Deep Component Analysis via Alternating Direction Neural Networks

Calvin Murdock, Ming-Fang Chang, and Simon Lucey

Carnegie Mellon University
{cmurdock,mingfanc,slucey}@cs.cmu.edu

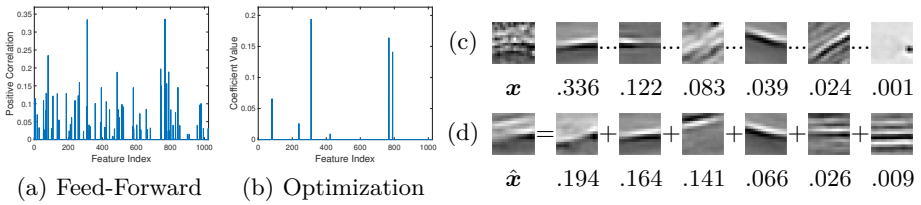
Abstract. Despite a lack of theoretical understanding, deep neural networks have achieved unparalleled performance in a wide range of applications. On the other hand, shallow representation learning with component analysis is associated with rich intuition and theory, but smaller capacity often limits its usefulness. To bridge this gap, we introduce Deep Component Analysis (DeepCA), an expressive multilayer model formulation that enforces hierarchical structure through constraints on latent variables in each layer. For inference, we propose a differentiable optimization algorithm implemented using recurrent Alternating Direction Neural Networks (ADNNs) that enable parameter learning using standard backpropagation. By interpreting feed-forward networks as single-iteration approximations of inference in our model, we provide both a novel perspective for understanding them and a practical technique for constraining predictions with prior knowledge. Experimentally, we demonstrate performance improvements on a variety of tasks, including single-image depth prediction with sparse output constraints.

Keywords: Component Analysis · Deep Learning · Constraints

1 Introduction

Deep convolutional neural networks have achieved remarkable success in the field of computer vision. While far from new [24], the increasing availability of extremely large, labeled datasets along with modern advances in computation with specialized hardware have resulted in state-of-the-art performance in many problems, including essentially all visual learning tasks. Examples include image classification [19], object detection [20], and semantic segmentation [10]. Despite a rich history of practical and theoretical insights about these problems, modern deep learning techniques typically rely on task-agnostic models and poorly-understood heuristics. However, recent work [6,28,43] has shown that specialized architectures incorporating classical domain knowledge can increase parameter efficiency, relax training data requirements, and improve performance.

Prior to the advent of modern deep learning, optimization-based methods like component analysis and sparse coding dominated the field of representation learning. These techniques use structured matrix factorization to decompose data into linear combinations of shared components. Latent representations are



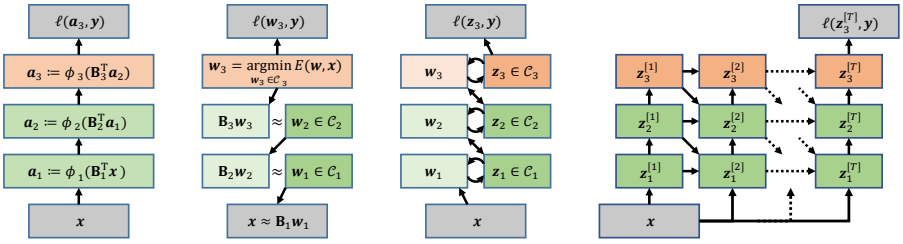
(a) Feed-Forward (b) Optimization

Fig. 1: An example of the "explaining away" conditional dependence provided by optimization-based inference. Sparse representations constructed by feed-forward nonnegative soft thresholding (a) have many more non-zero elements due to redundancy and spurious activations (c). On the other hand, sparse representations found by ℓ_1-penalized, nonnegative least-squares optimization (b) yield a more parsimonious set of components (d) that optimally reconstruct approximations of the data.

inferred by minimizing reconstruction error subject to constraints that enforce properties like uniqueness and interpretability. Importantly, unlike feed-forward alternatives that construct representations in closed-form via independent feature detectors, this iterative optimization-based approach naturally introduces conditional dependence between features in order to best explain data, a useful phenomenon commonly referred to as "explaining away" within the context of graphical models [4]. An example of this effect is shown in Fig. 1, which compares sparse representations constructed using feed-forward soft thresholding with those given by optimization-based inference with an ℓ_1 penalty. While many components in an overcomplete set of features may have high-correlation with an image, constrained optimization introduces competition between components resulting in more parsimonious representations.

Component analysis methods are also often guided by intuitive goals of incorporating prior knowledge into learned representations. For example, statistical independence allows for the separation of signals into distinct generative sources [22], non-negativity leads to parts-based decompositions of objects [25], and sparsity gives rise to locality and frequency selectivity [35]. Due to the difficulty of enforcing intuitive constraints like these with feed-forward computations, deep learning architectures are instead often motivated by distantly-related biological systems [39] or poorly-understand internal mechanisms such as covariate shift [21] and gradient flow [17]. Furthermore, while a theoretical understanding of deep learning is fundamentally lacking [47], even non-convex formulations of matrix factorization are often associated with guarantees of convergence [2], generalization [29], uniqueness [13], and even global optimality [16].

In order to unify the intuitive and theoretical insights of component analysis with the practical advances made possible through deep learning, we introduce the framework of Deep Component Analysis (DeepCA). This novel model formulation can be interpreted as a multilayer extension of traditional component analysis in which multiple layers are learned jointly with intuitive constraints intended to encode structure and prior knowledge. DeepCA can also be motivated



(a) Feed-Forward (b) DeepCA (c) ADNN (d) Unrolled ADNN

Fig. 2: A comparison between feed-forward neural networks and the proposed deep component analysis (DeepCA) model. Standard deep networks construct learned representations as feed-forward compositions of nonlinear functions (a). DeepCA instead treats them as unknown latent variables to be inferred by constrained optimization (b). To accomplish this, we propose a differentiable inference algorithm that can be expressed as an Alternating Direction Neural Network (ADNN) (c), a recurrent generalization of feed-forward networks that can be unrolled to a fixed number of iterations for learning via backpropagation (d).

from the perspective of deep neural networks by relaxing the implicit assumption that the input to a layer is constrained to be the output of the previous layer, as shown in Eq. 1 below. In a feed-forward network (left), the output of layer j, denoted $\boldsymbol{a}_j$, is given in closed-form as a nonlinear function of $\boldsymbol{a}_{j-1}$. DeepCA (right) instead takes a generative approach in which the latent variables $\boldsymbol{w}_j$ associated with layer j are *inferred* to optimally reconstruct $\boldsymbol{w}_{j-1}$ as a linear combination of learned components subject to some constraints $\mathcal{C}_j$.

$$\text{Feed-Forward: } \boldsymbol{a}_j = \phi(\mathbf{B}_j^{\mathsf{T}}\boldsymbol{a}_{j-1}) \implies \text{DeepCA: } \mathbf{B}_j\boldsymbol{w}_j \approx \boldsymbol{w}_{j-1} \text{ s.t. } \boldsymbol{w}_j \in \mathcal{C}_j \quad (1)$$

From this perspective, intermediate network "activations" cannot be found in closed-form but instead require explicitly solving an optimization problem. While a variety of different techniques could be used for performing this inference, we propose the Alternating Direction Method of Multipliers (ADMM) [5]. Importantly, we demonstrate that after proper initialization, a single iteration of this algorithm is equivalent to a pass through an associated feed-forward neural network with nonlinear activation functions interpreted as proximal operators corresponding to penalties or constraints on the coefficients. The full inference procedure can thus be implemented using Alternating Direction Neural Networks (ADNN), recurrent generalizations of feed-forward networks that allow for parameter learning using backpropagation. A comparison between standard neural networks and DeepCA is shown in Fig. 2. Experimentally, we demonstrate that recurrent passes through convolutional neural networks enable better sparsity control resulting in consistent performance improvements in both supervised and unsupervised tasks without introducing any additional parameters.

More importantly, DeepCA also allows for other constraints that would be impossible to effectively enforce with a single feed-forward pass through a net-

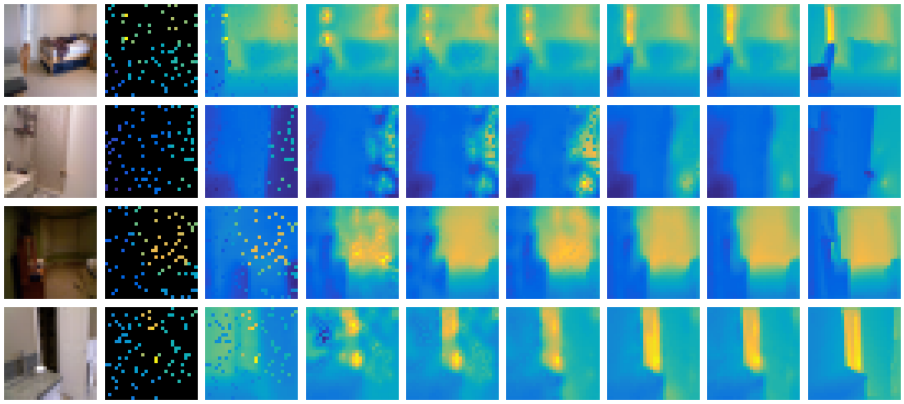


Fig. 3: A demonstration of DeepCA applied to single-image depth prediction using images concatenated with sparse sets of known depth values as input. Baseline feed-forward networks are not guaranteed to produce outputs that are consistent with the given depth values. In comparison, ADNNs with an increasing number of iterations ($T > 1$) learn to satisfy the sparse output constraints, resolving ambiguities for more accurate predictions without unrealistic discontinuities.

work. As an example, we consider the task of single-image depth prediction, a difficult problem due to the absence of three-dimensional information such as scale and perspective. In many practical scenarios, however, sparse sets of known depth outputs are available for resolving these ambiguities to improve accuracy. This prior knowledge can come from additional sensor modalities like LIDAR or from other 3D reconstruction algorithms that provide sparse depths around textured image regions. Feed-forward networks have been proposed for this problem by concatenating known depth values as an additional input channel [30]. However, while this provides useful context, predictions are not guaranteed to be consistent with the given outputs leading to unrealistic discontinuities. In comparison, DeepCA enforces the constraints by treating predictions as unknown latent variables. Some examples of how this behavior can resolve ambiguities are shown in Fig. 3 where ADNNs with additional iterations learn to propagate information from the given depth values to produce more accurate predictions.

In addition to practical advantages, our model also provides a novel perspective for conceptualizing deep learning techniques. Specifically, rectified linear unit (ReLU) activation functions [14], which are ubiquitous among many state-of-the-art models in a variety of applications, are equivalent to ℓ_1-penalized, sparse projections onto non-negativity constraints. Alongside the interpretation of feed-forward networks as single-iteration approximations of reconstruction objective functions, this suggests new insights towards better understanding the effectiveness of deep neural networks from the perspective of sparse approximation theory.

2 Background and Related Work

In order to motivate our approach, we first provide some background on matrix factorization, component analysis, and deep neural networks.

Component analysis is a common approach for shallow representation learning that approximately decomposes data $\boldsymbol{x} \in \mathbb{R}^d$ into linear combinations of learned components in $\mathbf{B} \in \mathbb{R}^{d \times k}$. This is typically accomplished by minimizing reconstruction error subject to constraints $\mathcal{C}$ on the coefficients that serve to resolve ambiguity or incorporate prior knowledge such as low-rank structure or sparsity. Some examples include Principal Component Analysis (PCA) [44] for dimensionality reduction and sparse dictionary learning [2] which accommodates overcomplete representations by enforcing sparsity.

While the problem of learning both the components and coefficients is typically non-convex, its structure naturally suggests simple alternating minimization strategies that are often guaranteed to converge [45]. However, these techniques typically require careful initialization in order to avoid poor local minima. This differs from backpropagation with stochastic gradient descent wherein random initializations are often sufficient. Alternatively, we consider a nested optimization problem that separates learning from inference:

$$\arg\min_{\mathbf{B}} \sum_{i=1}^{n} \tfrac{1}{2} \|\boldsymbol{x}^{(i)} - \mathbf{B}\boldsymbol{f}(\boldsymbol{x}^{(i)})\|_2^2 \quad \text{s.t. } \boldsymbol{f}(\boldsymbol{x}) = \arg\min_{\boldsymbol{w} \in \mathcal{C}} \tfrac{1}{2} \|\boldsymbol{x} - \mathbf{B}\boldsymbol{w}\|_2^2 \tag{2}$$

Here, the inference function $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^k$ is a potentially nonlinear transformation that maps data to their corresponding representations by solving an optimization problem with fixed parameters. For unconstrained PCA with orthogonal components, this inference problem has a simple closed-form solution given by the linear transformation $\boldsymbol{f}^{\text{PCA}}(\boldsymbol{x}) = \mathbf{B}^\mathsf{T}\boldsymbol{x}$. Substituting this into Eq. 2 results in a linear autoencoder with one hidden layer and tied weights, which has the same unique global minimum but can be trained by backpropagation [1].

With general constraints, inference typically cannot be accomplished in closed form but must instead rely on an iterative optimization algorithm. However, if this algorithm is composed as a finite sequence of differentiable transformations, then the model parameters can still be learned in the same way by backpropagating gradients through the steps of the inference algorithm. We extend this idea by representing an algorithm for inference in our DeepCA model as a recurrent neural network unrolled to a fixed number of iterations.

Recently, deep neural networks have emerged as the preferred alternative to component analysis for representation learning of visual data. Their ability to jointly learn multiple layers of abstraction has been shown to allow for encoding increasingly complex features such as textures and object parts [26]. Unlike with component analysis, inference is given in closed-form by design. Specifically, a representation is constructed by passing an image $\boldsymbol{x}$ through the composition of alternating linear transformations with parameters $\mathbf{B}_j$ and $\boldsymbol{b}_j$ and fixed nonlinear activation functions ϕ_j for layers $j = 1, \ldots, l$ as follows:

$$\boldsymbol{f}^{\text{DNN}}(\boldsymbol{x}) = \phi_l\big(\mathbf{B}_l^\mathsf{T} \cdots \phi_2(\mathbf{B}_2^\mathsf{T}(\phi_1(\mathbf{B}_1^\mathsf{T}\boldsymbol{x} - \boldsymbol{b}_1) - \boldsymbol{b}_2) \cdots - \boldsymbol{b}_l\big) \tag{3}$$

Instead of considering the forward pass of a neural network as an arbitrary nonlinear function, we interpret it as a method for approximate inference in an unsupervised generative model. This follows from previous work which has shown it to be equivalent to bottom-up inference in a probabilistic graphical model [38] or approximate inference in a multi-layer convolutional sparse coding model [36,40]. However, these approaches have limited practical applicability due to their reliance on careful hyperparameter selection and specialized optimization algorithms. While ADMM has been proposed as a gradient-free alternative to backpropagation for parameter learning [42], we use it only for inference which allows for simpler learning using backpropagation with arbitrary loss functions.

Aside from ADNNs, recurrent feedback has been proposed in other models to improve performance by iteratively refining predictions, especially for applications such as human pose estimation or image segmentation where outputs have complex correlation patterns [3, 7, 27]. While some methods also implement feedback by directly unrolling iterative algorithms, they are often geared towards specific applications such as graphical model inference [11, 18], solving under-determined inverse problems [12, 15, 41], or image alignment [28]. Similar to [46], DeepCA provides a more general mechanism for low-level feedback in arbitrary neural networks, but it is motivated by the more interpretable goal of minimizing reconstruction error subject to constraints on network activations.

3 Deep Component Analysis

Deep Component Analysis generalizes the shallow inference objective function in Eq. 2 by introducing additional layers $j = 1, \ldots, l$ with parameters $\mathbf{B}_j \in \mathbb{R}^{p_{j-1} \times p_j}$. Optimal DeepCA inference can then be accomplished by solving:

$$\boldsymbol{f}^*(\boldsymbol{x}) = \underset{\{\boldsymbol{w}_j\}}{\arg\min} \sum_{j=1}^{l} \tfrac{1}{2} \left\| \boldsymbol{w}_{j-1} - \mathbf{B}_j \boldsymbol{w}_j \right\|_2^2 + \Phi_j(\boldsymbol{w}_j) \quad \text{s.t. } \boldsymbol{w}_0 = \boldsymbol{x} \tag{4}$$

Instead of constraint sets $\mathcal{C}_j$, we use penalty functions $\Phi_j : \mathbb{R}^{p_j} \rightarrow \mathbb{R}$ to enable more general priors. Note that hard constraints can still be represented by indicator functions $I(\boldsymbol{w}_j \in \mathcal{C}_j)$ that equal zero if $\boldsymbol{w}_j \in \mathcal{C}_j$ and infinity otherwise. While we use pre-multiplication with a weight matrix $\mathbf{B}_j$ to simplify notation, our method also supports any linear transformation by replacing transposed weight matrix multiplication with its corresponding adjoint operator. For example, the adjoint of convolution is transposed convolution, a popular approach to upsampling in convolutional networks [34].

If the penalty functions are convex, this problem is also convex and can be solved using standard optimization methods. While this appears to differ substantially from inference in deep neural networks, we later show that it can be seen as a generalization of the feed-forward inference function in Eq. 3. In the remainder of this section, we justify the use of penalty functions in lieu of explicit nonlinear activation functions by drawing connections between non-negative ℓ_1 regularization and ReLU activation functions. We then propose a

general algorithm for solving Eq. 4 for the unknown coefficients and formalize the relationship between DeepCA and traditional deep neural networks, which enables parameter learning via backpropagation.

3.1 From Activation Functions to Constraints

Before introducing our inference algorithm, we first discuss the connection between penalties and their nonlinear proximal operators, which forms the basis of the close relationship between DeepCA and traditional neural networks. Ubiquitous within the field of convex optimization, proximal algorithms [37] are methods for solving nonsmooth optimization problems. Essentially, these techniques work by breaking a problem down into a sequence of smaller problems that can often be solved in closed-form by proximal operators $\phi : \mathbb{R}^d \to \mathbb{R}^d$ associated with penalty functions $\Phi : \mathbb{R}^d \to \mathbb{R}$ given by the solution to the following optimization problem, which generalizes projection onto a constraint set:

$$\phi(\boldsymbol{w}) = \underset{\boldsymbol{w}'}{\arg\min}\, \tfrac{1}{2} \left\|\boldsymbol{w} - \boldsymbol{w}'\right\|_2^2 + \Phi(\boldsymbol{w}') \tag{5}$$

Within the framework of DeepCA, we interpret nonlinear activation functions in deep networks as proximal operators associated with convex penalties on latent coefficients in each layer. While this connection cannot be used to generalize all nonlinearities, many can naturally be interpreted as proximal operators. For example, the sparsemax activation function is a projection onto the probability simplex [31]. Similarly, the ReLU activation function is a projection onto the nonnegative orthant. When used with a negative bias $\boldsymbol{b}$, it is equivalent to nonnegative soft-thresholding $\mathcal{S}_{\boldsymbol{b}}^+$, the proximal operator associated with nonnegative ℓ_1 regularization:

$$\Phi^{\ell_1^+}(\boldsymbol{w}) = I(\boldsymbol{w} \geq 0) + \textstyle\sum_p b_p \left|w_p\right| \quad \Longrightarrow \quad \phi^{\ell_1^+}(\boldsymbol{w}) = \mathcal{S}_{\boldsymbol{b}}^+(\boldsymbol{w}) = \mathrm{ReLU}(\boldsymbol{w} - \boldsymbol{b}) \tag{6}$$

While this equivalence has been noted previously as a means to theoretically analyze convolutional neural networks [36], DeepCA supports optimizing the bias $\boldsymbol{b}$ as an ℓ_1 penalty hyperparameter via backpropagation for adaptive regularization, which results in better control of representation sparsity.

In addition to standard activation functions, DeepCA also allows for enforcing additional constraints that encode prior knowledge if their corresponding proximal operators can be computed efficiently. For our example of single-image depth prediction with a sparse set of known outputs $\boldsymbol{y}$ provided as prior knowledge, the penalty function on the final output $\boldsymbol{w}_l$ is $\boldsymbol{\Phi}_l(\boldsymbol{w}_l) = I(\mathbf{S}\boldsymbol{w}_l = \boldsymbol{y})$ where the selector matrix $\mathbf{S}$ extracts the indices corresponding to the known outputs in $\boldsymbol{y}$. The associated proximal operator $\boldsymbol{\phi}_l$ projects onto this constraint set by simply correcting the outputs that disagree with the known constraints. Note that this would not be an effective output nonlinearity in a feed-forward network because, while the constraints would be technically satisfied, there is nothing to enforce that they be consistent with neighboring predictions leading to unrealistic discontinuities. In contrast, DeepCA inference minimizes the reconstruction error at each layer subject to these constraints by taking multiple iterations through the network.

3.2 Inference by the Alternating Direction Method of Multipliers

With the model parameters fixed, we solve our DeepCA inference problem using the Alternating Direction Method of Multipliers (ADMM), a general optimization technique that has been successfully used in a wide variety of applications [5]. To derive the algorithm applied to our problem, we first modify our objective function by introducing auxiliary variables $\boldsymbol{z}_j$ that we constrain to be equal to the unknown coefficients $\boldsymbol{w}_j$, as shown in Eq. 7 below.

$$\underset{\{\boldsymbol{w}_j, \boldsymbol{z}_j\}}{\arg\min} \sum_{j=1}^{l} \tfrac{1}{2} \left\| \boldsymbol{z}_{j-1} - \mathbf{B}_j \boldsymbol{w}_j \right\|_2^2 + \Phi_j(\boldsymbol{z}_j) \quad \text{s.t. } \boldsymbol{w}_0 = \boldsymbol{x},\ \forall j : \boldsymbol{w}_j = \boldsymbol{z}_j \tag{7}$$

From this, we construct the augmented Lagrangian $\mathcal{L}_\rho$ with dual variables $\boldsymbol{\lambda}$ and a quadratic penalty hyperparameter $\rho = 1$:

$$\mathcal{L}_\rho = \sum_{j=1}^{l} \tfrac{1}{2} \left\| \boldsymbol{z}_{j-1} - \mathbf{B}_j \boldsymbol{w}_j \right\|_2^2 + \Phi_j(\boldsymbol{z}_j) + \boldsymbol{\lambda}_j^\mathsf{T}(\boldsymbol{w}_j - \boldsymbol{z}_j) + \tfrac{\rho}{2} \left\| \boldsymbol{w}_j - \boldsymbol{z}_j \right\|_2^2 \tag{8}$$

The ADMM algorithm then proceeds by iteratively minimizing $\mathcal{L}_\rho$ with respect to each set of variables with the others fixed, breaking our full inference problem into smaller pieces that can each be solved in closed form. Due to the decoupling of layers in our DeepCA model, the latent activations can be updated incrementally by stepping through each layer in succession, resulting in faster convergence and computations that mirror the computational structure of deep neural networks. With only one layer, our objective function is separable and so this algorithm reduces to the classical two-block ADMM, which has extensive convergence guarantees [5]. For multiple layers, however, our problem becomes non-separable and so this algorithm can be seen as an instance of cyclical multi-block ADMM with quadratic coupling terms. While our experiments have shown this approach to be effective in our applications, theoretical analysis of its convergence properties is still an active area of research [9].

A single iteration of our algorithm proceeds by taking the following steps for all layers $j = 1, \ldots, l$ in succession:

1.) First, $\boldsymbol{w}_j$ is updated by minimizing the Lagrangian after fixing the associated auxiliary variable $\boldsymbol{z}_j$ from the previous iteration along with that of the previous layer $\boldsymbol{z}_{j-1}$ from the current iteration:

$$\begin{aligned} \boldsymbol{w}_j^{[t+1]} &:= \underset{\boldsymbol{w}_j}{\arg\min}\, \mathcal{L}_\rho(\boldsymbol{w}_j, \boldsymbol{z}_{j-1}^{[t+1]}, \boldsymbol{z}_j^{[t]}, \boldsymbol{\lambda}_j^{[t]}) \\ &= \left(\mathbf{B}_j^\mathsf{T}\mathbf{B}_j + \rho\mathbf{I}\right)^{-1} \left(\mathbf{B}_j^\mathsf{T}\boldsymbol{z}_{j-1}^{[t+1]} + \rho\boldsymbol{z}_j^{[t]} - \boldsymbol{\lambda}_j^{[t]}\right) \end{aligned} \tag{9}$$

This is an unconstrained linear least squares problem, so it's solution is given by solving a linear system of equations.

2.) Next, $\boldsymbol{z}_j$ is updated by fixing the newly updated $\boldsymbol{w}_j$ along with the next layer's coefficients $\boldsymbol{w}_{j+1}$ from the previous iteration:

$$\begin{aligned}\boldsymbol{z}_j^{[t+1]} &:= \arg\min_{\boldsymbol{z}_j} \mathcal{L}_\rho(\boldsymbol{w}_j^{[t+1]}, \boldsymbol{w}_{j+1}^{[t]}, \boldsymbol{z}_j, \boldsymbol{\lambda}_j^{[t]}) \\ &= \phi_j\big(\tfrac{1}{\rho+1}\mathbf{B}_{j+1}\boldsymbol{w}_{j+1}^{[t]} + \tfrac{\rho}{\rho+1}(\boldsymbol{w}_j^{[t+1]} + \tfrac{1}{\rho}\boldsymbol{\lambda}_j^{[t]})\big) \\ \boldsymbol{z}_l^{[t+1]} &:= \phi_j\big(\boldsymbol{w}_j^{[t+1]} + \tfrac{1}{\rho}\boldsymbol{\lambda}_j^{[t]}\big)\end{aligned} \tag{10}$$

This is the proximal minimization problem from Eq. 5, so its solution is given in closed form via the proximal operator $\boldsymbol{\phi}_j$ associated with the penalty function $\boldsymbol{\Phi}_j$. Note that for $j \neq l$, its argument is a convex combination of the current coefficients $\boldsymbol{w}_j$ and feedback that enforces consistency with the next layer.

3.) Finally, the dual variables $\boldsymbol{\lambda}_j$ are updated with the constraint violations scaled by the penalty parameter ρ.

$$\boldsymbol{\lambda}_j^{[t+1]} := \boldsymbol{\lambda}_j^{[t]} + \rho(\boldsymbol{w}_j^{[t+1]} - \boldsymbol{z}_j^{[t+1]}) \tag{11}$$

This process is then repeated until convergence. Though not available as a closed-form expression, in the next section we demonstrate how this algorithm can be posed as a recurrent generalization of a feed-forward neural network.

4 Alternating Direction Neural Networks

Our inference algorithm essentially follows the same pattern as a deep neural network: for each layer, a learned linear transformation is applied to the previous output followed by a fixed nonlinear function. Building upon this observation, we implement it using a recurrent network with standard layers, thus allowing the model parameters to be learned using backpropagation.

Recall that the $\boldsymbol{w}_j$ update in Eq. 9 requires solving a linear system of equations. While differentiable, this introduces additional computational complexity not present in standard neural networks. To overcome this, we implicitly assume that the parameters in over-complete layers are Parseval tight frames, i.e. so that $\mathbf{B}_j\mathbf{B}_j^\mathsf{T} = \mathbf{I}$. This property is theoretically advantageous in the field of sparse approximation [8] and has been used as a constraint to encourage robustness in deep neural networks [32]. However, in our experiments we found that it was unnecessary to explicitly enforce this assumption during training; with appropriate learning rates, backpropagating through our inference algorithm was enough to ensure that repeated iterations did not result in diverging sequences of variable updates. Thus, under this assumption, we can simplify the update in Eq. 9 using the Woodbury matrix identity as follows:

$$\boldsymbol{w}_j^{[t+1]} := \tilde{\boldsymbol{z}}_j^{[t]} + \tfrac{1}{\rho+1}\mathbf{B}_j^\mathsf{T}\big(\boldsymbol{z}_{j-1}^{[t+1]} - \mathbf{B}_j\tilde{\boldsymbol{z}}_j^{[t]}\big), \quad \tilde{\boldsymbol{z}}_j^{[t]} := \boldsymbol{z}_j^{[t]} - \tfrac{1}{\rho}\boldsymbol{\lambda}_j^{[t]} \tag{12}$$

As this only involves simple linear transformations, our ADMM algorithm for solving the optimization problem in our inference function $\boldsymbol{f}^*$ can be expressed

Algorithm 1: Feed-Forward

Input: $\boldsymbol{x}$, $\{\mathbf{B}_j, \boldsymbol{b}_j\}$
Output: $\{\boldsymbol{w}_j\}$, $\{\boldsymbol{z}_j\}$
Initialize: $\boldsymbol{z}_0 = \boldsymbol{x}$
for $j = 1, \dots, l$ **do**
  Pre-activation:
    $\boldsymbol{w}_j \coloneqq \mathbf{B}_j^\mathsf{T} \boldsymbol{z}_{j-1}$
  Activation:
    $\boldsymbol{z}_j \coloneqq \phi_j(\boldsymbol{w}_j - \boldsymbol{b}_j)$
end

Algorithm 2: Alternating Direction Neural Network

Input: $\boldsymbol{x}$, $\{\mathbf{B}_j, \boldsymbol{b}_j\}$
Output: $\{\boldsymbol{w}_j^{[T]}\}$, $\{\boldsymbol{z}_j^{[T]}\}$
Initialize: $\{\boldsymbol{\lambda}_j^{[0]}\} = \mathbf{0}$, $\{\boldsymbol{w}_j^{[1]}, \boldsymbol{z}_j^{[1]}\}$ from Alg. 1
for $t = 1, \dots, T-1$ **do**
  for $j = 1, \dots, l$ **do**
    Dual: Update $\boldsymbol{\lambda}_j^{[t]}$ (Eq. 11)
    Pre-activation: Update $\boldsymbol{w}_j^{[t+1]}$ (Eq. 12)
    Activation: Update $\boldsymbol{z}_j^{[t+1]}$ (Eq. 10)
  end
end

as a recurrent neural network that repeatedly iterates until convergence. In practice, however, we unroll the network to a fixed number of iterations T for an approximation of optimal inference so that $\boldsymbol{f}^{[T]}(\boldsymbol{x}) \approx \boldsymbol{f}^*(\boldsymbol{x})$. Our full algorithm is summarized in Algs. 1 and 2.

4.1 Generalization of Feed-Forward Networks

Given proper initialization of the variables, a single iteration of this algorithm is identical to a single pass through a feed-forward network. Specifically, if we let $\boldsymbol{\lambda}_j^{[0]} = \mathbf{0}$ and $\boldsymbol{z}_j^{[0]} = \mathbf{B}_j^\mathsf{T} \boldsymbol{z}_{j-1}^{[1]}$, where we again denote $\boldsymbol{z}_0^{[1]} = \boldsymbol{x}$, then $\boldsymbol{w}_j^{[1]}$ is equivalent to the pre-activation of a neural network layer:

$$\boldsymbol{w}_j^{[1]} \coloneqq \mathbf{B}_j^\mathsf{T} \boldsymbol{z}_{j-1}^{[1]} + \tfrac{1}{\rho+1} \mathbf{B}_j^\mathsf{T} \big(\boldsymbol{z}_{j-1}^{[1]} - \mathbf{B}_j (\mathbf{B}_j^\mathsf{T} \boldsymbol{z}_{j-1}^{[1]}) \big) = \mathbf{B}_j^\mathsf{T} \boldsymbol{z}_{j-1}^{[1]} \tag{13}$$

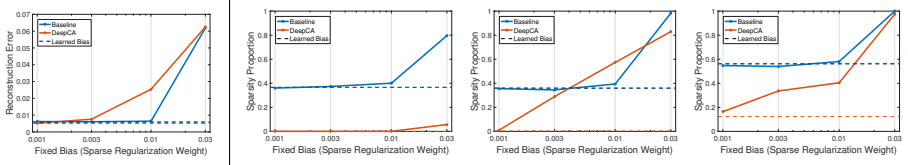
Similarly, if we initialize $\boldsymbol{w}_{j+1}^{[0]} = \mathbf{B}_{j+1}^\mathsf{T} \boldsymbol{w}_j^{[1]}$, then $\boldsymbol{z}_j^{[1]}$ is equivalent to the corresponding nonlinear activation using the proximal operator ϕ_j:

$$\boldsymbol{z}_j^{[1]} \coloneqq \phi_j \big(\tfrac{1}{\rho+1} \mathbf{B}_{j+1} (\mathbf{B}_{j+1}^\mathsf{T} \boldsymbol{w}_j^{[1]}) + \tfrac{\rho}{\rho+1} \boldsymbol{w}_j^{[1]} \big) = \phi_j \big(\boldsymbol{w}_j^{[1]} \big) \tag{14}$$

Thus, one iteration of our inference algorithm is equivalent to the standard feed-forward neural network given in Eq. 3, i.e. $\boldsymbol{f}^{[1]}(\boldsymbol{x}) = \boldsymbol{f}^{\mathrm{DNN}}(\boldsymbol{x})$, where nonlinear activation functions are interpreted as proximal operators corresponding to the penalties of our DeepCA model. Additional iterations through the network lead to more accurate inference approximations while explicitly satisfying constraints on the latent variables.

4.2 Learning by Backpropagation

With DeepCA inference approximated by differentiable ADNNs, the model parameters can be learned in the same way as standard feed-forward networks. Extending the nested component analysis optimization problem from Eq. 2, the



(a) Decoder Error (b) Layer 1 Sparsity (c) Layer 2 Sparsity (d) Layer 3 Sparsity

Fig. 4: A demonstration of the effects of fixed (solid lines) and learnable (dotted lines) bias parameters on the reconstruction error (a) and activation sparsity (b-d) comparing feed forward networks (blue) with DeepCA (red). All models consist of three layers each with 512 components. Due to the conditional dependence provided by recurrent feedback, DeepCA learns to better control the sparsity level in order improve reconstruction error. As ℓ_1 regularization weights, the biases converge towards zero resulting in denser activations and higher network capacity for reconstruction.

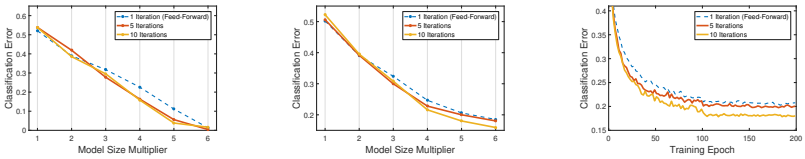
inference function $\boldsymbol{f}^{[T]}$ can be used as a generalization of feed-forward network inference $\boldsymbol{f}^{[1]}$ for backpropagation with arbitrary loss functions L that encourage the output to be consistent with provided supervision $\boldsymbol{y}^{(i)}$, as shown in Eq. 15 below. Here, only the latent coefficients $\boldsymbol{f}_l^{[T]}(\boldsymbol{x}^{(i)})$ from the last layer are shown in the loss function, but other intermediate outputs $j \neq l$ could also be included.

$$\underset{\{\mathbf{B}_j, \boldsymbol{b}_j\}}{\arg\min} \sum_{i=1}^{n} L\big(\boldsymbol{f}_l^{[T]}(\boldsymbol{x}^{(i)}),\, \boldsymbol{y}^{(i)}\big) \tag{15}$$

From an agnostic perspective, an ADNN can thus be seen as an end-to-end deep network architecture with a particular sequence of linear and nonlinear transformations and tied weights. More iterations ($T > 1$) result in networks with greater effective depth, potentially allowing for the representation of more complex nonlinearities. However, because the network architecture was derived from an algorithm for inference in our DeepCA model instead of arbitrary compositions of parameterized transformations, the greater depth requires no additional parameters and serves the very specific purpose of satisfying constraints on the latent variables while enforcing consistency with the model parameters.

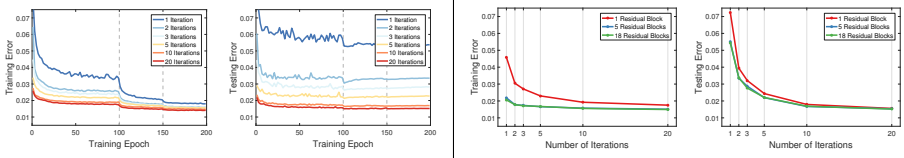
5 Experimental Results

In this section, we demonstrate some practical advantages of more accurate inference approximations in our DeepCA model using recurrent ADNNs over feed-forward networks. Even without additional prior knowledge, standard convolutional networks with ReLU activation functions still benefit from additional recurrent iterations as demonstrated by consistent improvements in both supervised and unsupervised tasks on the CIFAR-10 dataset [23]. Specifically, for an unsupervised autoencoder with an ℓ_2 reconstruction loss, Fig. 4 shows that the additional iterations of ADNNs allow for better sparsity control, resulting



(a) Training Error (b) Testing Error (c) Optimization

Fig. 5: The effect of increasing model size on training (a) and testing (b) classification error, demonstrating consistently improved performance of ADNNs over feed-forward networks, especially in larger models. The base model consists of two 3×3, 2-strided convolutional layers followed by one fully-connected layer with 4, 8, and 16 components respectively. Also shown are is the classification error throughout training (c).



(a) Training (b) Testing (c) Train Error (d) Test Error

Fig. 6: Quantitative results demonstrating the improved generalization performance of ADNN inference. The training (a) and testing (b) reconstruction errors throughout optimization show that more iterations ($T > 1$) substantially reduce convergence time and give much lower error on held-out test data. With a sufficiently large number of iterations, even lower-capacity models with encoders consisting of fewer residual blocks all achieve nearly the same level of performance with small discrepancies between training (c) and testing (d) errors.

in higher network capacity through denser activations and lower reconstruction error. This suggests that recurrent feedback allows ADNNs to learn richer representation spaces by explicitly penalizing activation sparsity. For supervised classification with a cross-entropy loss, ADNNs also see improved accuracy as shown in Fig. 5, particularly for larger models with more parameters per layer. Because we treat layer biases as learned hyperparameters that modulate the relative weight of ℓ_1 activation penalties, this improvement could again be attributed to this adaptive sparsity encouraging more discriminative representations across semantic categories.

While these experiments emphasize the importance of sparsity in deep networks and justify our DeepCA model formulation, the effectiveness of feed-forward soft thresholding as an approximation of explicit ℓ_1 regularization limits the amount of additional capacity that can be achieved with more iterations. As such, ADNNs provide much greater performance gains when prior knowledge is available in the form of constraints that *cannot* be effectively approximated by feed-forward nonlinearities. This is exemplified by our application of output-constrained single-image depth prediction where simple feed-forward correction

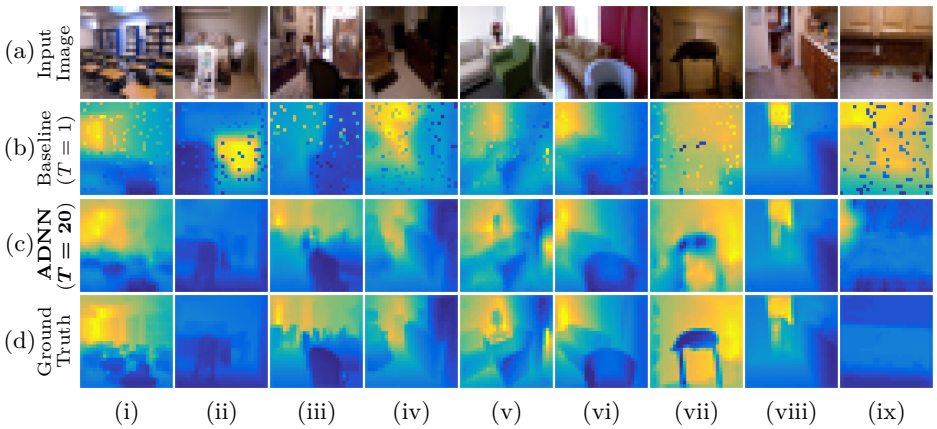
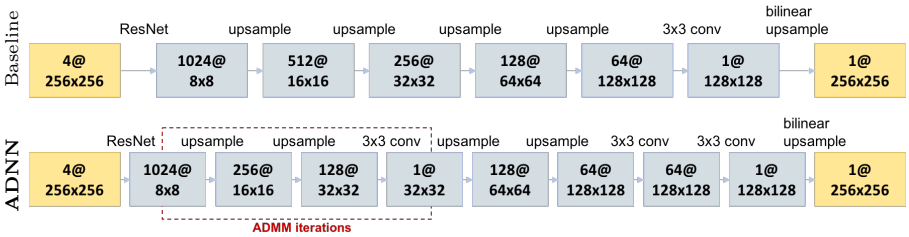


Fig. 7: Qualitative depth prediction results given a single image (a) and a sparse set of known depth values as input. Outputs of the baseline feed-forward model (b) are inconsistent with the constraints as evidenced by unrealistic discontinuities. An ADNN with $T = 20$ iterations (c) learns to enforce the constraints, resolving ambiguities for more detailed predictions that better agree with ground truth depth maps (d). Depending on the difficulty, additional iterations may have little effect on the output (viii) or be insufficient to consistently integrate the known constraint values (ix).

of the known depth values results in inconsistent discontinuities. We demonstrate this with the NYU-Depth V2 dataset [33], from which we sample 60k training images and 500 testing images from held-out scenes. To enable clearer visualization, we resize the images to 28×28 and then randomly sample 10% of the ground truth depth values to simulate known measurements. Following [30], our model architecture uses a ResNet encoder for feature extraction of the image concatenated with the known depth values as an additional input channel. This is followed by an ADNN decoder composed of three transposed convolution upsampling layers with biased ReLU nonlinearites in the first two layers and a constraint correction proximal operator in the last layer. Fig. 6 shows the mean absolute prediction errors of this model with increasing numbers of iterations and different encoder sizes. While all models have similar prediction error on training data, ADNNs with more iterations achieve significantly improved generalization performance, reducing the test error of the feed-forward baseline by over 72% from 0.054 to 0.015 with 20 iterations even with low-capacity encoders. Qualitative visualizations in Fig. 7 show that these improvements result from consistent constraint satisfaction that serves to resolve depth ambiguities.

In Figure 8, we also show qualitative and quantitative results on the full-sized images, an easier problem due to reduced ambiguities provided by higher-resolution details. While feed-forward models have achieved good performance given sufficient model capacity [30], they generalize poorly due to globally-biased prediction errors causing disagreement with the known measurements. By ex-



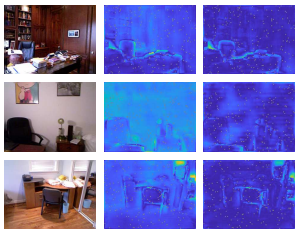


Image Baseline **ADNN**

Table 1: Quantitative Results

Method	ResNet	# Params	RMSE	Rel	δ_1	δ_2	δ_3
Baseline	18	1.5×10^7	0.54	0.16	79.2	94.7	99.4
ADNN	18	1.2×10^7	0.28	0.06	95.5	99.4	99.9
Baseline	10	8.8×10^6	0.56	0.16	79.8	94.6	99.4
ADNN	10	$\mathbf{6.5 \times 10^6}$	0.24	0.05	**97.3**	**99.6**	**99.9**
[30]	50	3.4×10^7	**0.23**	**0.04**	97.1	99.4	99.8

Fig. 8: Results on full-sized images from the NYU-Depth V2 dataset, comparing the feed-forward baseline and ADNN (with 10 iterations) architectures shown on top. On the left, example absolute error maps are visualized with lighter colors corresponding to higher errors and gray points indicating the locations of 200 randomly sampled measurements. On the right, quantitative metrics (following [30]) demonstrate the effect of changing the ResNet encoder size on prediction performance. Despite having far fewer learnable parameters, ADNNs perform comparably to a state-of-the-art feed-forward model due to explicit enforcement of the sparse output constraints.

plicitly enforcing agreement with the sparse output constraints, ADNNs reduce outliers and give improved test performance that is comparable with feed-forward networks requiring significantly more learnable parameters.

6 Conclusion

DeepCA is a novel deep model formulation that extends shallow component analysis techniques to increase representational capacity. Unlike feed-forward networks, intermediate network activations are interpreted as latent variables to be inferred using an iterative constrained optimization algorithm implemented as a recurrent ADNN. This allows for learning with arbitrary loss functions and provides a tool for consistently integrating prior knowledge in the form of constraints or regularization penalties. Due to its close relationship to feed-forward networks, which are equivalent to one iteration of this algorithm with proximal operators replacing nonlinear activation functions, DeepCA also provides a novel perspective from which to interpret deep learning, suggesting possible new directions for the analysis and design of network architectures from the perspective of sparse approximation theory.

References

1. Baldi, P., Hornik, K.: Neural networks and principal component analysis: Learning from examples without local minima. Neural networks **2**(1), 53–58 (1989)
2. Bao, C., Ji, H., Quan, Y., Shen, Z.: Dictionary learning for sparse coding: Algorithms and convergence analysis. Pattern Analysis and Machine Intelligence (PAMI) **38**(7), 1356–1369 (2016)
3. Belagiannis, V., Zisserman, A.: Recurrent human pose estimation. In: International Conference on Automatic Face & Gesture Recognition (FG) (2017)
4. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. Pattern Analysis and Machine Intelligence (PAMI) **35**(8), 1798–1828 (2013)
5. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine Learning **3**(1) (2011)
6. Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., Rother, C.: DSAC-differentiable RANSAC for camera localization. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
7. Carreira, J., Agrawal, P., Fragkiadaki, K., Malik, J.: Human pose estimation with iterative error feedback. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
8. Casazza, P.G., Kutyniok, G.: Finite frames: Theory and applications. Springer (2012)
9. Chen, C., Li, M., Liu, X., Ye, Y.: Extended ADMM and BCD for nonseparable convex minimization models with quadratic coupling terms: convergence analysis and insights. Mathematical Programming (2017)
10. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. Pattern Analysis and Machine Intelligence (PAMI) **PP**(99) (2017)
11. Chen, L.C., Schwing, A., Yuille, A., Urtasun, R.: Learning deep structured models. In: International Conference on Machine Learning (ICML) (2015)
12. Diamond, S., Sitzmann, V., Heide, F., Wetzstein, G.: Unrolled optimization with deep priors. arXiv preprint arXiv:1705.08041 (2017)
13. Gillis, N.: Sparse and unique nonnegative matrix factorization through data preprocessing. Journal of Machine Learning Research (JMLR) **13**(November), 3349–3386 (2012)
14. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: International Conference on Artificial Intelligence and Statistics (AISTATS) (2011)
15. Gregor, K., LeCun, Y.: Learning fast approximations of sparse coding. In: International Conference on Machine Learning (ICML) (2010)
16. Haeffele, B., Young, E., Vidal, R.: Structured low-rank matrix factorization: Optimality, algorithm, and applications to image processing. In: International Conference on Machine Learning (ICML) (2014)
17. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision (ECCV) (2016)
18. Hu, P., Ramanan, D.: Bottom-up and top-down reasoning with hierarchical rectified gaussians. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)

19. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
20. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., Murphy, K.: Speed/accuracy trade-offs for modern convolutional object detectors. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
21. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (ICML). pp. 448–456 (2015)
22. Jutten, C., Herault, J.: Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. Signal Processing **24**(1), 1–10 (1991)
23. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
24. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
25. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401**(6755), 788–791 (1999)
26. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: International Conference on Machine Learning (ICML) (2009)
27. Li, K., Hariharan, B., Malik, J.: Iterative instance segmentation. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
28. Lin, C.H., Lucey, S.: Inverse compositional spatial transformer networks. Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
29. Liu, T., Tao, D., Xu, D.: Dimensionality-dependent generalization bounds for k-dimensional coding schemes. Neural computation (2016)
30. Ma, F., Karaman, S.: Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In: International Conference on Robotics and Automation (ICRA) (2018)
31. Martins, A., Astudillo, R.: From softmax to sparsemax: A sparse model of attention and multi-label classification. In: International Conference on Machine Learning (ICML) (2016)
32. Moustapha, C., Piotr, B., Edouard, G., Yann, D., Nicolas, U.: Parseval networks: Improving robustness to adversarial examples. arXiv preprint arXiv:1704.08847 (2017)
33. Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: European Conference on Computer Vision (ECCV) (2012)
34. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: International Conference on Computer Vision (ICCV) (2015)
35. Olshausen, B.A., et al.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature **381**(6583), 607–609 (1996)
36. Papyan, V., Romano, Y., Elad, M.: Convolutional neural networks analyzed via convolutional sparse coding. Journal of Machine Learning Research (JMLR) **18**(83) (2017)
37. Parikh, N., Boyd, S., et al.: Proximal algorithms. Foundations and Trends® in Optimization **1**(3) (2014)
38. Patel, A.B., Nguyen, M.T., Baraniuk, R.: A probabilistic framework for deep learning. In: Advances in Neural Information Processing Systems (NIPS) (2016)

39. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Advances in Neural Information Processing Systems (NIPS) (2014)
40. Sulam, J., Papyan, V., Romano, Y., Elad, M.: Multi-layer convolutional sparse modeling: Pursuit and dictionary learning. arXiv preprint arXiv:1708.08705 (2017)
41. Sun, J., Li, H., Xu, Z., et al.: Deep ADMM-net for compressive sensing MRI. In: Advances in Neural Information Processing Systems (NIPS) (2016)
42. Taylor, G., Burmeister, R., Xu, Z., Singh, B., Patel, A., Goldstein, T.: Training neural networks without gradients: A scalable admm approach. In: International Conference on Machine Learning (ICML) (2016)
43. Tulsiani, S., Zhou, T., Efros, A.A., Malik, J.: Multi-view supervision for single-view reconstruction via differentiable ray consistency. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
44. Wold, S., Esbensen, K., Geladi, P.: Principal component analysis. Chemometrics and intelligent laboratory systems **2**(1-3), 37–52 (1987)
45. Xu, Y., Yin, W.: A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM Journal on imaging sciences **6**(3), 1758–1789 (2013)
46. Zamir, A.R., Wu, T.L., Sun, L., Shen, W., Malik, J., Savarese, S.: Feedback networks. In: Advances in Neural Information Processing Systems (NIPS) (2017)
47. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: International Conference on Learning Representations (ICLR) (2017)