Deeply-Sparse Signal rePresentations (DS^{2}P)

Demba Ba, *Member, IEEE*

***Abstract*—A recent line of work has sought to build a parallel between deep neural network architectures and sparse coding/recovery and estimation. Said line of work suggests, as pointed out by Papyan [1] et al., that a deep neural network architecture with ReLu nonlinearities arises from a finite sequence of cascaded sparse coding models, the outputs of which, except for the last element in the cascade, are sparse and unobservable. That is, intermediate outputs deep in the cascade are sparse, hence the title of this manuscript. We show here, using techniques from the dictionary learning/sparse coding literature, that if the measurement matrices in the cascaded sparse coding model (a) satisfy RIP and (b) all have sparse columns except for the last, they can be recovered with high probability in the absence of noise using an optimization algorithm that, beginning with the last element of the cascade, alternates between estimating the dictionary and the sparse code and then, at convergence, proceeds to the preceding element in the cascade. The method of choice in deep learning to solve this problem is by training an auto-encoder whose architecture we specify. Our algorithm provides a sound alternative, derived from the perspective of sparse coding, and with theoretical guarantees, as well as sample complexity assessments. In particular, the learning complexity depends on the maximum, across layers, of the product of the number of active neurons and the embedding dimension. Our proof relies on a certain type of sparse random matrix satisfying the RIP property. We use non-asymptotic random matrix theory to prove this. We demonstrate the deep dictionary learning algorithm via simulation.**

***Index Terms*—Dictionary Learning, Deep Neural Networks, Sample Complexity**

I. Introduction

Deep learning has been one of the most popular areas of research over the past few years, due in large part to the ability of deep neural networks to outperform humans at a number of cognition tasks, such as object and speech recognition.

Despite the mystique that has surrounded their success, recent work has started to provide answers to questions pertaining, on the one hand, to basic assumptions behind deep networks–when do they work?–and, on the other hand, to interpretability–why do they work? In [2], Patel explains deep learning from the perspective of inference in a hierarchical probabilistic graphical model. This leads to new inference algorithms based on belief propagation and its variants. In a series of articles, the authors from [1], [3], [4] consider deep convolutional networks through the lens of a multi-layer convolutional sparse coding (ML-CSC) model. The authors show a correspondence between the sparse approximation step in this multi-layer model and the encoding step (forward pass) in a related deep convolutional network. Specifically, they show that convolutional neural networks with ReLu nonlinearities can be interpreted as sequential algorithms to solve for the sparse codes in the ML-CSC model. The authors carry out a detailed theoretical analysis of when the sparse recovery algorithms succeed in the absence of noise, and when they are stable win the presence of noise. More recently, building on the work of Papyan et al, Ye et al [5] have shown that some of the key operations that arise in deep learning (e.g. pooling, ReLu) can be understood from the classical theory of filter banks in signal processing. In a separate line of work, Tishby [6] uses the information bottle neck principle from information theory to characterize the limits of a deep network from an information-theoretic perspective.

The works [2] and [1], [3], [4] relate the inference step (forward-pass) of neural networks to various generative models, namely graphical models in the former and the ML-CSC model in the latter. They do not provide theoretical guarantees for learning the filters (weights) of the respective generative models. Here, we take a more expansive approach than in [1], [2], [5] that connects deep networks to the theory of dictionary learning, to answer questions pertaining, not to basic assumptions and interpretability, but to the sample complexity of learning a deep network–how much data do you need to learn a deep network?

Classical dictionary learning theory [7] tackles the problem of estimating a *single* unknown transformation from data obtained through a sparse coding model. The theory gives bounds for the sample complexity of learning a dictionary as a function of the parameters of the sparse coding model. Two key features unite the works from [1], [2] and [5]. The first is (a) sparsity, and the second (b) the use of a hierarchy of transformations/representations as a proxy for the different layers in a deep neural networks. Classical dictionary learning theory does not, however, provide a framework for assessing the complexity of learning a hierarchy, or sequence, of transformations from data.

We formulate a deep version of the classical sparse-coding generative model from dictionary learning [7]: starting with a sparse code, a composition of linear transformations are applied to generate an observation. We constraint all the transformations in the composition, except for the last, to have sparse columns, so that their composition yields sparse representations at every step. We solve the deep dictionary learning–learning all of the transformation in the composition–problems by sequential alternating minimization, starting from the last transformation in the composition up to the first. Each alternating-minimization step involves a sparse approximation step, i.e. a search for a sparse input to each of the transformations in the composition. That's why, we constraint the intermediate matrices in the composition to be sparse. As pointed out by the authors in [8], who introduce the notion

D. Ba is with School of Engineering and Applied Sciences, Harvard University, Cambridge, MA (e-mail: demba@seas.harvard.edu)

Manuscript received April 19, 2005; revised August 26, 2015.

of cosparsity, this a sufficient but not necessary condition for producing sparse outputs at each level. We do not consider the cosparse setting. As we detail in the main text, our notion of depth refers to the number of transformations in the composition.

We begin the rest of our treatment by briefly introducing our notation (Section II). Our main contributions are threefold. First, in Section III we develop the connection between classical dictionary learning and deep recurrent sparse auto-encoders [9], [10], [11]. Second, we use this connection in Section IV to introduce the deep generative model for dictionary learning, and prove that, under regularity assumptions, the sequence of dictionaries can be learned and give a bound on the computational complexity of this learning as a function of the model parameters. Let the transformations in the composition be labeled 1 through L. Further letting r_ℓ be the dimension of the input of the ℓ^{th} transformation and $s_{\mathbf{Y}^{(\ell-1)}}$ the sparsity of this input, the computational complexity is $\mathcal{O}\left(\max_\ell \max(r_\ell^2, r_\ell s_{\mathbf{Y}^{(\ell-1)}})\right)$. As in [7], the term r_ℓ^2 seems to be an over-estimate from the proof techniques used. This bound can be interpreted as a statement regarding the complexity of learning deep versions of the recurrent auto-encoders from Section III. Indeed, in neural networks terminology, r_ℓ is the size of the embedding at ℓ^{th} layer and $s_{\mathbf{Y}^{(\ell-1)}}$ the number active neurons at that layer. Third, our proof relies on a certain type of sparse random matrix satisfying the RIP property. We prove this in Section V using results from non-asymptotic random matrix theory [12]. We demonstrate the deep dictionary learning algorithm via simulation in Section VI. The simulations suggest that the term r_ℓ^2 above is an artifact of the proof techniques we rely on to prove our main result. That is, the learning complexity depends on the maximum, across layers, of the product of the number of active neurons and the embedding dimension. We give concluding remarks in Section VII.

II. Notation

We use bold font for matrices and vectors, capital letters for matrices, and lower-case letters for vectors. For a matrix $\mathbf{A}$, $\mathbf{a}_j$ denotes its j^{th} column vector, and $\mathbf{A}_{ij}$ its element at the i^{th} row and the j^{th} column. For a vector $\mathbf{x}$, x_i denotes its i^{th} element. $\mathbf{A}^{\mathrm{T}}$ and $\mathbf{x}^{\mathrm{T}}$ refer, respectively, to the transpose of the matrix $\mathbf{A}$ and that of the vector $\mathbf{x}$. We use $||\mathbf{x}||_p$ to denote the ℓ_p norm of the vector $\mathbf{x}$. We use $\sigma_{\text{min}}(\mathbf{A})$ and $\sigma_{\text{max}}(\mathbf{A})$ to refer, respectively, to the minimum and maximum singular values of the matrix $\mathbf{A}$. We will also use $||\mathbf{A}||_2$ to denote the spectral norm (maximum singular value of a matrix). We will make it clear from context whether a quantity is a random variable/vector. We use $\mathbf{I}$ to refer the identity matrix. Its dimension we will be clear from the context. Let $r \in \mathbb{N}^+$. For a vector $\mathbf{x} \in \mathbb{R}^r$, $\text{Supp}(\mathbf{x}) \subset \{1, \cdots, r\}$ refers to set of indices corresponding to its nonzero entries.

III. Shallow Neural Networks and Sparse Estimation

The rectifier-linear unit–ReLu–is a popular nonlinearity in the neural-networks literature. Let $z \in \mathbb{R}$, the ReLu nonlinearity is the scalar-valued function defined as $\text{ReLu}(z) = \max(z, 0)$. In this section, we build a parallel between sparse approximation/ℓ_1-regularized regression and the ReLu$(\cdot)$ nonlinearity. This leads to a parallel between dictionary learning [7] and auto-encoder networks [10], [13]. In turn, this motivates an interpretation of learning a deep neural networks as a hierarchical, i.e. sequential, dictionary learning problem in cascade-of-sparse-coding models [1].

A. Unconstrained ℓ_1-regularization in one dimension

We begin by a derivation of the soft-thresholding operator from the perspective of sparse approximation. Let $y \in \mathbb{R}$, $\lambda > 0$, and consider the inverse problem

$$\min_{x \in \mathbb{R}} \frac{1}{2}(y - x)^2 + \lambda |x|. \tag{1}$$

It is well-known that the solution $\hat{x}$ to Equation 1 is given by the soft-thresholding operator, i.e.

$$\hat{x} = \text{sng}(y)\max(|y| - \lambda, 0) \coloneqq s_\lambda(y) = \text{ReLu}(y - \lambda) - \text{ReLu}(-y - \lambda). \tag{2}$$

For completeness, we give the derivation of this result in the appendix.

We show next that, subject to a non-negativity constraint, the solution to Equation 1 is a translated version of ReLu [14] applied to y, a form that is more familiar to researchers from the neural-networks community. That is, the ReLu$(\cdot)$ nonlinearity arises in the solution to a simple constrained ℓ_1-regularized regression problem in one dimension.

B. Constrained ℓ_1-regularization in one dimension

Consider the inverse problem

$$\min_{x \in \mathbb{R}^+} \frac{1}{2}(y - x)^2 + \lambda x. \tag{3}$$

The solution to Equation 3 is $\hat{x} = \max(y - \lambda, 0) = \text{ReLu}(y - \lambda)$.

For $y > 0$, the solution to Equation 3 is equivalent to that of Equation 1. For $y \leq 0$, the solution must be $\hat{x} = 0$. Suppose, for a contradiction, that $x > 0$, then the value of the objective function is $\frac{1}{2}(y - x)^2 + \lambda x$, which is strictly greater that $\frac{1}{2}y^2$, i.e. the objective function evaluated at $x = 0$.

The above result generalizes easily to the case when the observations and the optimization variable both live in higher dimensions and are related through a unitary transform.

C. Unconstrained ℓ_1-regularization in more that one dimension

Let $r \in \mathbb{N}^+$ and $\mathbf{y} \in \mathbb{R}^{r+}$, $\lambda > 0$, and $\mathbf{A}$ a unitary matrix. Consider the problem

$$\min_{\mathbf{x} \in \mathbb{R}^{r+}} \frac{1}{2} ||\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2 + \lambda ||\mathbf{x}||_1 . \tag{4}$$

Since $\mathbf{A}$ is unitary, i.e. an isometry, Equation 4 is equivalent to

$$\min_{\mathbf{x} \in \mathbb{R}^{r+}} \frac{1}{2} ||\tilde{\mathbf{y}} - \mathbf{x}||_2^2 + \lambda ||\mathbf{x}||_1 = \min_{\mathbf{x} \in \mathbb{R}^{r+}} \sum_{j=1}^{r} \frac{1}{2}(\tilde{y}_j - x_j)^2 + \lambda x_j, \tag{5}$$

where $\tilde{y}_j = \mathbf{a}_j^{\mathrm{T}}\mathbf{y}$, $j = 1, \cdots, r$. Equation 5 is separable in $x_1, \cdots, x_r$. For each x_j, the optimization is equivalent to Equation 3 with $\tilde{y}_j$ as the input, $j = 1, \cdots, r$. Therefore,

$$\hat{x}_j = \max(\tilde{y_j} - \lambda, 0) \quad (6)$$

$$= \begin{cases} \mathbf{a}_j^{\mathrm{T}}\mathbf{y} - \lambda & \text{if } \mathbf{a}_j^{\mathrm{T}}\mathbf{y} > \lambda; \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

$$= \begin{cases} \begin{bmatrix} \mathbf{a}_j \\ -\lambda \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} & \text{if } \begin{bmatrix} \mathbf{a}_i \\ -\lambda \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} > 0; \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

$$= \text{ReLu}\left(\begin{bmatrix} \mathbf{a}_j \\ -\lambda \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} \right). \quad (9)$$

Equation 9 states that, for $\mathbf{A}$ unitary, the solution to the ℓ_1-regularized least-squares problem with non-negativity constraints (Equation 4) is obtained component-wise, by projecting the vector $\begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix}$ onto the vector $\begin{bmatrix} \mathbf{a}_j \\ -\lambda \end{bmatrix}^{\mathrm{T}}$ and passing it through the ReLu$(\cdot)$. nonlinearity. Stated otherwise, a simple feed-forward neural network solves the inverse problem of Equation 4. Equation 9 also suggests that $-\lambda$ plays the role of the bias in neural networks. Allowing for different biases is akin to using a different regularization parameter for each of the components of $\mathbf{x}$. Applying the transformation $\mathbf{A}$ to he vector $\hat{\mathbf{x}}$ yields an approximate reconstruction $\hat{\mathbf{y}} = \mathbf{A}\hat{\mathbf{x}}$. We depict this two-stage process as a two-layer feed-forward neural network in Figure 1. The architecture depicted in the figure is called an auto-encoder [10], [13]. Given training examples, the weights of the network, which depend on $\mathbf{A}$, can be tuned by backpropagation. This suggests a connection between dictionary learning and auto-encoder architectures, which we elaborate upon below.

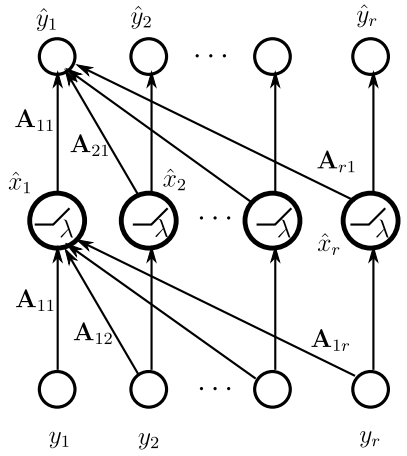

Fig. 1. Auto-encoder architecture motivated by the interpretation of the ReLu in the context of sparse approximation. The encoder solves the constrained ℓ_1-regularized least-squares problem. The decoder reconstructs the observations by applying the dictionary to the output of the encoder. The only parameter of this architecture is the dictionary $\mathbf{A}$, which can be learned by backpropagation. To simplify the figure, we only draw a subset of all of the connections.

Remark 1: The literature suggests that the parallel between the ReLu and sparse approximation dates to the work of [15]. Prior to this, while they do not explicitly make this connection, the authors from [14] discuss in detail the sparsity-promoting properties of the ReLu compared to other nonlinearities in neural networks.

D. Sparse coding, dictionary learning, and auto-encoders

We use the relationship between the ReLu$(\cdot)$ and ℓ_1-regularized regression to draw a parallel between dictionary learning [7] and a specific auto-encoder [10], [13] neural-network architecture.

a) Shallow sparse generative model.: Let $\mathbf{Y}$ be an $d \times n$ real-valued matrix generated as follows

$$\mathbf{Y} = \mathbf{A}\mathbf{X}, \mathbf{A} \in \mathbb{R}^{d \times r}, \mathbf{X} \in \mathbb{R}^{r \times n}. \quad (10)$$

Each column of $\mathbf{X}$ is an s-sparse vector, i.e. only s of its elements are non-zero [7]. The non-zero elements represent the coordinates or codes for the corresponding column of $\mathbf{Y}$ in the dictionary $\mathbf{A}$ [7].

Remark 2: We call this model "shallow" because there is only one transformation $\mathbf{A}$ to learn. In Section IV, we will contrast this with a "deep" generative model where we will learn each of the transformations that comprise the composition of multiple linear transformations applied to a sparse code.

b) Sparse coding and dictionary learning.: Given $\mathbf{Y}$, the goal is to estimate $\mathbf{A}$ and $\mathbf{X}$. Alternating minimization [7] is a popular algorithm to find $\mathbf{A}$ and $\mathbf{X}$ as the solution to

$$(\hat{\mathbf{A}}, \hat{\mathbf{X}}) = \arg\min_{\mathbf{X},\mathbf{A}} ||\mathbf{x}_i||_1, \forall i = 1, \cdots, n \text{ s.t. } \mathbf{Y} = \mathbf{A}\mathbf{X}, ||\mathbf{a}_j||_2 = 1, \forall j = 1, \cdots, r. \quad (11)$$

The algorithm solves Equation 11 by alternating between a sparse coding step, which updates the sparse codes given an estimate of the dictionary, and a dictionary update step, which updates the dictionary given estimates of the sparse codes. In unconstrained form, the sparse-coding step of the t^{th} iteration of Algorithm 1 is equivalent to solving Eq. 4, with a value for λ that depends on ϵ_t.

Suppose that instead of requiring equality in Equation 10, our goal where instead to solve the following problem

$$\min_{\mathbf{A},\mathbf{X}} \frac{1}{2} ||\mathbf{Y} - \mathbf{A}\mathbf{X}||_{\mathrm{F}}^2 + \lambda \sum_{i=1}^{n} ||\mathbf{x}_i||_1. \quad (12)$$

If $\mathbf{A}$ were a unitary matrix, the sparse-coding step could be solved exactly using Equation 9. The goal of the dictionary-learning step is to minimize the reconstruction error between $\mathbf{A}$ applied to the sparse codes, and the observations. In the neural-network literature, this two-stage process describes so-called auto-encoder architectures [10], [13].

Remark 3: We make the assumption that $\mathbf{A}$ is unitary to simplify the discussion and make the parallel between neural networks and dictionary learning more apparent. If $\mathbf{A}$ is not unitary, we can replace Equation 10 with the iterative soft-thresholding algorithm (ISTA) [16]: $\mathbf{x}_k = \text{soft}_{\frac{\lambda}{M}}\left(\mathbf{x}_{k-1} + \mathbf{A}^{\mathrm{T}}(\mathbf{y} - \mathbf{A}\mathbf{x}_{k-1})\right)$, where k indexes the iterations of the algorithm, and $M \geq \sigma_{\max}(\mathbf{A}^{\mathrm{T}}\mathbf{A})$.

c) Shallow, constrained, recurrent, spare auto-encoders.: We introduce an auto-encoder architecture for learning the model from Equation 10. This auto-encoder has an implicit connection with the alternating-minimization algorithm applied to the same model. Given $\mathbf{A}$, the encoder produces a sparse code using a finite (large) number of iterations of the ISTA algorithm [16]. The decoder applies $\mathbf{A}$ to the output of the decoder to reconstruct $\mathbf{y}$. We call this architecture a constrained recurrent sparse auto-encoder (CRsAE) [11], [17]. The constraint comes from the fact that the operations used by the encoder and the decoder are tied to each other through $\mathbf{A}$. Hence, the encoder and decoder are not independent, unlike in [10], [13]. The auto- encoder is called recurrent because of the ISTA algorithm, which is an iterative procedure. Figure 2 depicts this architecture.

There are two basic take-aways from the previous discussion

1) Constrained auto-encoders with ReLu nonlinearities capture the essence of the alternating-minimization algorithm for dictionary learning.
2) Therefore, the sample complexity of dictionary learning can give us insights on the hardness of learning neural networks.

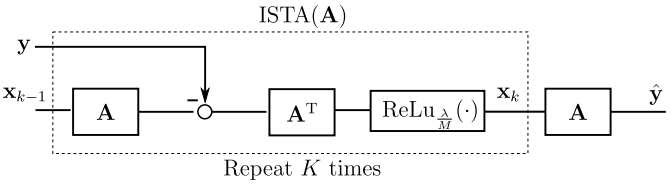

Fig. 2. Auto-encoder architecture motivated by alternating-minimization algorithm for sparse coding and dictionary learning. The encoder uses K (K large) iterations of the ISTA algorithm for sparse coding, starting with a guess $\mathbf{x}_0$ of the sparse code. The decoder reconstructs the observations by applying the dictionary to the output of the encoder. The only parameter of this architecture is the dictionary $\mathbf{A}$, which can be learned by backpropagation. L is a constant such that $M \geq \sigma_{\max}(\mathbf{A}^{\mathrm{T}}\mathbf{A})$.

d) How to use dictionary learning to assess the sample complexity of learning deep networks?: The "depth" of a neural network refers to the number of its hidden layers, excluding the output layer. A "shallow" network is one with two or three hidden layers [18]. A network with more than three hidden layers is typically called "deep". Using this definition, the architecture from Figure 2 would be called deep. This is because of K iterations of ISTA which, when unrolled [9], [10], [11] would constitute K separate layers. This definition, however, does not reflect the fact that the only unknown in the network is $\mathbf{A}$. Therefore, the number of parameters of the network is the same as that in a one-layer, fully-connected, feed-forward network.

A popular interpretation of deep neural networks is that they learn a hierarchy, or sequence, of transformations of data. Motivated by this interpretation, we define the *"depth" of a network*, not in relationship to its number of layers, but *as the number of underlying distinct transformations/mappings to be learned.*

Classic dictionary learning tackles the problem of estimating a *single* transformation from data [7]. Dictionary-learning theory characterizes the sample complexity of learning the model of Equation 10 under various assumptions. We can use these results to get insights on the complexity of learning the parameters of the auto-encoder from Figure 2. Classical dictionary learning theory does not, however, provide a framework for assessing the complexity of learning a hierarchy, or sequence, of transformations from data.

IV. Deep Sparse Signal Representations

Our goal is to build a "deep" (in the sense defined previously) version of the model from Equation 10, i.e. a generative model in which, starting with a sparse code, a composition of linear transformations are applied to generate an observation. What properties should such a model obey? In the previous section, we used the sparse generative model of Equation 10 to motivate the auto-encoder architecture of Figure 2. *The goal of the encoder is to produce sparse codes* [14]. We will construct a "deep" version of the auto-encoder and use it to infer desirable properties of the "deep" generative model.

A. Deep, constrained, recurrent, sparse auto-encoders

For simplicity, let us consider the case of two linear transformations $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$. $\mathbf{A}^{(1)}$ is applied to a sparse code, and $\mathbf{A}^{(2)}$ to its output to generate an observation. Applied to an observation, the goal of the ISTA($\mathbf{A}^{(2)}$) encoder is to produce sparse codes. One scenario when this will happen is if $\mathbf{A}^{(1)}$ applied to the sparse code produces sparse/approximately sparse observations, i.e. the image of $\mathbf{A}^{(1)}$ must be sparse/approximately sparse.

For the composition of more than two transformations, the requirement that the encoders applied in cascade produce sparse codes suggests that, starting with a sparse code, the output of each of the transformations, expect for the very last which gives the observations, must be approximately sparsely. As pointed out by the authors in [8], who introduce the notion of cosparsity, this a sufficient but not necessary condition. We do not consider the cosparse setting.

We specify our deep sparse generative model below, along with the assumptions that accompany the model.

B. Deep sparse generative model and coding

Let $\mathbf{Y}$ be the $d_L \times n$ real-valued matrix obtained by applying the composition of L linear transformations $\{\mathbf{A}^{(\ell)}\}_{\ell=1}^{L}$ to a matrix $\mathbf{X}$ of sparse codes

$$\begin{aligned} &\mathbf{Y} = \mathbf{A}^{(L)} \cdots \mathbf{A}^{(2)}\mathbf{A}^{(1)}\mathbf{X}, \\ &\mathbf{X} \in \mathbb{R}^{r_1 \times n}, \mathbf{A}^{(\ell)} \in \mathbb{R}^{d_\ell \times r_\ell}; \\ &\forall \ell = 1, \cdots, L-1, \text{all columns of } \mathbf{A}^{(\ell)} \text{ are } s^{(\ell)} \text{ sparse,} \\ &\text{and the nonzero entries uniformly bounded.} \end{aligned} \tag{13}$$

If we further assume that each column of $\mathbf{X}$ is s-sparse, i.e. at most s of the entries of each column are nonzero, the image of each of the successive transformations $\{\mathbf{A}^{(\ell)}\}_{\ell=1}^{L-1}$ will also be sparse. Finally, we apply the transformation $\mathbf{A}^{(L)}$ to obtain the observations $\mathbf{Y}$.

Given $\mathbf{Y}$, we would like solve the following problem

$$\min_{\mathbf{X},\{\mathbf{A}^{(\ell)}\}_{\ell=1}^{L}} ||\mathbf{x}_i||_1, \forall i = 1,\cdots,n \quad \text{s.t. } \mathbf{Y} = \mathbf{A}^{(L)}\cdots\mathbf{A}^{(2)}\mathbf{A}^{(1)}\mathbf{X}, \left|\left|\mathbf{a}_j^{(\ell)}\right|\right|_2 = 1, \forall j = 1,\cdots,r_\ell; \forall \ell = 1,\cdots,L. \tag{14}$$

Remark 4: If $\mathbf{A}^{(2)} = \mathbf{A}^{(3)} = \cdots = \mathbf{A}^{(L)} = \mathbf{I}$, Equation 13 reduces to the "shallow" sparse generative model from Equation 10, a problem that is well-studied in dictionary-learning literature [7], and for which the authors propose an alternating-minimization procedure whose theoretical properties they study in detail.

In what follows, it will be useful to define the matrix $\mathbf{Y}^{(\ell)} = \prod_{\ell'=1}^{\ell}\mathbf{A}^{(\ell)}\mathbf{X}$, namely the output of the ℓ^{th} operator in Equation 13, $\ell = 1,\cdots,L$. At depth ℓ, the columns of $\mathbf{Y}^{(\ell)}$, $\ell = 1,\cdots,L-1$ are sparse representations of the signal $\mathbf{Y}$, i.e. they are *deeply sparse*.

a) Reduction to a sequence of "shallow" problems: the case $L = 2$.: To solve Equation 14, we will reduce to the problem to a sequence of "shallow" problems of the form studied in [7]. To gain some intuition, let us consider the case when $L = 2$. We will proceed as follows

Step 1. Find $\mathbf{A}^{(2)}$ and $\mathbf{Y}^{(1)}$: We first solve the following problem

$$(\hat{\mathbf{A}}^{(2)}, \hat{\mathbf{Y}}^{(1)}) = \arg\min_{\mathbf{A}^{(2)},\mathbf{Y}^{(1)}} \left|\left|\mathbf{y}_i^{(1)}\right|\right|_1, \forall i = 1,\cdots,n \quad \text{s.t. } \mathbf{Y} = \mathbf{A}^{(2)}\mathbf{Y}^{(1)}, \left|\left|\mathbf{a}_j^{(2)}\right|\right|_2 = 1, \forall j = 1,\cdots,r_2. \tag{15}$$

We solve Equation 15 using the alternating-minimization algorithm from [7]–Algorithm 1 below–which under regularity assumptions, guarantees that, with high probability, $(\hat{\mathbf{A}}^{(2)}, \hat{\mathbf{Y}}^{(1)}) = (\mathbf{A}^{(2)}, \mathbf{Y}^{(1)})$.

Step 2. Find $\mathbf{A}^{(1)}$ and $\mathbf{X}$. We can now solve

$$(\hat{\mathbf{A}}^{(1)}, \hat{\mathbf{X}}) = \arg\min_{\mathbf{A}^{(1)},\mathbf{X}} ||\mathbf{x}_i||_1, \forall i = 1,\cdots,n, \quad \text{s.t. } \mathbf{Y}^{(1)} = \mathbf{A}^{(1)}\mathbf{X}, \left|\left|\mathbf{a}_j^{(1)}\right|\right|_2 = 1, \forall j = 1,\cdots,r_1. \tag{16}$$

Appealing once again to Algorithm 1 from [7], we can conclude that, with high probability, we have solved for $\mathbf{A}^{(2)}$, $\mathbf{A}^{(1)}$ and $\mathbf{X}$.

Remark 5: At this point, the reader would be justified in asking the following question: $\mathbf{A}^{(1)}$ is a matrix with sparse columns that should satisfy RIP, do such matrices exist? In Section V, we will answer this question in the affirmative for a certain class of matrices for which the nonzero entries of each column are chosen at random. We will appeal to standard results from random matrix theory [12].

We now state explicitly our assumptions on the "deep" generative model of Equation 13. These assumptions will let us give guarantees and sample-complexity estimates for the success, for arbitrary L, of the sequential alternating-minimization algorithm described above for $L = 2$. The reader can compare these assumptions to assumptions A1–A7 from [7]. As in [7], we assume, without any loss in generality that the columns of $\{\mathbf{A}^{(\ell)}\}_{\ell=1}^{L}$ all have unit ℓ_2 norm, i.e. $\left|\left|\mathbf{a}_j^{(\ell)}\right|\right|_2 = 1$, $j = 1,\cdots,r_\ell$, $\ell = 1,\cdots,L$.

Algorithm 1: AltMinDict($\mathbf{Y}$,$\mathbf{A}(0)$,ϵ_0,s,T): Alternating minimization algorithm for dictionary learning

Input: Samples $\mathbf{Y}$, initial dictionary estimate $\mathbf{A}(0)$, accuracy sequence ϵ_t, sparsity level s, and number of iterations T.

for $t = 0$ *to* $T-1$ **do**
  for $i = 1$ *to* n **do**
    $\mathbf{X}(t+1)_i = \arg\min_{\mathbf{x}}$ s.t. $||\mathbf{y}_i - \mathbf{A}(t)\mathbf{x}||_2 \leq \epsilon_t$
  Threshold: $\mathbf{X}(t+1) = \mathbf{X}(t+1) \cdot *(\mathbb{I}[\mathbf{X}(t+1) > 9s\epsilon_t])$
  Estimate $\mathbf{A}(t+1) = \mathbf{Y}\mathbf{X}(t+1)^+$
  Normalize: $\mathbf{A}(t+1)_i = \frac{\mathbf{a}(t+1)_i}{||\mathbf{a}(t+1)_i||_2}$

Output: $(\mathbf{A}(T), \mathbf{X}(T))$

b) Assumptions:: Let $\mathbf{Y}^{(0)} = \mathbf{X}$, $s_{\mathbf{Y}^{(0)}} = s$, and $\forall \ell = 1,\cdots,L$, $s_{\mathbf{Y}^{(\ell)}} = s\prod_{\ell'=1}^{\ell} s^{(\ell)}$, $\ell = 1,\cdots,L-1$.

(A1) **Dictionary Matrices satisfying RIP**: For each $\ell = 1,\cdots,L$, the dictionary matrix $\mathbf{A}^{(\ell)}$ has $2s_{\mathbf{Y}^{(\ell-1)}}$-RIP constant of $\delta_{2s_{\mathbf{Y}^{(\ell-1)}}} < 0.1$.

(A2) **Spectral Condition of Dictionary Elements:** For each $\ell = 1,\cdots,L$, the dictionary matrix $\mathbf{A}^{(\ell)}$ has bounded spectral norm, for some constant $\mu^{(\ell)} > 0$, $\left|\left|\mathbf{A}^{(\ell)}\right|\right|_2 < \mu^{(\ell)}\sqrt{\frac{r_\ell}{d_\ell}}$.

(A3) **Non-zero Entries in Coefficient Matrix**: The non-zero entries of $\mathbf{X}$ are drawn i.i.d. from a distribution such that $\mathbb{E}[(\mathbf{X}_{ij})^2] = 1$, and satisfy the following a.s.: $|\mathbf{X}_{ij}| \leq M^{(0)}, \forall, i, j$.

(A4) **Sparse Coefficient Matrix**: The columns of the coefficient matrix have s non-zero entries which are selected uniformly at random from the set of all s-sized subsets of $\{1,\cdots,r_1\}$. It is required that $s \leq \frac{d_1^{1/6}}{c_2\mu^{(1)\,1/3}}$, for some universal constant c_2. We further require that, for $\forall \ell = 1,\cdots,L-1$, $s_{\mathbf{Y}^{(\ell)}} \leq \frac{d_{\ell+1}^{1/6}}{c_2\mu^{(\ell+1)\,1/3}}$.

(A5) **Sample Complexity**: For some universal constant $c_3 = 4$, and given failure parameters $\{\delta_\ell\}_{\ell=1}^{L} > 0$, the number of samples n needs to satisfy, $\forall \ell = 1,\cdots,L$

$$n \geq c_3 \max(r_\ell^2, r_\ell {M^{(\ell-1)}}^2 s_{\mathbf{Y}^{(\ell-1)}}) \log\left(\frac{2r_\ell}{\delta}\right). \tag{17}$$

Here $\frac{|\mathbf{Y}_{ij}^{(\ell)}|}{\sigma^{(\ell)}} \leq M^{(\ell)}$, $\sigma^{2(\ell)} = \text{var}\mathbf{Y}_{ij}^{(\ell)}$, $\ell = 1,\cdots,L-1$.

(A6) **Initial dictionary with guaranteed error bound**: It is assumed that, $\forall \ell = 1,\cdots,L$, we have access to an initial dictionary estimate $\mathbf{A}^{(\ell)}$ such that

$$\max_{i\in\{1,\cdots,r_\ell\}} \min_{z\in\{-1,1\}} \left|\left|z\mathbf{A}_i^{(\ell)}(0) - \mathbf{A}_i^{(\ell)}\right|\right|_2 \leq \frac{1}{2592 s_{\mathbf{Y}^{(\ell-1)}}^2}. \tag{18}$$

(A7) **Choice of Parameters for Alternating Minimization**: For all $\ell = 1,\cdots,L$, AltMinDict($\mathbf{Y}^{(\ell)}$, $\mathbf{A}^{(\ell)}$, $\epsilon_0^{(\ell)}$) uses a

sequence of accuracy parameters $\epsilon_0^{(\ell)} = \frac{1}{2592 s^2_{\mathbf{Y}^{(\ell-1)}}}$ and

$$\epsilon_{t+1}^{(\ell)} = \frac{25050 \mu_\ell s^3_{\mathbf{Y}^{(\ell-1)}}}{\sqrt{d_\ell}} \epsilon_t^{(\ell)}. \tag{19}$$

We are now in a position to state our main result regarding the ability to learn the "deep" generative model of Equation 13, i.e. recover $\{\mathbf{A}^{(\ell)}\}_{\ell=1}^L$ under assumptions A1–A7.

C. Learning the "deep" sparse coding model by sequential alternating minimization

Algorithm 2 describes the "deep" dictionary learning algorithm. The Algorithm requires the specification of a variable $1 \leq \bar{\ell} \leq L$. Given $1 \leq \bar{\ell} \leq L$, Algorithm 2 solves for $\{\hat{\mathbf{A}}^{(\ell)}\}_{\ell=\bar{\ell}}^L$, starting with $\hat{\mathbf{A}}^{(L)}$ and then sequentially working its to $\hat{\mathbf{A}}^{(\bar{\ell})}$.

Algorithm 2: Deep dictionary learning algorithm

Input: Samples $\mathbf{Y}$, number of levels $1 \leq \bar{\ell} \leq L$, initial dictionary estimates $\{\mathbf{A}^{(\ell)}(0)\}_{\ell=\bar{\ell}}^L$, accuracy sequences $\{\epsilon_t^{(\ell)}\}_{\ell=\bar{\ell}}^L$, sparsity levels $\{s_{\mathbf{Y}^{(\ell-1)}}\}_{\ell=\bar{\ell}}^L$.

$\hat{\mathbf{Y}}^{(L)} = \mathbf{Y}$
$\ell = L$
while $\ell \geq \bar{\ell}$ **do**
  $\left(\hat{\mathbf{A}}^{(\ell)}, \hat{\mathbf{Y}}^{(\ell-1)}\right) =$ AltMinDict$(\hat{\mathbf{Y}}^{(\ell)}, \mathbf{A}^{(\ell)}(0), \epsilon_0^{(\ell)}, s_{\mathbf{Y}^{(\ell-1)}}, \infty)$
  $\ell = \ell - 1$

Output: $\{\hat{\mathbf{A}}^{(\ell)}, \hat{\mathbf{Y}}^{(\ell-1)}\}_{\ell=\bar{\ell}}^L$

Theorem 1 (Exact recover of the "deep" generative model): Let us denote by E_ℓ the event $\{\hat{\mathbf{A}}^{(\ell)} = \mathbf{A}^{(\ell)}\}$, $\ell = 1, \cdots, L$. Let $1 \leq \bar{\ell} \leq L$, then $\forall \bar{\ell}$

$$\mathbb{P}[\cap_{\ell=\bar{\ell}}^L E_\ell] \geq \prod_{\ell=\bar{\ell}}^L (1 - 2\delta_\ell). \tag{20}$$

The Theorem states that, with the given probability, we can learn all of the L transformations in the deep sparse generative model. Assumption A5 is a statement about the complexity of this learning: the computational complexity is $\mathcal{O}\left(\max_\ell \max(r_\ell^2, r_\ell s_{\mathbf{Y}^{(\ell-1)}})\right)$. This can be interpreted as a statement regarding the complexity of learning deep versions of the recurrent auto-encoders from Section III. Indeed, in neural networks terminology, r_ℓ is the size of the embedding at ℓ^{th} layer and $s_{\mathbf{Y}^{(\ell-1)}}$ the number active neurons at that layer. The simulations (Section VI) suggest that the term r_ℓ^2 above is an artifact of the proof techniques we rely on to arrive at our main result. That is, the learning complexity depends on the maximum, across layers, of the product of the number of active neurons and the embedding dimension.

We will prove the result by induction on $\bar{\ell}$. Before proceeding with the proof, let us discuss in detail the case when $L = 2$ in Equation 13 and $\bar{\ell} = 1$. We focus on exact recovery of $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(1)}$ and defer computation of the probability in Equation 20 to the proof that will follow.

Intuition behind the proof: Algorithm 2 begins by solving for $\left(\hat{\mathbf{A}}^{(2)}, \hat{\mathbf{Y}}^{(1)}\right)$. If we can show that the algorithm succeeds for this pair, in particular that $\hat{\mathbf{A}}^{(2)} = \mathbf{A}^{(2)}$, then it follows that $\hat{\mathbf{A}}^{(1)} = \mathbf{A}^{(1)}$ in the following iteration of the algorithm. This is because, if the first iteration were to succeed, then $\mathbf{Y}^{(1)} = \mathbf{A}^{(1)}\mathbf{X}$, which is the very model of Equation 10, which was treated in detail [7]. If we can show that the *sparse* matrix $\mathbf{A}^{(1)}$ follows RIP–topic of the the next section Section V–then we can apply Theorem 1 from [7] to guarantee recovery of $\mathbf{A}^{(1)}$.

Focusing on $\mathbf{Y} = \mathbf{A}^{(2)}\mathbf{Y}^{(1)}$, the key point we would like to explain is that, in Equation 10, the properties of $\mathbf{X}$ that allow the authors from [7] to prove their main result also apply to $\mathbf{Y} = \mathbf{A}^{(2)}\mathbf{Y}^{(1)}$. This is not directly obvious because $\mathbf{Y}^{(1)}$ is the product of a sparse matrix and the matrix $\mathbf{X}$ of codes. We address these points one at a time in the following remark

Remark 6:

1) We first note that, since the columns of $\mathbf{X}$ are i.i.d., so are those of $\mathbf{Y}^{(1)}$. Moreover, since both the entries of $\mathbf{A}^{(1)}$ and $\mathbf{X}$ are bounded by assumptions, so are those of $\mathbf{Y}^{(1)}$.
2) By construction, $\mathbf{Y}^{(1)}$ is a sparse matrix: each of its columns is at most $s \cdot s^{(1)}$ sparse. It is not trivial, however, to compute $\mathbb{P}[\mathbf{Y}_{ij}^{(1)} \neq 0]$. Luckily, we do not need this probability explicitly, as long as we can either bound it, or bound the singular values of the matrix $\mathbf{Y}^{(1)}$ and the matrix of indicators values of its nonzero entries. It is not hard to show that $\mathbb{P}[\mathbf{Y}_{ij}^{(1)} \neq 0] \leq \frac{s \cdot s^{(1)}}{r_2}$
3) It is not hard to show that Lemma 3.2 from [7] applies to $\mathbf{Y}^{(1)}$. Lemma 3.2 relies on Lemmas A1 and A2, which give bounds for the matrix of indicator values for the nonzero entries of $\mathbf{X}$. For $\mathbf{Y}^{(1)} = \mathbf{A}^{(1)}\mathbf{X}$, we can replace, in the proof of Lemma 3.2, the matrix of indicators of its nonzero entries with the product of the matrix of indicators of the nonzero entries for $\mathbf{A}^{(1)}$ and $\mathbf{X}$ respectively. This yields a bound that now depends on the n, r_2 and the sparsity level $s \cdot s^{(1)}$.
4) Applying Lemma A1 and A2 from [7] to $\mathbf{A}^{(1)}$ and $\mathbf{X}$, respectively, yields bounds for their lowest and largest singular values. Using standard singular value relationships, this gives a version of Lemma A2 for —$\mathbf{Y}^{(1)}$, i.e. bounds for its lowest and largest singular values. A version of Lemmas A3 and A4 for $\mathbf{Y}^{(1)}$ directly follows.
5) Since $\mathbf{Y}^{(1)}$ is $s \cdot s^{(1)}$-sparse, $\mathbb{P}[\mathbf{Y}_{ij}^{(1)} \neq 0] \leq \frac{s \cdot s^{(1)}}{r_2}$, we can prove the Bernstein inequalities to obtain the *upper bounds* from Lemma A5. These upper bounds are all that are necessary to prove Lemma A6 for $\mathbf{Y}^{(1)}$.
6) The version of Lemma 3.3–the center piece of [7]–for $\mathbf{Y}^{(1)}$ them follows.

   The interested reader can verify all of the above for herself. A detailed technical exposition of these points would lead to a tedious and unnecessary digression, without adding much intuition. Using induction, it can be shown that this remark applied to $\mathbf{Y}^{(\ell)}$ for all $\ell = 1, \cdots, L-1$.

We can now to apply Theorem 1 from [7] to $\mathbf{Y} = \mathbf{A}^{(2)}\mathbf{Y}^{(1)}$,

guaranteeing recovery of $\mathbf{A}^{(2)}$.

Proof 1: We proceed by induction on $\bar{\ell}$.

Base case: $\bar{\ell} = L$. In this case, $\mathbf{Y} = \mathbf{A}^{(L)}\mathbf{Y}^{(L-1)}$. Following the remark above, $\mathbf{Y}^{(L-1)}$ obeys the properties of $\mathbf{X}$ from Theorem 1 in [7]. Under A1–A7, this theorem guarantees that $\hat{\mathbf{A}}^{(L)}$, the limit as $T \to \infty$ of $\mathbf{A}^{(L)}(T)$ converges to $\mathbf{A}^{(L)}$ with probability at least $1-2\delta_L$. Therefore, $\mathbb{P}[E_L] \geq 1-2\delta_L$, proving the base case.

Induction: Suppose the Theorem is true for $\bar{\ell}$, we will show that is true for $\bar{\ell}-1$.

Conditioned on the event $\cap_{\ell=\bar{\ell}}^{L} E_\ell$, $\hat{\mathbf{Y}}^{(\bar{\ell}-1)} = \mathbf{Y}^{(\bar{\ell}-1)} = \mathbf{A}^{(\bar{\ell}-1)}\mathbf{Y}^{(\bar{\ell}-2)}$. Therefore, under A1–A7, the limit $\hat{\mathbf{A}}^{(\bar{\ell}-1)}$ as $T \to \infty$ of $\mathbf{A}^{(\bar{\ell}-1)}(T)$ converges to $\mathbf{A}^{(\bar{\ell}-1)}$ with probability at least $1-2\delta_{\bar{\ell}-1}$. Therefore, $\mathbb{P}[\cap_{\ell=\bar{\ell}-1}^{L} E_\ell] = \mathbb{P}[E_{\bar{\ell}-1} | \cap_{\ell=\bar{\ell}}^{L} E_\ell]\mathbb{P}[\cap_{\ell=\bar{\ell}}^{L} E_\ell] = (1-2\delta_{\bar{\ell}-1})\prod_{\ell=\bar{\ell}}^{L}(1-2\delta_\ell) = \prod_{\ell=\bar{\ell}-1}^{L}(1-2\delta_\ell)$.

This completes the proof.

D. Alternate algorithm for learning the "deep" generative model

Algorithm 2 learns the model of Equation 13 sequentially, starting with $\mathbf{A}^{(L)}$ and ending with $\mathbf{A}^{(1)}$. In this section, we sketch out a learning procedure, Algorithm 3, that proceeds in the opposite way. We begin by giving the intuition for this procedure for the case $L = 3$.

Alternate learning algorithm: the case $L = 3$. As in the case of Algorithm 2, the procedure relies on the sequential application of Algorithm 1. We first learn the product $\mathbf{A}^{(3)}\mathbf{A}^{(2)}\mathbf{A}^{(1)}$. Having learned this product, we then use it to learn the product $\mathbf{A}^{(3)}\mathbf{A}^{(2)}$, which automatically yields $\mathbf{A}^{(1)}$. Finally, we use $\mathbf{A}^{(3)}\mathbf{A}^{(2)}$ to learn $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$. Algorithm 3, and the procedure just described, are related to the hierarchical version [19] of the Proximal Alternating Linearized Minimization [20], except that the dictionary update stage in PALM employs a proximal operator, as opposed to least-squares in our case.

The sequential procedure described above poses, however, one technical difficulty. To learn the product $\mathbf{A}^{(3)}\mathbf{A}^{(2)}\mathbf{A}^{(1)}$, a sufficient condition [7] is that it must satisfy RIP of order $2s$. Assumptions A1 only requires that the matrices $\mathbf{A}^{(3)}$, $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(1)}$ satisfy RIP *separately*. We now show that assumption A1 has implications on the RIP constant of a certain order of the product matrix.

Before stating the result, we introduce some notation and present the alternate algorithm. We let $\mathbf{A}^{(0)} = \mathbf{Y}^{(0)} = \mathbf{X}$ and

$$\mathbf{A}^{(\bar{\ell}\to L)} = \prod_{\ell=\bar{\ell}}^{L} \mathbf{A}^{(\ell)} \quad (21)$$

$$\mathbf{Y} = \mathbf{A}^{(\bar{\ell}\to L)}\mathbf{Y}^{(\bar{\ell}-1)}, \forall \bar{\ell} = 1, \cdots, L. \quad (22)$$

$$\mathbf{A}^{(0\to L)} = \mathbf{Y}. \quad (23)$$

Algorithm 3: Alternate algorithm for deep dictionary learning algorithm

Input: Samples $\mathbf{Y}$, number of levels $1 \leq \bar{\ell} \leq L$, initial dictionary estimates $\{\mathbf{A}^{(\ell\to L)}(0)\}_{\ell=1}^{\bar{\ell}}$, accuracy sequences $\{\epsilon_t^{(\ell\to L)}\}_{\ell=1}^{\bar{\ell}}$, sparsity levels $\{s^{(\ell-1)}\}_{\ell=1}^{\bar{\ell}}$.

$\hat{\mathbf{A}}^{(0\to L)} = \mathbf{Y}$

$\ell = 1$

while $\ell \leq \bar{\ell}$ **do**

  $\left(\hat{\mathbf{A}}^{(\ell\to L)}, \hat{\mathbf{A}}^{(\ell-1)}\right) =$ AltMinDict$(\hat{\mathbf{A}}^{(\ell-1\to L)}, \mathbf{A}^{(\ell\to L)}(0), \epsilon_0^{(\ell\to L)}, s_{\mathbf{A}^{(\ell-1)}}, \infty)$

  $\ell = \ell + 1$

Output: $\{(\hat{\mathbf{A}}^{(\ell\to L)}, \hat{\mathbf{A}}^{(\ell-1)})\}_{\ell=1}^{\bar{\ell}}$

Theorem 2 (RIP-like property of $\mathbf{A}^{(\bar{\ell}\to L)}$): Suppose $\mathbf{y}^{(\bar{\ell}-1)}$ is $2s_{\mathbf{Y}^{(\bar{\ell}-1)}}$ sparse, then $\forall \bar{\ell} = 1, \cdots, L$

$$\prod_{\ell=\bar{\ell}}^{L}(1-\delta_{2s_{\mathbf{Y}^{(\ell-1)}}})\left|\left|\mathbf{y}^{(\bar{\ell}-1)}\right|\right|_2^2 \leq$$

$$\left|\left|\mathbf{A}^{(\bar{\ell}\to L)}\mathbf{y}^{(\bar{\ell}-1)}\right|\right|_2^2 \leq \prod_{\ell=\bar{\ell}}^{L}(1+\delta_{2s_{\mathbf{Y}^{(\ell-1)}}})\left|\left|\mathbf{y}^{(\bar{\ell}-1)}\right|\right|_2^2. \quad (24)$$

Proof 2: We proceed by induction on $\bar{\ell}$.

Base case: $\bar{\ell} = L$. The theorem is true for this case by assumption A1.

Induction: Suppose the theorem is true for $\bar{\ell}$, we will show that it holds true for $\bar{\ell}-1$. Let $\mathbf{y}^{(\bar{\ell}-2)}$ be a $2s_{\mathbf{Y}^{(\bar{\ell}-2)}}$-sparse vector

$$\left|\left|\mathbf{A}^{(\bar{\ell}-1\to L)}\mathbf{y}^{(\bar{\ell}-2)}\right|\right|_2^2 = \left|\left|\mathbf{A}^{(\bar{\ell}\to L)}\mathbf{A}^{(\bar{\ell}-1)}\mathbf{y}^{(\bar{\ell}-2)}\right|\right|_2^2. \quad (25)$$

$\mathbf{A}^{(\bar{\ell}-1)}\mathbf{y}^{(\bar{\ell}-2)}$ is a $2s_{\mathbf{Y}^{(\bar{\ell}-1)}}$-sparse vector, allowing us to apply our inductive hypothesis

$$\prod_{\ell=\bar{\ell}}^{L}(1-\delta_{2s_{\mathbf{Y}^{(\ell-1)}}})\left|\left|\mathbf{A}^{(\bar{\ell}-1)}\mathbf{y}^{(\bar{\ell}-2)}\right|\right|_2^2 \leq$$

$$\left|\left|\mathbf{A}^{(\bar{\ell}\to L)}\mathbf{A}^{(\bar{\ell}-1)}\mathbf{y}^{(\bar{\ell}-2)}\right|\right|_2^2 \leq$$

$$\prod_{\ell=\bar{\ell}}^{L}(1+\delta_{2s_{\mathbf{Y}^{(\ell-1)}}})\left|\left|\mathbf{A}^{(\bar{\ell}-1)}\mathbf{y}^{(\bar{\ell}-2)}\right|\right|_2^2. \quad (26)$$

The result follows by assumption A1 since $\mathbf{A}^{(\bar{\ell}-1)}$ satisfies the RIP of order $2s_{\mathbf{Y}^{(\bar{\ell}-2)}}$.

A direct consequence of the theorem is that $\forall \bar{\ell} = 1, \cdots, L$, the RIP constant of $\mathbf{A}^{(\bar{\ell}\to L)}$ must be smaller than or equal to $\max\left(1-\prod_{\ell=\bar{\ell}}^{L}(1-\delta_{2s_{\mathbf{Y}^{(\ell-1)}}}), \prod_{\ell=\bar{\ell}}^{L}(1+\delta_{2s_{\mathbf{Y}^{(\ell-1)}}})-1\right)$. As long as this quantity is less than 0.1, we can expect Algorithm 3 with $\bar{\ell} = L$ to succeed in recovering all L dictionaries.

V. Concentration of eigenvalues of column-sparse random matrices with dependent sub-Gaussian entries

The proof of our main result, Theorem 1, relies on random sparse matrices satisfying RIP. Here we show that a class of random sparse matrices indeed satisfies RIP.

A. Sparse random sub-Gaussian matrix model

Let $\mathbf{A} \in \mathbb{R}^{d\times r}$ be a matrix with r columns $(\mathbf{a}_j)_{j=1}^r$. Let $\mathbf{U} \in \mathbb{R}^{d\times r}$ be a binary random matrix with columns $(\mathbf{u}_j)_{j=1}^r \in \{0,1\}^d$ that are i.i.d. $s_{\mathbf{A}}$-sparse binary random vectors each obtained by selecting $s_{\mathbf{A}}$ entries from $\mathbf{u}_i$ *without* replacement, and letting $\mathbf{U}_{ij}$ be the indicator random variable of whether a given entry $j = 1,\cdots,d$ was selected. Let $\mathbf{V} \in \mathbb{R}^{d\times r}$ be a random matrix with i.i.d. entries distributed according to a zero-mean sub-Gaussian random variable V with variance 1, $|V| \leq 1$ almost surely, and sub-Gaussian norm $||V||_{\psi_2}$–we adopt the notation $||\cdot||_{\psi_2}$ from [12] to denote the sub-Gaussian norm of a random variable. We consider the following generative model for the entries of $\mathbf{A}$:

$$\mathbf{A}_{ij} = \sqrt{\frac{d}{s_{\mathbf{A}}}}\mathbf{U}_{ij}\mathbf{V}_{ij}, i = 1,\cdots,d;\ j = 1,\cdots,r. \tag{27}$$

It is not hard to verify that the random matrix $\mathbf{A}$ thus obtained is such that $E[\mathbf{a}_i\mathbf{a}_i^{\mathrm{T}}] = \mathbf{I}$. To see this we note the following properties of the generative model for $\mathbf{A}$

- $P[\mathbf{U}_{ij} = 1] = \frac{s_{\mathbf{A}}}{d}$.
- Let $j \neq j'$, $P[\mathbf{U}_{ij} = 1 \cap \mathbf{U}_{ij'} = 1] = \frac{s_{\mathbf{A}}}{d}\cdot\frac{s_{\mathbf{A}}-1}{d-1}$.
- $E[\mathbf{A}_{ij}^2] = E\left[\left(\sqrt{\frac{d}{s_{\mathbf{A}}}}\mathbf{U}_{ij}\mathbf{V}_{ij}\right)^2\right] = \sqrt{\frac{d}{s_{\mathbf{A}}}}\cdot P[\mathbf{U}_{ij}=1]\cdot E[\mathbf{V}_{ij}^2] = 1$.
- Let $j \neq j'$, $E[\mathbf{A}_{ij}A_{ij'}] = E[\mathbf{U}_{ij}U_{ij'}\mathbf{V}_{ij}\mathbf{V}_{ij'}] = E[\mathbf{U}_{ij}\mathbf{U}_{ij'}]\underbrace{E[\mathbf{V}_{ij}\mathbf{V}_{ij'}]}_{0} = 0$.
- $||\mathbf{a}_j||_2^2 = s_{\mathbf{A}}\cdot\frac{d}{s_{\mathbf{A}}} = d$ a.s, $j = 1,\cdots,r$.

Ultimately, we would like to understand the concentration behavior of the singular values of 1) $\mathbf{A}^{\mathrm{T}}$, and 2) sub-matrices of $\mathbf{A}$ that consist of a sparse subset of columns (RIP-like results). We fist recall the following result from non-asymptotic random matrix theory [12], and apply it obtain a concentration result on the singular values of the matrix $\mathbf{A}^{\mathrm{T}}$.

Theorem 3 (Restatement of Theorem 5.39 from [12] (Sub-Gaussian rows)): Let $\mathbf{W} \in \mathbb{R}^{r\times d}$ matrix whose rows $\{(\mathbf{W}^{\mathrm{T}})_j\}_{j=1}^r$ ($(\mathbf{W}^{\mathrm{T}})_j$ is the j^{th} column of $\mathbf{W}^{\mathrm{T}}$) are independent sub-Gaussian isotropic random vectors in $\mathbb{R}^d$. Then for every $t \geq 0$, with probability at least $1-\exp(-ct^2)$ one has

$$\sqrt{r}-C\sqrt{d}-t \leq \sigma_{\min}(\mathbf{W}) \leq \sigma_{\max}(\mathbf{W}) \leq \sqrt{r}+C\sqrt{d}+t. \tag{28}$$

Here, $C = C_K$, $c = c_K \geq 0$ depend only on the sub-Gaussian norm $K = \max_j \left|\left|(\mathbf{W}^{\mathrm{T}})_j\right|\right|_{\psi_2}$ of the rows.

Before we can apply the above result to $\mathbf{W} = \mathbf{A}^{\mathrm{T}}$, we need to demonstrate that the columns of $\mathbf{A}$ are sub-Gaussian random vectors, defined as follows

Definition 4 (Definition 5.22 from [12] (Sub-Gaussian random vectors)): We say that a random vector $\mathbf{x}$ in $\mathbb{R}^d$ is sub-Gaussian if the one-dimensional marginals $\langle\mathbf{x},\mathbf{z}\rangle$ are sub-Gaussian random variables for all $\mathbf{z}$ in $\mathbb{R}^d$. The sub-Gaussian norm of $\mathbf{x}$ is defined as

$$||\mathbf{x}||_{\psi_2} = \sup_{\mathbf{z}\in\mathcal{S}^{d-1}} ||\langle\mathbf{x},\mathbf{z}\rangle||_{\psi_2}. \tag{29}$$

Theorem 5 (Columns of $\mathbf{A}$ are sub-Gaussian random vectors): For every $j = 1,\cdots,r$, $\mathbf{a}_j$ is a sub-Gaussian random vector. Moreover,

$$||\mathbf{a}_j||_{\psi_2}^2 \leq \frac{d}{s_{\mathbf{A}}}C_1||V||_{\psi_2}^2, \tag{30}$$

where C_1 is a universal constant.

Proof 3: We show this by bounding $\left|\left|\sqrt{\frac{s_{\mathbf{A}}}{d}}\mathbf{a}_j\right|\right|_{\psi_2}$:

$$\left|\left|\sqrt{\frac{s_{\mathbf{A}}}{d}}\mathbf{a}_j\right|\right|_{\psi_2}\sup_{\mathbf{z}\in\mathcal{S}^{d-1}}\left|\left|\langle\sqrt{\frac{s_{\mathbf{A}}}{d}}\mathbf{a}_j,\mathbf{z}\rangle\right|\right|_{\psi_2} \tag{31}$$

$$=\sup_{\mathbf{z}\in\mathcal{S}^{d-1}}\sup_{p\geq 1}p^{-1/2}\mathbb{E}\left[\left|\sum_{i=1}^d z_i\mathbf{U}_{ij}\mathbf{V}_{ij}\right|^p\right]^{1/p} \tag{32}$$

$$=\sup_{\mathbf{z}\in\mathcal{S}^{d-1}}\sup_{p\geq 1}p^{-1/2}\left(\mathbb{E}_{(\mathbf{U}_{ij})_{i=1}^d}\left[\mathbb{E}\left[\left|\sum_{i=1}^d z_i\mathbf{U}_{ij}\mathbf{V}_{ij}\right|^p\middle|(\mathbf{U}_{ij})_{i=1}^d\right]\right]\right)^{1/p} \tag{33}$$

$$\leq\sup_{\mathbf{z}\in\mathcal{S}^{d-1}}\sup_{p\geq 1}p^{-1/2}\mathbb{E}_{(\mathbf{U}_{ij})_{i=1}^d}\left(\mathbb{E}\left[\left|\sum_{i=1}^d z_i\mathbf{U}_{ij}\mathbf{V}_{ij}\right|^p\middle|(\mathbf{U}_{ij})_{i=1}^d\right)^{1/p}\right] \tag{34}$$

$$\leq\sup_{\mathbf{z}\in\mathcal{S}^{d-1}}\mathbb{E}_{(\mathbf{U}_{ij})_{i=1}^d}\left[\sup_{p\geq 1}p^{-1/2}\left(\mathbb{E}\left|\sum_{i=1}^d z_i\mathbf{U}_{ij}\mathbf{V}_{ij}\right|^p\middle|(\mathbf{U}_{ij})_{i=1}^d\right)^{1/p}\right] \tag{35}$$

Conditioned on $(\mathbf{U}_{ij})_{i=1}^d$, $sup_{p\geq 1}p^{-1/2}\left(\mathbb{E}\left[\left|\sum_{i=1}^d z_i\mathbf{U}_{ij}\mathbf{V}_{ij}\right|^p\right]\right)^{1/p}$ is the sub-Gaussian norm of the sum of $s_{\mathbf{A}}$ independent sub-Gaussian random variables $\mathbf{V}_{1j},\cdots,\mathbf{V}_{s_{\mathbf{A}}j}$. Therefore, according to Lemma 5.9 in [12],

$$\left(\sup_{p\geq 1}p^{-1/2}\left(\mathbb{E}\left[\left|\sum_{i=1}^d z_i\mathbf{U}_{ij}\mathbf{V}_{ij}\right|^p\middle|(\mathbf{U}_{ij})_{i=1}^d\right]\right)^{1/p}\right)^2 \tag{36}$$

$$= \left|\left|\sum_{i\in\{1,\cdots,s_A\}} z_i\mathbf{V}_{ij}\right|\right|_{\psi_2}^2 \tag{37}$$

$$\leq C_1||V||_{\psi_2}^2\sum_{i\in\{1,\cdots,s_{\mathbf{A}}\}} z_i^2 \tag{38}$$

$$\leq C_1||V||_{\psi_2}^2||\mathbf{z}||_2^2. \tag{39}$$

Putting Equation 39 back into Equation 35 yields

$$||\mathbf{a}_j||_{\psi_2} \leq \sqrt{\frac{d}{s_{\mathbf{A}}}}\sup_{\mathbf{z}\in\mathcal{S}^{d-1}}\mathbb{E}_{(\mathbf{U}_{ij})_{i=1}^d}\left[\sqrt{C_1}||V||_{\psi_2}||\mathbf{z}||_2\right] \tag{40}$$

$$= \sqrt{\frac{d}{s_{\mathbf{A}}}}\sup_{\mathbf{z}\in\mathcal{S}^{d-1}}\sqrt{C_1}||V||_{\psi_2}||\mathbf{z}||_2 \tag{41}$$

$$\leq \sqrt{\frac{d}{s_{\mathbf{A}}}}\sqrt{C_1}||V||_{\psi_2}. \tag{42}$$

Remark 7: The assumption that the V_{ij}'s are i.i.d is not restrictive. The proof above effectively relies on a bound on the sub-Gaussian norm of the vector of nonzero entries of the column $\mathbf{a}_i$. If we were, for instance, to pick the nonzero entries to be spherically distributed (so that they are not independent), Remark 2 following Lemma 5.24 in [12], gives a sketch of a proof that such a vector would be a sub-Gaussian random vector (with sub-Gaussian norm bounded by a constant).

B. Concentration properties of minimum and maximum singular values of $\mathbf{A}$

We now have all of the requisites to apply Theorem 3 to the matrix $\mathbf{A}$ defined in Equation 27.

Lemma 6: Let $\mathbf{A}$ be the sparse random matrix obtained according to Equation 27. There exist universal constant c and C such that with probability at least $1-\exp\left(-ct^2\right)$)

$$\sqrt{r}(1-\delta) \leq \sigma_{\min}(\mathbf{A}^{\mathrm{T}}) \leq \sigma_{\max}(\mathbf{A}^{\mathrm{T}}) \leq \sqrt{r}(1-\delta), \delta = C\sqrt{\frac{d}{r}} + \frac{t}{\sqrt{r}}, \tag{43}$$

and

$$\left|\left|\frac{1}{r}\mathbf{A}\mathbf{A}^{\mathrm{T}} - \mathbf{I}\right|\right|_2 \leq \max(\delta, \delta^2). \tag{44}$$

Proof 4: The first part of the Lemma follows from applying the Theorem 3 to $\mathbf{A}^{\mathrm{T}}$ from Equation 27. The second part follows from Lemma 5.36 in [12] showing that the equivalence of the first and second parts.

Remark 8: According to Theorem 3, the constants c and C depend only on the sub-Gaussian norm of the columns of $\mathbf{A}$, which Theorem 5 gives a bound for. This bound highlights the main difference between the case when the $\mathbf{A}$ is a sparse matrix according to our model, as opposed to a dense matrix. Indeed, when $\mathbf{A}$ is dense, $s_{\mathbf{A}} = d$, so that the bound from Theorem 5 reduces to known bounds on the sub-Gaussian norm of a matrix with i.i.d. sub-Gaussian entries. When $\mathbf{A}$ is sparse, Theorem 5 implies that the constants c and C are larger than in the dense case, leading to looser bounds on the minimum and maximum eigenvalues of th $\mathbf{A}^{\mathrm{T}}$.

C. RIP properties of subsets of columns of $\mathbf{A}$

Consider a subset $T_0 \subset \{1, 2, \cdots, r\}$ s.t. $|T_0| = S$, and let $\mathbf{A}_{T_0}$ denote the $d \times S$ matrix corresponding to the subset of columns of $\mathbf{A}$ from Equation 27 indexed by T_0. We want an RIP-like result of $\mathbf{A}_{T_0}$, i.e. we seek a bound for the difference between the spectral norm of $\mathbf{A}_{T_0}^{\mathrm{T}}\mathbf{A}_{T_0}$ (appropriately normalized) and the identity matrix.

Definition 7 (Restatement of Definition from [12] (Restricted isometries)): A $d \times r$ matrix $\mathbf{A}$ satisfies the restricted isometry property of order $S \geq 1$ if there exists $\delta_S \geq 0$ such that the inequality

$$(1-\delta_S)\,||\mathbf{x}||_2^2 \leq ||\mathbf{A}\mathbf{x}||_2^2 \leq (1-\delta_S)\,||\mathbf{x}||_2^2 \tag{45}$$

holds for all $\mathbf{x} \in \mathbb{R}^r$ with $|Supp(\mathbf{x})| \leq S$. The smallest number $\delta_S = \delta_S(\mathbf{A})$ is called the restricted isometry constant of $\mathbf{A}$.

We first recall a result from [12] which, along with the results from the previous section, will give us the desired bound. The result applies to one of two models for $\mathbf{A}$.

Row-independent model: the rows of $\mathbf{A}$ are independent sub-Gaussian isotropic random vectors in $\mathbb{R}^r$.

Column-independent model: The columns $\mathbf{a}_j$ of $\mathbf{A}$ are independent sub-Gaussian isotropic random vectors in $\mathbb{R}^d$ with $||\mathbf{a}_j||_2 = \sqrt{d}$ a.s.

The column-independent model is the one of interest here.

Theorem 8 (Restatement of Theorem 5.65 from [12] (Sub-Gaussian restricted isometries)): Let $\mathbf{A} \in \mathbb{R}^{d\times r}$ sub-Gaussian random matrix with independent rows or columns, which follows either of the two models above. Then the normalized matrix $\bar{\mathbf{A}} = \frac{1}{\sqrt{d}}\mathbf{A}$ satisfies the following for every sparsity level $1 \leq S \leq r$ and every number $\delta \in (0,1)$:

$$\text{if } d \geq C\delta^{-2}S\log(er/S), \text{ then } \delta_S(\bar{\mathbf{A}}) \leq \delta \tag{46}$$

with probability at least $1-2\exp(-c\delta^2 d)$. Here, $C = C_K$, $c = c_K \geq 0$ depend only on the sub-Gaussian norm $K = \max_j ||\mathbf{a}_j||_{\psi_2}$ of the rows or columns of $\mathbf{A}$. We can apply this theorem to $\mathbf{A}$ from Equation 27 to conclude that, with a sufficient number of measurements d, $\bar{\mathbf{A}}$ satisfies the RIP of order S.

Lemma 9: Let $\mathbf{A}$ be the sparse random matrix obtained according to Equation 27. Then the normalized matrix $\bar{\mathbf{A}} = \frac{1}{\sqrt{d}}\mathbf{A}$ satisfies the following for every sparsity level $1 \leq S \leq r$ and every number $\delta \in (0,1)$:

$$\text{if } d \geq C\delta^{-2}S\log(er/S), \text{ then } \delta_S(\bar{\mathbf{A}}) \leq \delta \tag{47}$$

with probability at least $1-2\exp(-c\delta^2 d)$, where $C = C_K$, $c = c_K \geq 0$ depend only on the bound from Equation 42 on the sub-Gaussian norm $||\mathbf{a}_j||_{\psi_2}$ of the columns of $\mathbf{A}$.

Proof 5: By construction, $\mathbf{A}$ follows the column-independent model. In addition, we have proved in Theorem 5 that the columns of $\mathbf{A}$ are sub-Gaussian random vectors. The result follows from applying Theorem 8 to $\mathbf{A}$ from Equation 27.

Remark 9: Note that $\bar{\mathbf{A}}_{ij} = \frac{1}{\sqrt{d}}\frac{\sqrt{d}}{\sqrt{s_{\mathbf{A}}}}\mathbf{U}_{ij}\mathbf{V}_{ij} = \frac{1}{\sqrt{s_{\mathbf{A}}}}\mathbf{U}_{ij}\mathbf{V}_{ij}$.

VI. Simulations

We use simulated data to demonstrate the capability of Algorithms 2 and 3 to learn the "deep" generative model from Equation 13.

a) Simulations.: As both algorithms are computationally demanding, in that they require the solutions to many convex optimization problems, we consider the case when $L = 2$. We used the simulation studies from [7] to guide our choice of parameters

1) We chose $\mathbf{A}^{(2)}$ to be of size 100×200, i.e. $d_2 = 100$ and $r_2 = 200$. We chose its entries to be i.i.d. $\mathcal{N}(0, \frac{1}{d_2})$.
2) We chose $\mathbf{A}^{(1)}$ to be of size 200×800, i.e. $d_1 = 200$ and $r_1 = 800$. We chose its entries according to the sparse random matrix model from Equation 27, letting $\{\mathbf{V}_{ij}\}_{i=1,j=1}^{200,800}$ be i.i.d. Rademacher random variables ($+/-1$ with equal probability). Each column of $\mathbf{A}^{(1)}$ has sparsity level $s^{(1)} = 3$.
3) We chose $n = 6400$, so that $\mathbf{X}$ is of size 800×6400. Similar to [7], we chose the non-zero entries of $\mathbf{X}$ to be i.i.d $\mathrm{U}([-2,-1] \cup [1,2])$, and pick $s = 3$ (sparsity level of each column of $\mathbf{X}$).
4) For a given matrix $\mathbf{A}$ and its estimate $\hat{\mathbf{A}}$, we use the following error metric to assess the success of the algorithm at learning $\mathbf{A}$

$$\mathrm{err}(\hat{\mathbf{A}}, \mathbf{A}) = \max_i \sqrt{1 - \frac{\langle \mathbf{A}_i, \hat{\mathbf{A}}_i \rangle}{\|\mathbf{A}_i\|_2^2 \|\hat{\mathbf{A}}_i\|_2^2}} \tag{48}$$

We declare the algorithm as successful when this error is on the order of 10^{-5}.

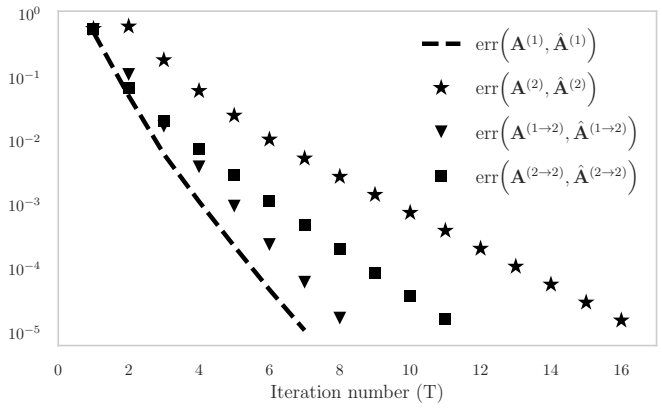

Fig. 3. Semi-log plot of the error between the true dictionaries and the recovered ones as a function of the number of iterations of Algorithm 1. The figure shows that the recovered dictionaries converge to the true ones linearly (exponentially). We refer the reader to the main text for additional interpretations of the simulations.

N.B.: Note the fact that, n is much smaller than r_1^2 and r_2^2, and yet the simulations will demonstrate that the dictionaries can be learned exactly. As in [7], this highlights the fact that the term r_ℓ^2 in the complexity bound is an artifact of the proof techniques.

b) Initialization.: To initialize Algorithms 2 and 3, we need to initialize $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(2\to2)}$. For a generic matrix $\mathbf{A}$ of size $d \times r$, following the simulation studies from [7], we let $\mathbf{A}(0) = \mathbf{A} + \mathbf{Z}$, where entries of $\mathbf{Z}$ are i.i.d. $0.5 \cdot \mathcal{N}(0, \frac{1}{d})$.

c) Implementation.: We implemented Algorithms 2 and 3 using Python programming language. We used `cvxpy` [21] with the `MOSEK` [22] solver to find the solutions to the optimization problems from the inner loop of the alternating-minimization procedure (Algorithm 1). The authors in [7] use the GradeS [23] algorithm for this inner loop because it is faster. In our experience, cvxpy was much more stable numerically. The inner loop of Algorithm 1 is embarrassingly parallelizable. Therefore, we also implemented a distributed version of our algorithm using the Python module `dask` [24]. The code is hosted on `bitbucket`. The author is happy to make it available upon request[1].

d) Results.: Figure 3 depicts the error between the true dictionaries and the ones recovered using Algorithm 2 or Algorithm 3. We obtained $\hat{\mathbf{A}}^{(1)}$ and $\hat{\mathbf{A}}^{(2)}$ using Algorithm 2, while $\hat{\mathbf{A}}^{(1\to2)}$ (product matrix) and $\hat{\mathbf{A}}^{(2)}$ were obtained using Algorithm 3. For Algorithm 3, we chose to plot the recovery error for $\hat{\mathbf{A}}^{(1\to2)}$ to demonstrate its ability to recover the product matrix. The difference in the number of iterations comes from our choice of termination criterion at a value on the order of 10^{-5}.

The figure demonstrates the linear (exponentially) converge of the recovered dictionary to the true ones. The error curve for $\hat{\mathbf{A}}^{(2)}$ is consistent with the simulation studies from [7]. In this example ($L = 2$), $\mathbf{A}^{(2\to2)}$ and $\mathbf{A}^{(2)}$ are the same matrix. The error rate appears to be faster for the former than the latter. We hypothesize that this is because the sparsity-level parameter for $\mathbf{A}^{(2\to2)}$ is $s^{(1)} = 3$, while that for $\mathbf{A}^{(2)}$ is $s \cdot s^{(1)} = 9$. The amount of data required in Theorem 1 from [7] scales linearly with the sparsity level. The error rate for $\mathbf{A}^{(1)}$ is much faster than all of the others, i.e. it would appear that $\mathbf{A}^{(1)}$ is easier to learn. This should not be surprising as $\mathbf{A}^{(1)}$ is a matrix that is very sparse (3-sparse in fact). Therefore, its columns have far fewer that $d_1 = 100$ degrees of freedom. The proof techniques utilized in [7] and which we rely upon do not explicitly take into account the sparsity of $\mathbf{A}^{(1)}$. All that the theorem gives is an upper bound on the rate of convergence, without an accompanying lower bound. Studying the effect of sparsity of the matrix of interest in dictionary learning would be an interesting area of further inquiry. Overall, our simulations demonstrate the ability of Algorithms 2 and 3 to recover the deep generative model from Equation 13. We would like to stress that Figure 3 required on the order of 5 hours to generate on a PC with 8 GB of RAM and 4 cores with 2 threads each.

[1]https://bitbucket.org/demba/ds2p/src/master/

VII. Discussion

We have provided insights as to the complexity of learning deep neural networks by building a link between deep recurrent auto-encoders [10] and classical dictionary learning theory [7]. We used this link to develop a deep version of the classical sparse coding model from dictionary learning. Starting with a sparse code, a cascade of linear transformations are applied in succession to generate an observations. Each transformation in the cascade is constrained to have sparse columns, except for the last one. We developed a sequential alternating-minimization algorithm to learn this deep generative model from observations and proved that, under assumptions stated in detail, the algorithm is able to learn the underlying transformations in the cascade that generated the observations. The computational complexity of the algorithm is a function of the dimension of the inputs of each of the transformations in the cascade, as well as the respective sparsity level of these inputs. In particular, the complexity is $\mathcal{O}\left(\max_\ell \max(r_\ell^2, r_\ell s_{\mathbf{Y}^{(\ell-1)}})\right)$, where r_ℓ is the size of the embedding at ℓ^{th} layer and $s_{\mathbf{Y}^{(\ell-1)}}$ the number active neurons at that layer. The simulations (Section VI) suggest that the term r_ℓ^2 above is an artifact of the proof techniques we rely on to arrive at our main result. That is, the learning complexity depends on the maximum, across layers, of the product of the number of active neurons and the embedding dimension. The proof relies on a certain family of sparse random matrices satisfying the RIP. We study the properties of this family of matrices using results from non-asymptotic random matrix theory.

In future work, we will study the case when the matrices in the cascade satisfy the mutual-incoherence property rather than RIP. In dictionary learning, good initialization procedures are known for the mutually-incoherent case [7]. As they stand, our results do not provide a good guarantees for a case when we know good initialization procedures. It will be interesting to study the coherence properties of sparse random matrices. As the deep dictionary learning algorithm relies on the solution of convex optimization problems, we will explore distributed implementations of these algorithms, as well as ones on GPUs.

One impressive fact about deep learning is the GPU-based infrastructure that has been developed to train virtually any kind of network imaginable. It would of great interest to develop a similar infrastructure, backed by TensorFlow [25] for sparsity-regularized inverse problems. We also plan to train shallow and deep versions of the recurrent sparse auto-encoders to demonstrate that they can solve the shallow and deep dictionary learning problems respectively. We have already demonstrated this in [11] for the convolutional case.

VIII. Appendix

A. Derivation of soft-thresholding operator

The solution $\hat{x}$ must satisfy

$$y = x + \lambda \partial |x|. \tag{49}$$

Case $y > \lambda$: The idea is to show that, if $y > \lambda$, then necessarily $x > 0$. Suppose for a contradiction that $x < 0$, then Equation 49 yields $y = x - \lambda < 0$, a contradiction. Similarly, if $x = 0$, then $y = \lambda \partial x$, where the sub-gradient $\partial x \in [-1, 1]$, a contradiction since $y > \lambda$. Therefore, if $y > lambda$, Equation 49 yields $\hat{x} = y - \lambda$.

Case $y < \lambda$: This case proceeds similarly as above.

Case $|y| \leq \lambda$: Suppose, for contradiction, that $x > 0$, then Equation 49 yields $y > \lambda$. Similarly, if $x < 0$, we obtain $y < -\lambda$. Therefore, if $|y| \leq \lambda$, $\hat{x} = 0$.

Together, the three cases above yield the soft-thresholding function of Equation 2 as the solution to Equation 1.

References

[1] V. Papyan, Y. Romano, and M. Elad, "Convolutional neural networks analyzed via convolutional sparse coding," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2887–2938, 2017.

[2] A. B. Patel, T. Nguyen, and R. G. Baraniuk, "A probabilistic theory of deep learning," *arXiv preprint arXiv:1504.00641*, 2015.

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

[4] J. Sulam, V. Papyan, Y. Romano, and M. Elad, "Multilayer convolutional sparse modeling: Pursuit and dictionary learning," *IEEE Transactions on Signal Processing*, vol. 66, no. 15, pp. 4090–4104, 2018.

[5] J. C. Ye, Y. Han, and E. Cha, "Deep convolutional framelets: A general deep learning framework for inverse problems," *SIAM Journal on Imaging Sciences*, vol. 11, no. 2, pp. 991–1048, 2018.

[6] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *Information Theory Workshop (ITW), 2015 IEEE*. IEEE, 2015, pp. 1–5.

[7] A. Agarwal, A. Anandkumar, P. Jain, and P. Netrapalli, "Learning Sparsely Used Overcomplete Dictionaries via Alternating Minimization," *SIAM Journal on Optimization*, vol. 26, no. 4, pp. 2775–2799, 2016.

[8] A. Aberdam, J. Sulam, and M. Elad, "Multi-layer sparse coding: The holistic way," *SIAM Journal on Mathematics of Data Science*, vol. 1, no. 1, pp. 46–77, 2019.

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

[10] J. T. Rolfe and Y. LeCun, "Discriminative recurrent sparse auto-encoders," *arXiv preprint arXiv:1301.3775*, 2013.

[11] B. Tolooshams, S. Dey, and D. Ba, "Scalable convolutional dictionary learning with constrained recurrent sparse auto-encoders," in *Machine Learning for Signal Processing (MLSP), 2018 IEEE 28th International Workshop on*. IEEE, 2018.

[12] R. Vershynin, "Introduction to the non-asymptotic analysis of random matrices," *arXiv preprint arXiv:1011.3027*, 2010.

[13] H. Sreter and R. Giryes, "Learned convolutional sparse coding," *arXiv preprint arXiv:1711.00328*, 2017.

[14] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[15] A. Fawzi, M. Davies, and P. Frossard, "Dictionary learning for fast classification based on soft-thresholding," *International Journal of Computer Vision*, vol. 114, no. 2-3, pp. 306–321, 2015.

[16] 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.

[17] B. Tolooshams, S. Dey, and D. Ba, "Deep residual auto-encoders for expectation maximization-based dictionary learning," pp. 1–13, 2019, arXiv:1904.08827.

[18] S. Theodoridis, *Machine learning: a Bayesian and optimization perspective*. Academic Press, 2015.

[19] L. Le Magoarou and R. Gribonval, "Chasing butterflies: In search of efficient dictionaries," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 3287–3291.

[20] 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.

[21] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[22] "The MOSEK optimization software." [Online]. Available: http://www.mosek.com/

[23] R. Garg and R. Khandekar, "Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 337–344.

[24] Dask Development Team, *Dask: Library for dynamic task scheduling*, 2016. [Online]. Available: http://dask.pydata.org

[25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.