}$. Then,

$$\mathbf{D}^{(2)} = \mathbf{D}_1\mathbf{D}_2 = \left[\mathbf{D}_1\mathbf{C}_1^{(2)}, \mathbf{D}_1\mathbf{C}_2^{(2)}, \dots, \mathbf{D}_1\mathbf{C}_{m_2}^{(2)}\right]. \tag{25}$$

Each term $\mathbf{D}_1\mathbf{C}_i^{(2)}$ is the product of a concatenation of banded circulant matrices and a banded circulant matrix. Because the atoms in each $\mathbf{C}_i^{(2)}$ have a stride of m_1 (the number of filters in $\mathbf{D}_1$) each of these products is in itself a banded circulant matrix. This is illustrated in Figure 9, where it becomes clear that the first atom in $\mathbf{C}_1^{(2)}$ (of length $n_2 m_1$) linearly combines atoms from the first n_2 blocks of m_1 filters in $\mathbf{D}_1$ (in this case $n_2 = 2$). These block are simply the unique set of filters shifted at every position. The second column in $\mathbf{C}_1^{(2)}$ will do the same for the next set n_2 blocks, starting from the second one, etc.

From the above discussion, $\mathbf{D}_1\mathbf{C}_1^{(2)}$ results in a banded circulant matrix of dimension $N \times N$. In particular, the band of this matrix is given by the dimension of the filters in the first dictionary (n_1) plus the number of blocks combined by $\mathbf{C}_1^{(2)}$ minus one. In other words, the effective dimension of the filters in $\mathbf{D}_1\mathbf{C}_1^{(2)}$ is given by $n_2^{\text{eff}} = n_2 + n_1 - 1$.

The effective dictionary $\mathbf{D}^{(2)} = \mathbf{D}_1\mathbf{D}_2$ is simply a concatenation of m_2 such banded circulant matrices. In other words, $\mathbf{D}^{(2)}$ is a convolutional dictionary with filters of dimension n_2^{eff}. The same analysis can be done for the effective dictionary at every layer, $\mathbf{D}^{(i)}$, resulting in an effective dimension of $n_i^{\text{eff}} = n_i + n_{i-1}^{\text{eff}} - 1$, and so $n_L^{\text{eff}} = \sum_{i=1}^{L} n_i - (L-1)$.

Finally, note that $\mathbf{D}^{(i)}$ has Nm_i columns, and thus there will be m_i local filters in the effective CSC model. $\square$

B Another stability result for the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem

Theorem 8. *(Stability of the solution to the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem):*
Suppose $\mathbf{x}(\boldsymbol{\gamma}_i) \in \mathcal{M}_{\boldsymbol{\lambda}}$ 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 [8] $\|\boldsymbol{\gamma}_i\|_{0,\infty}^{s} = \lambda_i < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{D}_i)}\right)$, *for $1 \leq i \leq L$. Consider the set $\{\hat{\boldsymbol{\gamma}}_i\}_{i=1}^{L}$ to be the solution of*

[8]The assumption that $\|\boldsymbol{\gamma}_i\|_{0,\infty}^{s} = \lambda_i$ can be relaxed to $\|\boldsymbol{\gamma}_i\|_{0,\infty}^{s} \leq \lambda_i$, with slight modifications of the result.

the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem. If $\|\boldsymbol{\gamma}_L\|_{0,\infty}^s < \frac{1}{2}\left(1+\frac{1}{\mu(\mathbf{D}^{(L)})}\right)$ then

$$\|\boldsymbol{\gamma}_i - \hat{\boldsymbol{\gamma}}_i\|_2^2 \leq \frac{4\mathcal{E}_0^2}{1-(2\|\boldsymbol{\gamma}_L\|_{0,\infty}^s - 1)\mu(\mathbf{D}^{(L)})} \prod_{j=i+1}^{L} \left[1 + (2\|\boldsymbol{\gamma}_j\|_{0,\infty}^s - 1)\mu(\mathbf{D}_j)\right]. \tag{26}$$

Proof. Given that the original signal $\mathbf{x}$ satisfies $\|\mathbf{y}-\mathbf{x}\|_2 \leq \mathcal{E}_0$, the solution to the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem, $\hat{\mathbf{x}}$ must satisfy

$$\|\mathbf{y}-\hat{\mathbf{x}}\|_2 \leq \|\mathbf{y}-\mathbf{x}\|_2 \leq \mathcal{E}_0, \tag{27}$$

as this is the signal which provides a lowest ℓ_2 (data-fidelity) term. In addition, $\|\hat{\boldsymbol{\gamma}}_L\|_{0,\infty}^s = \lambda_L < \frac{1}{2}(1+\frac{1}{\mu(\mathbf{D}^{(L)})})$. Therefore, from the same arguments presented in [23], it follows that

$$\|\boldsymbol{\gamma}_L - \hat{\boldsymbol{\gamma}}_L\|_2^2 \leq \frac{4\mathcal{E}_0^2}{1-(2\|\boldsymbol{\gamma}_L\|_{0,\infty}^s - 1)\mu(\mathbf{D}^{(L)})} = \mathcal{E}_L^2. \tag{28}$$

Because the solution $\hat{\mathbf{x}}(\{\hat{\boldsymbol{\gamma}}_i\}) \in \mathcal{M}_{\boldsymbol{\lambda}}$, then $\hat{\boldsymbol{\gamma}}_{L-1} = \mathbf{D}_L\hat{\boldsymbol{\gamma}}_L$. Therefore

$$\|\boldsymbol{\gamma}_{L-1} - \hat{\boldsymbol{\gamma}}_{L-1}\|_2^2 = \|\mathbf{D}_L(\boldsymbol{\gamma}_L - \hat{\boldsymbol{\gamma}}_L)\|_2^2 \leq (1+\delta_{2k})\|\boldsymbol{\gamma}_L - \hat{\boldsymbol{\gamma}}_L\|_2^2, \tag{29}$$

where δ_{2k} is the S-RIP of $\mathbf{D}_L$ with constant $2k = 2\|\boldsymbol{\gamma}_L\|_{0,\infty}^s$. This follows from the triangle inequality of the $\ell_{0,\infty}$ norm and the fact that, because $\hat{\boldsymbol{\gamma}}_L$ is a solution to the $\mathcal{P}_{\mathcal{M}_{\boldsymbol{\lambda}}}$ problem, $\|\hat{\boldsymbol{\gamma}}_L\|_{0,\infty}^s \leq \lambda_L = \|\boldsymbol{\gamma}_L\|_{0,\infty}^s$. The S-RIP can in turn be bounded with the mutual coherence [23] as $\delta_k \leq (k-1)\mu(\mathbf{D}_L)$, from which one obtains

$$\|\boldsymbol{\gamma}_{L-1} - \hat{\boldsymbol{\gamma}}_{L-1}\|_2^2 \leq \mathcal{E}_L^2\ (1+(2\|\boldsymbol{\gamma}_L\|_{0,\infty}^s - 1)\mu(\mathbf{D}_L)). \tag{30}$$

From similar arguments, extending this to an arbitrary i^{th} layer,

$$\|\boldsymbol{\gamma}_i - \hat{\boldsymbol{\gamma}}_i\|_2^2 \leq \mathcal{E}_L^2 \prod_{j=i+1}^{L} (1+(2\|\boldsymbol{\gamma}_j\|_{0,\infty}^s - 1)\mu(\mathbf{D}_j)). \tag{31}$$

□

For the sake of simplicity, one can relax the above bounds further obtaining that, subject to the assumptions in Theorem 8,

$$\|\boldsymbol{\gamma}_i - \hat{\boldsymbol{\gamma}}_i\|_2^2 \leq \mathcal{E}_L^2\ 2^{(L-i)}. \tag{32}$$

This follows simply by employing the fact that $\|\boldsymbol{\gamma}_i\|_{0,\infty}^s < \frac{1}{2}\left(1+\frac{1}{\mu(\mathbf{D}_i)}\right)$.

C Local stability of the S-RIP

Lemma 2. Local one-sided near isometry:
If $\mathbf{D}$ is a convolutional dictionary satisfying the Stripe-RIP condition with constant δ_k, then

$$\|\mathbf{D}\boldsymbol{\gamma}\|_{2,\infty}^{2,p} \leq (1+\delta_k)\ \|\boldsymbol{\gamma}\|_{2,\infty}^{2,s} \tag{33}$$

Proof. Consider the patch-extraction operator $\mathbf{P}_i$ from the signal $\mathbf{x} = \mathbf{D}\boldsymbol{\gamma}$, and $\mathbf{S}_i$ the operator that extracts the corresponding stripe from $\boldsymbol{\gamma}$ such that $\mathbf{P}_i\mathbf{x} = \boldsymbol{\Omega}\mathbf{S}_i\boldsymbol{\gamma}$, where $\boldsymbol{\Omega}$ is a local stripe dictionary [23]. Denote the i^{th} stripe by $\mathbf{s}_i = \mathbf{S}_i\boldsymbol{\gamma}$. Furthermore, denote by $\bar{\mathbf{S}}_i$ the operator that *extracts the support* of $\mathbf{s}_i$ from $\boldsymbol{\gamma}$. Clearly, $\mathbf{x} = \mathbf{D}\bar{\mathbf{S}}_i^T\bar{\mathbf{S}}_i\boldsymbol{\gamma}$. Note that $\|\mathbf{P}_i\|_2 = \|\mathbf{S}_i\|_2 = 1$. Then,

$$\begin{aligned}
\|\mathbf{D}\boldsymbol{\gamma}\|_{2,\infty}^{p} &= \max_i \|\mathbf{P}_i\mathbf{D}\bar{\mathbf{S}}_i^T\bar{\mathbf{S}}_i\boldsymbol{\gamma}\|_2 && (34)\\
&\leq \max_i \|\mathbf{P}_i\|_2\ \|\mathbf{D}\bar{\mathbf{S}}_i^T\bar{\mathbf{S}}_i\boldsymbol{\gamma}\|_2 && (35)\\
&\leq \max_i \|\mathbf{D}\bar{\mathbf{S}}_i^T\|_2\|\bar{\mathbf{S}}_i\boldsymbol{\gamma}\|_2 && (36)\\
&\leq \max_i \|\mathbf{D}\bar{\mathbf{S}}_i^T\|_2\ \max_j \|\bar{\mathbf{S}}_j\boldsymbol{\gamma}\|_2. && (37)
\end{aligned}$$

Note that

$$\max_j \|\bar{\mathbf{S}}_j \boldsymbol{\gamma}\|_2 = \max_j \|\mathbf{S}_j \boldsymbol{\gamma}\|_2 = \|\boldsymbol{\gamma}\|^s_{2,\infty}, \tag{38}$$

as the non-zero entries in $\bar{\mathbf{S}}_j\boldsymbol{\gamma}$ and $\mathbf{S}_j\boldsymbol{\gamma}$ are the same. On the other hand, denoting by $\lambda_{max}(\cdot)$ the maximal eigenvalue of the matrix in its argument, $\|\mathbf{D}\bar{\mathbf{S}}_i^T\|_2 = \sqrt{\lambda_{max}\left(\bar{\mathbf{S}}_i\mathbf{D}^T\mathbf{D}\bar{\mathbf{S}}_i^T\right)}$, and if $\mathcal{T} = Supp(\boldsymbol{\gamma})$,

$$\lambda_{max}\left(\bar{\mathbf{S}}_i\mathbf{D}^T\mathbf{D}\bar{\mathbf{S}}_i^T\right) \leq \lambda_{max}\left(\mathbf{D}_{\mathcal{T}}^T\mathbf{D}_{\mathcal{T}}\right), \tag{39}$$

because[9] $\bar{\mathbf{S}}_i\mathbf{D}^T\mathbf{D}\bar{\mathbf{S}}_i^T$ is a principal sub-matrix of $\mathbf{D}_{\mathcal{T}}^T\mathbf{D}_{\mathcal{T}}$. Thus, also $\|\mathbf{D}\bar{\mathbf{S}}_i^T\|_2 \leq \|\mathbf{D}_{\mathcal{T}}\|_2$.

The Stripe-RIP condition, as in Equation (14), provides a bound on the square of the singular values of $\mathbf{D}_{\mathcal{T}}$. Indeed, $\|\mathbf{D}_{\mathcal{T}}\|_2^2 \leq (1+\delta_k)$, for every $\mathcal{T} : \|\mathcal{T}\|^s_{0,\infty} = k$. Including these in the above one obtains the desired claim:

$$\|\mathbf{D}\boldsymbol{\gamma}\|^p_{2,\infty} \leq \max_i \|\mathbf{D}\bar{\mathbf{S}}_i^T\|_2 \max_j \|\bar{\mathbf{S}}_j\boldsymbol{\gamma}\|_2 \leq \sqrt{1+\delta_k}\|\boldsymbol{\gamma}\|^s_{2,\infty}. \tag{40}$$

□

D Recovery guarantees for pursuit algorithms

D.1 Convex relaxation case

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 $\mathbf{y} = \mathbf{x} + \mathbf{v}$, $\|\mathbf{v}\|^p_{2,\infty} \leq \epsilon_0$. Assume that all representations $\boldsymbol{\gamma}_i$ satisfy the N.V.S. property for the respective dictionaries $\mathbf{D}_i$, and that $\|\boldsymbol{\gamma}_i\|^s_{0,\infty} = \lambda_i < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{D}_i)}\right)$, for $1 \leq i \leq L$ and $\|\boldsymbol{\gamma}_L\|^s_{0,\infty} = \lambda_L \leq \frac{1}{3}\left(1 + \frac{1}{\mu(\mathbf{D}^{(L)})}\right)$. Consider solving the Pursuit stage in Algorithm 1 as*

$$\hat{\boldsymbol{\gamma}}_L = \arg\min_{\boldsymbol{\gamma}} \|\mathbf{y} + \mathbf{D}^{(L)}\boldsymbol{\gamma}\|_2^2 + \zeta_L\|\boldsymbol{\gamma}\|_1, \tag{41}$$

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, for every $1 \leq i \leq L$ layer,

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

where $\epsilon_L = \frac{15}{2}\,\epsilon_0\sqrt{\|\boldsymbol{\gamma}_j\|^p_{0,\infty}}$ is the error at the last layer, and c_j is a coefficient that depends on the ratio between the local dimensions of the layers, $c_j = \left\lceil \frac{2n_{j-1}-1}{n_j} \right\rceil$.

Proof. Denote $\boldsymbol{\Delta}_i = \hat{\boldsymbol{\gamma}}_i - \boldsymbol{\gamma}_i$. From [23] (Theorem 19), the solution $\hat{\boldsymbol{\gamma}}_L$ will satisfy:

i) $\mathcal{S}(\hat{\boldsymbol{\gamma}}_L) \subseteq \mathcal{S}(\boldsymbol{\gamma}_L)$; and

ii) $\|\boldsymbol{\Delta}_L\|_\infty \leq \frac{15}{2}\,\epsilon_0$.

As shown in [16], given the ℓ_∞ bound of the representation error, we can bound its $\ell_{2,\infty}$ norm as well, obtaining

$$\|\boldsymbol{\Delta}_L\|^p_{2,\infty} \leq \|\boldsymbol{\Delta}_L\|_\infty\sqrt{\|\boldsymbol{\Delta}_L\|^p_{0,\infty}} \leq \frac{15}{2}\,\epsilon_0\sqrt{\|\boldsymbol{\gamma}_L\|^p_{0,\infty}}, \tag{42}$$

because, since $\mathcal{S}(\hat{\boldsymbol{\gamma}}_L) \subset \mathcal{S}(\boldsymbol{\gamma}_L)$, $\|\boldsymbol{\Delta}_L\|^s_{0,\infty} \leq \|\boldsymbol{\gamma}_L\|^s_{0,\infty}$. Define $\epsilon_L = \frac{15}{2}\,\epsilon_0\sqrt{\|\boldsymbol{\gamma}_L\|^p_{0,\infty}}$.

[9]The inequality in (39) can be shown by considering the equivalent expression $\lambda_{max}\left(\mathbf{S}_i\mathbf{D}_{\mathcal{T}}^T\mathbf{D}_{\mathcal{T}}\mathbf{S}_i^T\right)$, where the matrix $\mathbf{D}_{\mathcal{T}}^T\mathbf{D}_{\mathcal{T}}$ is real and symmetric, and the matrix $\mathbf{S}_i$ is semi-orthogonal; i.e. $\mathbf{S}_i\mathbf{S}_i^T = \mathbf{I}$. Thus, from Poincaré Separation Theorem, $\lambda_{min}\left(\mathbf{D}_{\mathcal{T}}^T\mathbf{D}_{\mathcal{T}}\right) \leq \lambda\left(\mathbf{S}_i\mathbf{D}_{\mathcal{T}}^T\mathbf{D}_{\tau}\mathbf{S}_i^T\right) \leq \lambda_{max}\left(\mathbf{D}_{\mathcal{T}}^T\mathbf{D}_{\mathcal{T}}\right)$.

Recall that the N.V.S. property states that the entries in $\boldsymbol{\gamma}$ will no cause the support of the atoms in $\mathbf{D}$ cancel each other; i.e., $\|\mathbf{D}\boldsymbol{\gamma}\|_0 = \|\mathbf{D}_\mathcal{T}\|_\infty^0$ (Definition 5). In other words, this provides a bound on the cardinality of the vector resulting from the multiplication of $\mathbf{D}$ with any sparse vector with support $\mathcal{T}$. Concretely, if $\boldsymbol{\gamma}$ satisfies the N.V.S., then $\|\mathbf{D}\boldsymbol{\gamma}\|_0 \geq \|\mathbf{D}\hat{\boldsymbol{\gamma}}\|_0$.

Consider now the estimate at the $L-1$ layer, obtained as $\hat{\boldsymbol{\gamma}}_{L-1} = \mathbf{D}_L\hat{\boldsymbol{\gamma}}_L$. Because $\boldsymbol{\gamma}_L$ satisfies the N.V.S. property, and $\mathcal{S}(\hat{\boldsymbol{\gamma}}_L) \subseteq \mathcal{S}(\boldsymbol{\gamma}_L)$, then $\|\hat{\boldsymbol{\gamma}}_{L-1}\|_0 \leq \|\boldsymbol{\gamma}_{L-1}\|_0$, and more so $\mathcal{S}(\hat{\boldsymbol{\gamma}}_{L-1}) \subseteq \mathcal{S}(\boldsymbol{\gamma}_{L-1})$.

On the other hand, recalling Lemma 2 and denoting by δ_{λ_L} the Stripe-RIP constant of the $\mathbf{D}_L$ dictionary, and because $\|\boldsymbol{\Delta}_L\|_{0,\infty}^s \leq \|\boldsymbol{\gamma}_L\|_{0,\infty}^s \leq \lambda_L$,

$$\|\boldsymbol{\Delta}_{L-1}\|_{2,\infty}^{2,p} = \|\mathbf{D}_L\boldsymbol{\Delta}_L\|_{2,\infty}^{2,p} \leq (1+\delta_{\lambda_L})\|\boldsymbol{\Delta}_L\|_{2,\infty}^{2,s}. \tag{43}$$

Notice that by employing the above Lemma, we have bounded the **patch-wise** $\ell_{2,\infty}$ norm of $\boldsymbol{\Delta}_{L-1}$ in terms of the **stripe-wise** $\ell_{2,\infty}$ of $\boldsymbol{\Delta}_L$. Recalling the derivation from [16] (Section 7.1), at each i^{th} layer, a stripe includes up to $(2n_{i-1}-1)/n_i$ patches. Define $c_i = \left\lceil \frac{2n_{i-1}-1}{n_i} \right\rceil$. From this, one can bound the square of the ℓ_2 norm of a stripe with the norm of the maximal patch within it - this is true for every stripe, and in particular for the stripe with the maximal norm. This implies that $\|\boldsymbol{\Delta}_L\|_{2,\infty}^{2,s} \leq c_L\|\boldsymbol{\Delta}_L\|_{2,\infty}^{2,p}$. Then,

$$\|\boldsymbol{\Delta}_{L-1}\|_{2,\infty}^{2,p} \leq (1+\delta_k)\|\boldsymbol{\Delta}_L\|_{2,\infty}^{2,s} \leq (1+\delta_{\lambda_L})c_L\|\boldsymbol{\Delta}_L\|_{2,\infty}^{2,p}. \tag{44}$$

Employing the result in Eq. (42),

$$\|\boldsymbol{\Delta}_{L-1}\|_{2,\infty}^{2,p} \leq (1+\delta_k)c_L\|\boldsymbol{\Delta}_L\|_{2,\infty}^{2,p} \leq (1+\delta_k)\ c_L\ \epsilon_L^2. \tag{45}$$

We can further bound the Stripe-RIP constant by $\delta_k \leq (k-1)\mu(\mathbf{D})$ [23], obtaining

$$\|\boldsymbol{\Delta}_{L-1}\|_{2,\infty}^{2,p} \leq (1+(\|\boldsymbol{\gamma}_L\|_{0,\infty}^s-1)\mu(\mathbf{D}_L))\ \epsilon_L^2\ c_L. \tag{46}$$

Iterating this analysis for the remaining layers yields

$$\|\hat{\boldsymbol{\gamma}}_i - \boldsymbol{\gamma}_i\|_{2,\infty}^{2,p} \leq \epsilon_L^2 \prod_{j=i+1}^{L} c_j\ (1+(\|\boldsymbol{\gamma}_j\|_{0,\infty}^s-1)\mu(\mathbf{D}_j)). \tag{47}$$

This general result can be relaxed for the sake of simplicity. Indeed, considering that $\|\boldsymbol{\gamma}_i\|_{0,\infty}^s < \frac{1}{2}\left(1+\frac{1}{\mu(\mathbf{D}_i)}\right)$, for $1 \leq i \leq L$,

$$1+(\|\boldsymbol{\gamma}_j\|_{0,\infty}^s-1)\mu(\mathbf{D}_j) < 3/2, \tag{48}$$

and so

$$\|\hat{\boldsymbol{\gamma}}_i - \boldsymbol{\gamma}_i\|_{2,\infty}^{p} \leq \epsilon_L \prod_{j=i+1}^{L} \sqrt{\frac{3c_j}{2}} \tag{49}$$

□

D.2 Greedy case

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 $\|\boldsymbol{\gamma}_i\|_{0,\infty}^s = \lambda_i < \frac{1}{2}\left(1+\frac{1}{\mu(\mathbf{D}_i)}\right)$, for $1 \leq i \leq L$, and

$$\|\boldsymbol{\gamma}_L\|_{0,\infty}^s < \frac{1}{2}\left(1+\frac{1}{\mu(\mathbf{D}^{(L)})}\right) - \frac{1}{\mu(\mathbf{D}^{(L)})}\cdot\frac{\epsilon_0}{|\gamma_L^{min}|}, \tag{50}$$

where γ_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

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

Proof. Given that $\boldsymbol{\gamma}_L$ satisfies Equation (50), from [23] (Theorem 17) one obtains that

$$\|\hat{\boldsymbol{\gamma}}_L - \boldsymbol{\gamma}_L\|_2^2 \leq \frac{\mathcal{E}_0^2}{1-\mu(\mathbf{D}^{(L)})(\|\boldsymbol{\gamma}_L\|_{0,\infty}^s - 1)}. \tag{51}$$

Moreover, if the OMP algorithm is run for $\|\boldsymbol{\gamma}_L\|_0$ iterations, then all the non-zero entries are recovered, i.e., *Supp*$(\hat{\boldsymbol{\gamma}}_L) =$ *Supp*$(\boldsymbol{\gamma}_L)$. Therefore, $\|\hat{\boldsymbol{\gamma}}_L - \boldsymbol{\gamma}_L\|_{0,\infty}^s \leq \|\boldsymbol{\gamma}_L\|_{0,\infty}^s = \lambda_L$.

Now, let $\hat{\boldsymbol{\gamma}}_{L-1} = \mathbf{D}_L \hat{\boldsymbol{\gamma}}_L$. Regarding the support of $\hat{\boldsymbol{\gamma}}_{L-1}$, because $\boldsymbol{\gamma}_L$ satisfies the N.V.S. property, $\|\hat{\boldsymbol{\gamma}}_{L-1}\|_0 \leq \|\boldsymbol{\gamma}_{L-1}\|_0$. More so, all entries in $\hat{\boldsymbol{\gamma}}_{L-1}$ will correspond to non-zero entries in $\boldsymbol{\gamma}_{L-1}$. In other words,

$$\textit{Supp}(\hat{\boldsymbol{\gamma}}_{L-1}) \subseteq \textit{Supp}(\boldsymbol{\gamma}_{L-1}). \tag{52}$$

Consider now the error at the $L-1$ layer, $\|\boldsymbol{\gamma}_{L-1} - \hat{\boldsymbol{\gamma}}_{L-1}\|_2^2$. Since $\|\boldsymbol{\gamma}_{L-1} - \hat{\boldsymbol{\gamma}}_{L-1}\|_{0,\infty}^s \leq \|\boldsymbol{\gamma}_{L-1}\|_{0,\infty}^s$, we can bound this error in terms of the Stripe RIP:

$$\|\boldsymbol{\gamma}_{L-1} - \hat{\boldsymbol{\gamma}}_{L-1}\|_2^2 = \|\mathbf{D}_L(\boldsymbol{\gamma}_L - \hat{\boldsymbol{\gamma}}_L)\|_2^2 \leq (1+\delta_{\lambda_L})\|\boldsymbol{\gamma}_L - \hat{\boldsymbol{\gamma}}_L\|_2^2, \tag{53}$$

We can further bound the SRIP constant as $\delta_k \leq (k-1)\mu(\mathbf{D})$, from which one obtains

$$\|\hat{\boldsymbol{\gamma}}_{L-1} - \boldsymbol{\gamma}_{L-1}\|_2^2 \leq \frac{\mathcal{E}_0^2}{1-\mu(\mathbf{D}^{(L)})(\|\boldsymbol{\gamma}_L\|_{0,\infty}^s - 1)}(1 + (\|\boldsymbol{\gamma}_L\|_{0,\infty}^s - 1)\mu(\mathbf{D}_L)). \tag{54}$$

From similar arguments, one obtains analogous claims for any i^{th} layer; i.e.,

$$\|\hat{\boldsymbol{\gamma}}_i - \boldsymbol{\gamma}_i\|_2^2 \leq \frac{\mathcal{E}_0^2}{1-\mu(\mathbf{D}^{(L)})(\|\boldsymbol{\gamma}_L\|_{0,\infty}^s - 1)} \prod_{j=i+1}^{L} (1 + (\|\boldsymbol{\gamma}_j\|_{0,\infty}^s - 1)\mu(\mathbf{D}_j)). \tag{55}$$

This bound can be further relaxed for the sake of simplicity. Because $\|\boldsymbol{\gamma}_i\|_{0,\infty}^s < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{D}_i)}\right)$, for $1 \leq i \leq L$, then $(1 + (\|\boldsymbol{\gamma}_L\|_{0,\infty}^s - 1)\mu(\mathbf{D}_L)) < 3/2$, and so

$$\|\hat{\boldsymbol{\gamma}}_i - \boldsymbol{\gamma}_i\|_2^2 \leq \frac{\mathcal{E}_0^2}{1-\mu(\mathbf{D}^{(L)})(\|\boldsymbol{\gamma}_L\|_{0,\infty}^s - 1)} \left(\frac{3}{2}\right)^{L-i}. \tag{56}$$

□

E Sparse Dictionaries

Lemma 3. Dictionary Sparsity Condition
Consider the ML-CSC model $\mathcal{M}_{\boldsymbol{\lambda}}$ described by the the dictionaries $\{\mathbf{D}_1\}_{i=1}^L$ and the layer-wise $\ell_{0,\infty}$-sparsity levels $\lambda_1, \lambda_2, \ldots, \lambda_L$. Given $\boldsymbol{\gamma}_L : \|\boldsymbol{\gamma}_L\|_{0,\infty}^s \leq \lambda_L$ and constants $c_i = \left\lceil \frac{2n_{i-1}-1}{n_i} \right\rceil$, the signal $\mathbf{x} = \mathbf{D}^{(L)}\boldsymbol{\gamma}_L \in \mathcal{M}_{\boldsymbol{\lambda}}$ if

$$\|\mathbf{D}_i\|_0 \leq \frac{\lambda_{i-1}}{\lambda_i c_i}, \quad \forall\, 1 < i \leq L. \tag{57}$$

Proof. This lemma can be proven simply by considering that the patch-wise $\ell_{0,\infty}$ of the representation $\boldsymbol{\gamma}_{L-1}$ can be bounded by $\|\boldsymbol{\gamma}_{L-1}\|_{0,\infty}^p \leq \|\mathbf{D}_L\|_0 \|\boldsymbol{\gamma}_L\|_{0,\infty}^s$. Thus, if $\|\mathbf{D}_L\|_0 \leq \lambda_{L-1}/\lambda_L$ and $\|\boldsymbol{\gamma}_L\|_{0,\infty}^s \leq \lambda_L$, then $\|\boldsymbol{\gamma}_{L-1}\|_{0,\infty}^p \leq \lambda_{L-1}$. Recalling the argument in [16] (Section 7.1), a stripe from the i^{th} layer includes up to $c_i = \lceil (2n_{i-1}-1)/n_i \rceil$ patches. Therefore, $\|\boldsymbol{\gamma}_{L-1}\|_{0,\infty}^s \leq c_L \|\boldsymbol{\gamma}_{L-1}\|_{0,\infty}^p$, and so $\boldsymbol{\gamma}_{L-1}$ will satisfy its corresponding sparsity constraint if $\|\mathbf{D}_L\|_0 \leq \lambda_{L-1}/(c_L\lambda_L)$. Iterating this argument for the remaining layers proves the above lemma.

□