Deep Double Sparsity Encoder:
Learning to Sparsify Not Only Features But Also Parameters

Zhangyang Wang†, Liang Zhang‡, Yingzhen Yang†, Jiayu Zhou§, Georgios B. Giannakis‡, Thomas S. Huang †
†Beckman Institute, University of Illinois at Urbana-Champaign
‡Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities.
§Department of Computer Science and Engineering, Michigan State University

Abstract
This paper emphasizes the significance to jointly exploit the problem structure and the parameter structure, in the context of deep modeling. As a specific and interesting example, we describe the deep double sparsity encoder (DDSE), which is inspired by the double sparsity model for dictionary learning. DDSE simultaneously sparsifies the output features and the learned model parameters, under one unified framework. In addition to its intuitive model interpretation, DDSE also possesses compact model size and low complexity. Extensive simulations compare DDSE with several carefully-designed baselines, and verify the consistently superior performance of DDSE. We further apply DDSE to the novel application domain of brain encoding, with promising preliminary results achieved.

Introduction
Whereas off-the-shelf deep models keep finding promising applications, it has been gradually recognized to incorporate the problem structure into the design of deep architectures. Such customized deep architectures can benefit from their problem-specific regularizations, and improve the performance. In particular, there has been a blooming interest in bridging sparse coding (Wang et al. 2015) and deep models. Starting from (Gregor and LeCun 2010), many work (Wang, Ling, and Huang 2016), (Wang et al. 2016c), (Wang et al. 2016b), (Wang et al. 2016d) leveraged similar ideas on fast trainable regressors, and constructed feed-forward network approximations to solve the variants of sparse coding models. Lately, (Xin et al. 2016) demonstrated both theoretically and empirically that a trained deep network is potentially able to recover \( l_0 \)-based sparse representations under milder conditions. The recent work (Wang et al. 2016a) further showed that a conventional feed-forward deep network could be interpreted as a stack of multiple unfolded and truncated regression models. It formally correlates a deep network with a hierarchical sparsifying process, where the intermediate features are approximating sparse codes.

The paper proceeds along this direction to embed sparsity regularization into the target deep model, and simultaneously exploits the structure of model parameters into the design of the model architecture. Up to our best knowledge, it is the first principled and unified framework, that jointly sparsifies both learned features and model parameters. The resulting deep feed-forward network, called deep double sparsity encoder (DDSE), enjoys a compact structure, a clear interpretation, an efficient implementation, and competitive performance, as verified by various comparison experiments. Its promising performance also manifests in the novel application domain of brain encoding.

Related Work
Network Implementation of Sparse Coding

![Figure 1: (a) The recursive system diagram for Eqn. (2); (b) a 3-layer neural network, unfolded and truncated to \( k = 2 \) iterations from (a).](image)

We start from the classical sparse coding model (Wang et al. 2015) \((\|D\|_2 = 1 \text{ by default hereinafter})\):

\[
\mathbf{a} = \arg \min_{\mathbf{a}} \frac{1}{2} \|\mathbf{x} - \mathbf{D}\mathbf{a}\|_2^2 + \lambda \|\mathbf{a}\|_1. \tag{1}
\]

\(\mathbf{x} \in \mathbb{R}^n\) denotes the input data, \(\mathbf{a} \in \mathbb{R}^m\) is the sparse code feature, \(\mathbf{D} \in \mathbb{R}^{n \times m}\) is the dictionary, and \(\lambda\) is the sparsity regularization coefficient. \(\mathbf{D}\) is usually chosen to be overcomplete, i.e. \(m > n\). Eqn. (1) could be solved by the iterative shrinkage and thresholding algorithm (ISTA) (Blumensath and Davies 2008) (\(\mathbf{u}\) is a vector and \(u_i\) is its \(i\)-th element):

\[
\mathbf{z}_{k+1} = \mathcal{N}(\mathcal{L}_1(\mathbf{x}) + \mathcal{L}_2(\mathbf{z}_k)), \text{ where:}
\]

\[
\mathcal{L}_1(\mathbf{x}) = \mathbf{D}^T \mathbf{x}, \quad \mathcal{L}_2(\mathbf{z}_k) = (\mathbf{I} - \mathbf{D}^T \mathbf{D}) \mathbf{z}_k, \tag{2}
\]

\[
\mathcal{N}(\mathbf{u})_i = \text{sign} (u_i) (|u_i| - \lambda)_+, \quad \lambda > 0
\]
where \( z^k \in \mathbb{R}^m \) denotes the intermediate output of the \( k \)-th iteration, \( k = 0, 1, \ldots \). \( L_1 \) and \( L_2 \) are linear operators that both hinge on \( D \), while \( \mathcal{N} \) is the element-wise soft shrinkage.

Eqn. (2) could be equivalently expressed by the recursive system in Figure 1(a), whose fixed point is expected to be the solution \( a \) of (1). Furthermore, Figure 1(a) could be unfolded and truncated to \( k \) iterations, to construct a \((k+1)\)-layer feedforward network (Gregor and LeCun 2010), as in Figure 1(b) Without any further tuning, the resulting learned ISTA architecture will output a trainable regressor overfitting, among others.

The Projected Gradient Descent Algorithm

Let \( \mathcal{G} \) denote the nonlinear mapping from the data to the last hidden feature before the loss, the optimization problem of another parameter to be learned jointly is the threshold \( \lambda \) in \( \mathcal{N} \). It is handled identically as in (Wang, Ling, and Huang 2016).

Deep Double Sparsity Encoder

The Proposed Model

Given \( D_0 \) and \( S \), we substitute (3) into (2) to obtain:

\[
L_1(x) = S^T D_0^2 x, \quad L_2(z^k) = (I - S^T D_0 S) z^k, \quad (4)
\]

with the iterative formula of \( z^k \) and the form of \( \mathcal{N} \) remaining the same. Compared to (2), \( S \) now becomes the trainable parameter in place of \( D \).

To simplify (4), we first eliminate \( D_0^T D_0 \) from \( L_2(z^k) \). Given the training data \( X_S \in R^{n \times t} = \{ x_i \}, i = 1, 2, \ldots, t \), and assuming \( X_S \) to have zero mean, we choose \( D_0 \) as the (full) eigenvector matrix of \( X_S X_S^T \) (i.e., the covariance matrix of \( X_S \)). The obtained \( D_0 \) constitutes an orthonormal basis for \( R^n \). Further, \( D_0^T x \) performs the PCA projection of \( x \), denoted as: \( x_{PCA} = D_0^T x \). The formula (4) is reduced to:

\[
L_1(x) = W_1 x_{PCA}, \quad L_2(z^k) = (I - W_3 W_2) z^k, \quad \text{where} \quad W_1 = S^T, \quad W_2 = S, \quad W_3 = S^T. \quad (5)
\]

We introduce three new variables in (5): \( W_1 \in R^{m \times n}, W_2 \in R^{m \times m}, \) and \( W_3 \in R^{m \times n} \). Both \( W_1 \) and \( W_3 \) have no more than \( s \) nonzero elements per column, while \( W_2 \) has no more than \( s \) nonzero elements per row. Figure 2 depicts the resulting deep double sparsity encoder (DDSE), unfolded and truncated from (5) (up to \( k = 2 \)). We purposely model \( W_2 \) and \( W_3 \) as two separate layers (with no nonlinearity in between), so that we could specify the proper row- or column-wise sparsity constraint on each.

Furthermore, under the loss function \( \mathcal{F}_\theta \), \( W_1, W_2 \) and \( W_3 \) can again be learned via end-to-end learning, instead of being constructed from any pre-computed \( S \). In this way, the DDSE network is solved over \( X_S \) by back-propagation, where \( W_l (l = 1, 2, 3) \) are treated as fully-connected layers. Different from (Wang, Ling, and Huang 2016), \( W_2 \) and \( W_3 \) are untied throughout iterations, in order to enlarge the learning capacity. We also relax the formulation (5), by decoupling \( W_l (l = 1, 2, 3) \) with each other, e.g., it is no longer required that \( W_1 = W_3 \), or \( W_2^T = W_3 \), during training. For simplicity, we use the same \( s \) for all \( W_l \).

Double Sparsity Model for Dictionary Learning

A crucial consideration in employing the sparse coding model (1) is the choice of the dictionary \( D \). It has been observed that the learned dictionary atoms are highly structured, with noticeably regular patterns (Peng et al. 2015). This gives rise to the hypothesis that the dictionary atoms themselves may have some underlying sparse structure over a more fundamental dictionary. (Rubinstein, Zibulevsky, and Elad 2010) proposed a double sparsity model, suggesting that each atom of the dictionary has itself a sparse representation over some pre-specified base dictionary \( D_0 \). The dictionary is therefore expressed as:

\[
D = D_0 S, \quad ||S(:, i)||_0 \leq s, \forall i, \quad (3)
\]

where \( S \) is the sparse atom representation matrix, which has no more than \( s \) nonzero elements per column (\( s \ll n, m \)). We also assume \( D_0 \in R^{n \times n} \) and \( S \in R^{n \times m} \). Note that in (Rubinstein, Zibulevsky, and Elad 2010), \( D_0 \) is chosen as \( R^{m \times m} \), and \( S \in R^{m \times m} \). We make slightly different choices in order for orthogonal \( D_0 \), whose benefits will be shown next. The base dictionary \( D_0 \) spans the signal space, and will generally be chosen to have a quick implementation. The new parametric structure of \( D \) leads to a simple and flexible dictionary representation which is both adaptive and efficient.

Advantages of the double sparsity model (3) also include compact representation, stability under noise and reduced overfitting, among others.

Figure 2: The proposed deep double sparsity encoder, unfolded and truncated to \( k = 2 \) iterations. The parameters \( W_l (l = 1, 2, 3) \) are subject to the constraints in Eqn. (6).
Apart from the constraints, the objective in (6) is usually minimized by the stochastic gradient descent (SGD) algorithm (γ is the learning rate):
\[
W_l = W_l - \gamma \frac{\partial F}{\partial W_l}, l = 1, 2, 3.
\] (7)

It is guaranteed to converge to a stationary point, under a few stricter assumptions than ones satisfied here Bottou 2010. With the constraints in (6) specifying the feasible sets, we move forward to the Projected Gradient Descent (PGD) algorithm:
\[
P_l(W) = \mathcal{P}_l(W_l), l = 1, 2, 3.
\] (8)

where \(\mathcal{P}_l\) is the projection onto the feasible set for \(W_l\). When \(l = 1, 3\), \(\mathcal{P}_l\) keeps the \(s\) largest-magnitude elements in each row of \(W_l\), and zeros out others. For \(l = 2\), \(\mathcal{P}_l\) is the same hard thresholding operator, but on a column-wise basis.

Since both the objective and feasible sets of (6) are non-convex, there is no convergence guarantee for PGD in (8). However, many literatures, e.g., Blumensath and Davies 2008, have demonstrated that solving such problems with PGD is well executed in practice. The stochastic implementation of PGD is also straightforward.

### Complexity Analysis

#### Model parameter complexity

For \(k\)-iteration DDSE, each \(W_l\) \((l = 1, 2, 3)\) is a sparse matrix of \(sm\) nonzero elements. The total amount of parameters in DDSE is \((2k + 1)sm\). In contrast, the LISTA network in Figure 1(b) takes \(mn + km^2\) parameters, assuming its \(L_2\) parameters not tied across iterations as well. Since \(s \ll m, n\), the parameter ratio turns out to be \(\frac{(2k+1)sm}{mn+km^2} = \frac{(2k+1)s}{n+km} \approx \frac{2s}{m} \ll 1\), as \(k \to \infty\). DDSE can thus be stored and loaded much more compactly, due to the sparse structure of \(W_l\). More importantly, DDSE can ensure the sufficient capacity and flexibility of learning by using large \(m\), and meanwhile effectively regularizes the learning process by choosing small \(s\).

#### Inference time complexity

The efficient multiplication of a sparse matrix with \(sm\) nonzero elements, and an arbitrary input vector, takes \(sm\) time. Given a \(k\)-iteration DDSE, the inference time complexity of one sample \(X \in \mathbb{R}^{n}\) is \(O((2k + 1)sm)\). In comparison, LISTA has a time complexity of \(O(mn + km^2)\). Again, when \(k \to \infty\), \(\frac{(2k+1)sm}{mn+km^2} \to \frac{2s}{m} \ll 1\).

#### Remark on the number of layers

When (5) is unfolded and truncated to \(k\) iterations, the obtained DDSE has 1 \(W_1\) layer, \(k - 1\) \(W_2\) layers, and \(k - 1\) \(W_3\) layers. However, since \(W_2\) and \(W_3\) are always linearly concatenated within each iteration, with no nonlinearity in between, we can also consider \(W_1W_2 \in \mathbb{R}^{n \times m}\) as one layer, whose two factors are individually regularized. Hence, we treat a DDSE unfolded to \(k\) iterations as a \((k+1)\)-layer network, which also follows the LISTA convention Gregor and LeCun 2010.

As a typical case in deep learning, SGD is widely used where it is not guaranteed to converge in theory, but behaves well in practice.

### Relationship to Existing Techniques

Many regularization techniques have been proposed to reduce overfitting in deep learning, such as dropout (Krizhevsky, Sutskever, and Hinton 2012), that set a randomly selected subset of activations to zero within each layer. (Wan et al. 2013) further introduced dropconnect for regularizing fully-connected layers, which instead sets a randomly selected subset of weights to zero during training. The proposed DDSE model implies an adaptive regime for dropconnect, where the selection of “dropped” weights is decided not randomly, but by data-driven hard thresholding. Besides, both dropout and dropconnect are only applied to training, and are unable to reduce the actual model size.

DDSE could be alternatively viewed to have a weight decay penalty, which is enforced by hard \(l_0\) constraints. The skip connections (a.k.a. shortcuts) in DDSE is also reminiscent of the residual learning strategy (He et al. 2016).

Last, we noticed a very recent work (Jin et al. 2016), that is related to DDSE. Similarly to (6), (Jin et al. 2016) proposed to impose explicit cardinality constraints to layer-wise parameters during training. However, their work and DDSE are substantially different in at least two-folds:

- As for hard-thresholding parameters, (Jin et al. 2016) investigated how it was feasible, while DDSE gives insights to why it is plausible, from a sparse coding and dictionary learning perspective. Recall the key point of DDSE that \(\mathbf{D} = \mathbf{D}_0\mathbf{S}\) and then the orthonormal \(\mathbf{D}_0\) vanishes. For structured signals (e.g., image, speech), a learned dictionary is also highly structured. It could be further decomposed as the product of a more fundamental basis (easily specified from data), and a sparse atom representation matrix. In short, DDSE provides a clear interpretation, that the inherent sparsity assumption of parameters is plausible, thanks to the structured basis of the signal space.

- (Jin et al. 2016) discussed the general training strategy, while our focus is to design DDSE as a specific model. DDSE is intended to pursue sparse features (as derived from the sparse coding model (1)), while pruning model parameters. It is explicitly targeted at discriminative many feature learning tasks where sparsity is known as desirable (Coates and Ng 2011), such as classification and clustering.

### Experiments

#### Implementation

The proposed DDSE is implemented with the CUDA ConvNet package (Krizhevsky, Sutskever, and Hinton 2012). We use a constant learning rate of 0.01, with the momentum parameter fixed at 0.9, and a batch size of 128. Neither dropout nor dropconnect is applied unless specified otherwise. We manually decrease the learning rate if the network stops improving as in (Krizhevsky, Sutskever, and Hinton 2012) according to a schedule determined on a validation set.

As suggested by (5), we first subtract the mean and conduct PCA over the training data \(X_S\). We adopt the multi-step update strategy in (Jin et al. 2016), namely, updating \(W_l\) by SGD without the cardinality constraints for several (15 by default) iterations, before the projection \(\mathcal{P}_l\) \((l = 1, 2, 3)\). It
both accelerates training by reducing the time of performing hard thresholding, and encourage DDSE to learn more informative parameters to make pruning more reliable.

While many neural networks are trained well with random initializations, it has been discovered that poor initializations can still hamper the effectiveness of first-order methods (Sutskever et al. 2013). On the other hand, it is much easier to initialize DDSE in the right regime. We first initialize $S$ by setting $s$ randomly selected elements to be one for each column, and zero elsewhere. Based on the correspondence relationships in (5), $W_{is}(i=1,2,3)$ are all trivially initialized from $S$. That helps DDSE achieve a steadily decreasing curve of training errors, without common tricks such as annealing the learning rate, which may be indispensable if random initialization is applied.

Simulation and Comparison

In the simulation experiments, we use the first 60,000 samples of the MNIST dataset for training and the last 10,000 for testing. A MNIST sample is a $28 \times 28$ gray-scale image, i.e., $n = 784$. Common data augmentations (noise, blur, flipping, rotation, and scaling) are applied. In addition to a $k$-iteration DDSE, we design five baselines for comparison:

- **Baseline I**: a $(k+1)$-layer fully-connected network, whose first layer $\in R^{r \times n}$ and remaining $k$ layers $\in R^{r \times r}$.
- **Baseline II**: Baseline I regularized by dropout, with a ratio of 0.5 (as in (Krizhevsky, Sutskever, and Hinton 2012)) for each layer.
- **Baseline III**: Baseline I regularized by dropconnect, with a ratio of 0.5 (as in (Wan et al. 2013)) for each layer.
- **Baseline IV**: a LISTA network, unfolded and truncated to $k$ iterations from (1). We also apply dropout to regularize its fully-connected layers.
- **Baseline V**: a network inspired by (Jin et al. 2016), by removing all “shortcuts” in DDSE while leaving all else unchanged.

All comparison methods are ensured to have the identical layer dimensions. They are jointly tuned with the softmax loss for the classification task. The default configuration parameters are $s = \frac{1}{4}n$, $m = 1,024$, $t = 60,000$, and $k = 2$. We further vary each of the four parameters, while keeping others unchanged, in our controlled experiments below.

**Sparsity level** $s$ Figure 3 varies the sparsity ratio $s/n$ from 0.1 to 0.6, and plots the corresponding error rates for all methods. Baselines I - IV are not parameterized by $s$ and thus not affected. Comparing Baselines II and III with Baseline I certifies that applying (even random) regularizations avoids overfitting and improves generalization. Baseline V and DDSE both benefit further from their more sophisticated regularization on the parameters. DDSE outperforms Baseline V with noticeable margins at all $s/n$ ratios, and reaches the best overall performance at $s/n = 0.25$.

As displayed in Figure 3, the performance of Baseline V and DDSE will both be degraded with either too small or too large $s/n$ ratios. Whereas increasing $s/n$ may loose the regularization effect, a small $s/n$ also implies over-regularization, limiting the representation power of free parameters. In the random dropout/dropconnect cases, the popular practice is to choose $s/n$ around 0.5. (Jin et al. 2016) also observed the best $s/n$ to be between 0.4 and 0.5. DDSE seems to admit a lower “optimal” $s/n$ (around 0.25). It implies that DDSE could attain more competitive performance with less parameters (i.e., lower $s/n$), by “smartly” selecting non-zero elements in a data-driven way.

![Figure 3](image3.png)

**Feature dimension** $m$ In (1), the choice of $m$ corresponds to the dimensionality of the learned sparse code feature, and turns into the hidden layer dimensions of DDSE, etc. As illustrated in Figure 4, we start from $m = 800$, and raise it up to 2,000. Not surprisingly, the performance of Baseline I is degraded with $m$ growing larger, due to obviously overfitting. All other methods, regularized in various ways, all seem to benefit from larger $m$ values. Among them, DDSE consistently outperforms others, with a 0.2% error rate margin over Baseline IV (the second best). It proves effective to handle highly over-complete and redundant basis, and hence to learn more sparse hidden features.

![Figure 4](image4.png)

**Training sample size** $t(t_s)$ DDSE is meant to seek a trade-off between “data-driven” and “model-based” methods. By
confining the degrees of freedom of parameters and permitting only certain sparse combinations over a pre-specified base dictionary, the parameter structure model (5) may enable us to reduce, in some cases significantly, the amount of training data required to reliably approximate and recover the underlying nonlinear mapping of the deep model.

We empirically verify our conjecture, by the following comparison experiment. A small subset of size $t_s$ is drawn from $\mathbf{X}_{\Sigma}$ (the MNIST dataset with $t = 60,000$ samples), where each class is sampled proportionally. We range the ratio $t_s/t$ from 0.1 to 1. Figure 5 shows that DDSE leads to dramatically more robust learning and generalization, under insufficient training data. Even when $t_s/t$ is as low as 0.05, DDSE only bears a slight performance loss of 2.46%, while Baselines IV and V are degraded for more than 6% and 4%, respectively. It is also noteworthy that, to achieve the same performance level of DDSE at $t_s/t = 0.05$, Baselines IV and V requires approximately $t_s/t = 0.4$, Baselines II and III take $t_s/t = 0.5$, and Baseline I even needs $t_s/t > 0.8$. Those observations strongly support our hypothesis, that DDSE greatly alleviates the need for large training data, by exploiting the prior knowledge of parameter structure. In addition, we note that Baseline V slightly outperforms Baseline IV in Figure 5. Recall that similarly to DDSE, the regularization on the Baseline V parameter is also enforced by data-driven adaptive sparsity. Under small training data, it appears more effective than the random dropout.

**Number of layers $k + 1$** The last simulation investigates how well DDSE and other methods can be scaled to deeper cases. We grow $k$ from 1 to 6, resulting to 2 to 7-layer networks. The comparison in Figure 6 evidently demonstrates the superiority of DDSE with all $k$ values. Besides, it is also interesting to see from Figure 6 that Baseline IV obtains a significant performance advantage over Baseline V as $k$ grows. It is opposite to the observation in Figure 5. On one hand, it might be attributed to the utility of “shortcuts”, as analyzed in (He et al. 2016). On the other hand, we believe that the incorporation of the original problem structure (1) also places deep models in good conditions: increasing $k$ is resemblant to running (2) up to more iterations, and thus solving (1) more precisely.

**Concluding remarks** We conclude from the above experiments, that both the problem structure (“sparsifying features”) and the parameter structure (“sparsifying parameters”) have contributed to the superior performance of DDSE.

By the comparison to Baselines II and III, the sophisticated regularization of DDSE is found to be more powerful than random ones such as dropout/dropconnect. Compared to Baseline IV, DDSE further utilizes the double sparsity structure of the dictionary (3) as a priori, which accounts for its improved performance in all aspects. In the meanwhile, exploiting the structure of the original problem (1), that encourages sparse and more discriminative features, also helps DDSE outperform Baseline V consistently.

Besides, although the simulations are only intended for proof-of-concepts, the result of default-configured DDSE has already had comparable results to the 6-layer neural network in (Ciresan et al. 2010), and the committee of 25 neural networks trained with elastic distortions in (Meier et al. 2011).

**Application to Brain Encoding** DDSE shows superior performance in simulations. In particular, the experiments on varying $t(t_s)$ has identified a sharp performance margin for DDSE over the others. It thus brings up the possibility for deep learning applied with small data. In this part, we explore the applicability of DDSE to the novel application domain of brain encoding (Kay et al. 2008).

**Background and Motivation** Brain encoding refers to the challenging task to predict the brain activity, e.g., blood oxygenation level dependent (BOLD) responses, from the stimuli. Lately, (Kay et al. 2013) developed a two-stage cascaded second-order contrast (SOC) model, that accepts a grayscale image as input and predicts BOLD responses in early visual cortex as output. The SOC model has only eight controlling parameters: it heavily relies on specific nonlinear computations, that are summarized from neuroscience expertise.
In contrast to SOC, we study more parameterized, data-driven deep models. Such models are more flexible, by allowing the data to inform the model as to what types of computations are necessary. One major obstacle arises from the fact that training data are extremely limited and thus model parameters are precious. It is also infeasible to artificially increase the data volume, such as generating “synthetic” data or perform data augmentation. Therefore, the classical “data-driven” learning setting does not directly apply here.

**Dataset**

We refer to Kendrick Kay et. al.’s publicly available datasets of BOLD responses in visual cortex, measured by functional magnetic resonance imaging (fMRI) in human subjects. Specifically, we adopt their stimulus set 2, stimulus set 3, (response) dataset 4, and (response) dataset 5.

All stimuli are band-pass filtered grayscale images. Following (Kay et al. 2013), we resize all the stimuli to 150×150 pixels. Stimulus sets 2 and 3 consist of 156 and 35 distinct stimuli, respectively. The responses at a total of 200 voxels are recorded. On each voxel, a scalar fMRI measurement was measured given each input stimulus. Note that each voxel needs to train a separate brain encoding model. Dataset 4 consists of one person’s responses to stimulus set 2, while dataset 5 has the same person’s responses to stimulus set 3. Details about the datasets could be found at (Kay et al. 2013).

Our goal is to train a regression-type model using stimulus set 2 and dataset 4 (as the *training set*). The model is used to generate predictions and be evaluated on stimulus set 3 and dataset 5 (as the *testing set*). Obviously, it is an ill-conditioned “small data” problem.

**Model and Experiment**

We are motivated to apply DDSE to the brain encoding scenario, for three reasons:

- The sensory processing in the brain suggests a sparse coding strategy over a *highly over-complete basis*, when finding stimuli that effectively activate the neurons (Olshausen and Field 1997). Such a neuroscience ground advocates to sparsify the features in the brain encoding model.

- As proven above, DDSE is capable to exploit small training data more effectively, and avoids overfitting.

To adapt DDSE for the regression task, we replace the softmax function, with a max pooling operator: $R^m \rightarrow R$, followed by a mean square error (MSE) loss. With $n = 22,500$, we choose $k = 2$, $m = 30,000$ for the over-complete representation, and $s = 30 (s/m = 0.1\%)$ for a highly sparse dictionary structure. A fully-connected (FC) network is also constructed for comparison, with three hidden layers of 100-dimension, and regularized by dropout. We initialize the first layer of FC using PCA, and the remaining two layers using layer-wise pre-training, to carefully ensure its proper convergence. A larger FC network is found to be difficult to converge, and presents very bad performance.

The preliminary experiment has found encouraging results. The model accuracy is quantified as the percentage of variance explained ($R^2$) in the measured response amplitudes by the cross-validated predictions of the response amplitudes (see page. 14, (Kay et al. 2013) for definitions). $R^2$ ranges between [0, 100]: the higher $R^2$ is, the more accurate the model is. In terms of averaged $R^2$ performance across 200 voxels, the SOC baseline obtains 87.7028, and the averaged $R^2$ for FC is as poor as 68.6027. DDSE leads to the best averaged $R^2$ of 89.8022. Figure 7 also compares the voxel-wise $R^2$ performance of SOC, FC, and DDSE. While FC turns fairly untrustworthy at many voxels, DDSE gives the most reliable predictions on the majority of them.

As for the future work, it is natural to extend (1) and (3) to the convolutional counterparts, and to have convolutional DDSE applied to the brain encoding model of visual stimuli.

**Summary**

The study of DDSE showcases how jointly exploiting the problem structure and the parameter structure improves the deep modeling. Both simulations and the application to brain encoding have verified its consistently superior performance, as well as robustness to highly insufficient training data. In our future work, a wide variety of parameter structures will be exploited for different models as a priori, such as the subspace structure (Peng et al. 2016), and the tree structure (Baraniuk et al. 2010).
References

[Baraniuk et al. 2010] Baraniuk, R. G.; Cevher, V.; Duarte, M. F.; and Hegde, C. 2010. Model-based compressive sensing. IEEE Transactions on Information Theory 56(4):1982–2001.

[Blumensath and Davies 2008] Blumensath, T., and Davies, M. E. 2008. Iterative thresholding for sparse approximations. Journal of Fourier Analysis and Applications.

[Bottou 2010] Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT’2010. Springer.

[Ciresan et al. 2010] Ciresan, D.; Meier, U.; Gambardella, L.; and Schmidhuber, J. 2010. Deep big simple neural nets excel on handwritten digit recognition. http://arxiv.org/pdf/1003.0358.

[Coates and Ng 2011] Coates, A., and Ng, A. Y. 2011. The importance of encoding versus training with sparse coding and vector quantization. In ICML.

[Erhan et al. 2010] Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.-A.; Vincent, P.; and Bengio, S. 2010. Why does unsupervised pre-training help deep learning? JMLR.

[Gregor and LeCun 2010] Gregor, K., and LeCun, Y. 2010. Learning fast approximations of sparse coding. In ICML.

[He et al. 2016] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. IEEE CVPR.

[Jin et al. 2016] Jin, X.; Yuan, X.; Feng, J.; and Yan, S. 2016. Training skinny deep neural networks with iterative hard thresholding methods. arXiv preprint arXiv:1607.05423.

[Kay et al. 2008] Kay, K. N.; Naselaris, T.; Prenger, R. J.; and Gallant, J. L. 2008. Identifying natural images from human brain activity. Nature 452(7185):352–355.

[Kay et al. 2013] Kay, K. N.; Winawer, J.; Rokem, A.; Mezer, A.; and Wandell, B. A. 2013. A two-stage cascade model of bold responses in human visual cortex. PLoS Comput Biol 9(5):e1003079.

[Krizhevsky, Sutskever, and Hinton 2012] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In NIPS.

[Meier et al. 2011] Meier, U.; Ciresan, D. C.; Gambardella, L. M.; and Schmidhuber, J. 2011. Better digit recognition with a committee of simple neural nets. In ICDAR. IEEE.

[Olshausen and Field 1997] Olshausen, B. A., and Field, D. J. 1997. Sparse coding with an overcomplete basis set: A strategy employed by v1? Vision research 37(23):3311–3325.

[Peng et al. 2015] Peng, X.; Lu, C.; Yi, Z.; and Tang, H. 2015. Connections between nuclear norm and frobenius norm based representation. arXiv preprint arXiv:1502.07423.

[Peng et al. 2016] Peng, X.; Xiao, S.; Feng, J.; Yau, W.-Y.; and Yi, Z. 2016. Deep subspace clustering with sparsity prior. In The 25th International Joint Conference on Artificial Intelligence.

[Rubinstein, Zibulevsky, and Elad 2010] Rubinstein, R.; Zibulevsky, M.; and Elad, M. 2010. Double sparsity: Learning sparse dictionaries for sparse signal approximation. IEEE Transactions on signal processing 58(3):1553–1564.

[Sutskever et al. 2013] Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In ICML.

[Wan et al. 2013] Wan, L.; Zeiler, M.; Zhang, S.; Cun, Y. L.; and Fergus, R. 2013. Regularization of neural networks using dropconnect. In Proceedings of the 30th International Conference on Machine Learning (ICML-13). 1058–1066.

[Wang et al. 2015] Wang, Z.; Yang, J.; Zhang, H.; Wang, Z.; Yang, Y.; Liu, D.; and Huang, T. S. 2015. Sparse Coding and its Applications in Computer Vision. World Scientific.

[Wang et al. 2016a] Wang, Z.; Chang, S.; Ling, Q.; Huang, S.; Hu, X.; Shi, H.; and Huang, T. S. 2016a. Stacked approximated regression machine: A simple deep learning approach. In NIPS.

[Wang et al. 2016b] Wang, Z.; Chang, S.; Liu, D.; Ling, Q.; and Huang, T. S. 2016b. D3: Deep dual-domain based fast restoration of jpeg-compressed images. In IEEE CVPR.

[Wang et al. 2016c] Wang, Z.; Chang, S.; Zhou, J.; Wang, M.; and Huang, T. S. 2016c. Learning a task-specific deep architecture for clustering. SDM.

[Wang et al. 2016d] Wang, Z.; Yang, Y.; Chang, S.; Ling, Q.; and Huang, T. S. 2016d. Learning deep $\ell_\infty$ encoder for hashing. In IJCAI.

[Wang, Ling, and Huang 2016] Wang, Z.; Ling, Q.; and Huang, T. 2016. Learning deep $\ell_0$ encoders. AAAI.

[Xin et al. 2016] Xin, B.; Wang, Y.; Gao, W.; and Wipf, D. 2016. Maximal sparsity with deep networks? NIPS.