DATA DRIVEN CONVOLUTIONAL SPARSE CODING FOR VISUAL RECOGNITION

Yijie Zeng⋆ *Jichao Chen*⋆ *Guang-Bin Huang*⋆

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

ABSTRACT

Convolutional sparse coding (CSC) has become an important method in image processing and computer vision. In this paper we focus on visual recognition problems and apply CSC as a feature learning method. We propose a task-specific approach to treat the dictionary of CSC as parameters for a larger learning framework. These parameters are differentiable under mild conditions, and could be updated end-to-end using back-propagation when the errors from the task objectives are provided. We perform several experiments to show that such method provides a more discriminate representation compared with previous CSC methods, and this data driven approach is effective for visual recognition problems.

Index Terms— convolutional sparse coding, dictionary learning, sparse representation, supervised learning, visual recognition

1. INTRODUCTION

Sparse coding (SC) algorithms aims to represent a vector with the linear combination of a set of over-complete basis under sparsity constraints. It has been successfully applied to different tasks, notably computer vision and visual recognition problems [1], where the learned representations serve as discriminative features for higher level vision tasks. Traditionally, dictionaries for such representations are learned by extracting patches in an image database and then using unsupervised methods such as K-Means [2]. This relies on the assumption that these overlapping patches are independent of each other, thus it fails to model shift invariance within the same image and is not optimal. On the other hand, convolutional sparse coding (CSC) attempts to remedy this by using *convolution operation*:

$$\min_{\{\alpha_i\}_{i=1}^m, \{d_i\}_{i=1}^m} \frac{1}{2}\|x - \sum_{i=1}^m d_i * \alpha_i\|_2^2 + \lambda \sum_{i=1}^m \|\alpha_i\|_1, \quad \text{s.t.} \quad \|d_i\|_2 \leq 1, \forall\, i = 1, \ldots, m. \tag{1}$$

Instead of learning from patches, the whole input vector $x \in \mathbb{R}^{\mathrm{N}}$ is represented as the combination of m feature maps $\{\alpha_i\}_{i=1}^m$ under m convolutional filters $\{d_i\}_{i=1}^m$, where $\alpha_i \in \mathbb{R}^{\mathrm{N}}$ and $d_i \in \mathbb{R}^{\mathrm{n}}$. In this paper, we demonstrate that the corresponding dictionaries d_i could be learned using a data driven approach. We utilize label information for each image and back-propagate errors [3] from our task objective to the target sparse code. Using optimality condition of (1), the derivative with respect to the objective could be obtained and dictionary is updated using stochastic gradient descent.

Related works. Due to the difficulties of solving (1), the developments of convolutional sparse coding algorithms have been focusing on efficient solvers for CSC. Most of them adopt ADMM [4] and solve (1) in frequency domain [5, 6, 7]. Recent works in [8] also discuss the theoretical foundation of CSC, and they propose several efficient algorithms to solve CSC on time domain. Although efficient computational methods are not the main focus of this paper, we have to stress that CSC enforces a heavier computational load compared with sparse coding, and an efficient solver is the key to a successful CSC algorithm.

Traditional methods for sparse feature learning is split into two phases: a training phase for *unsupervised* learning of dictionaries and an encoding phase for decomposing images as sparse representations of dictionary basis [9]. The encoded (sparse) features are then used to train a classifier. The idea of learning dictionaries using convolutional dictionary learning is discussed in [10, 11, 12], where they aim to build a *hierarchical* representation by stacking up multiple convolutional sparse coding modules. Although these methods have been successful in visual recognition problems, it would be more desirable if dictionaries could adapt to specific vision tasks. Such method is discussed in [13, 14] under patch-based sparse dictionary learning contexts. In particular, [14] proposes to update the dictionary in a supervised way and achieves competitive results. Recently, [15] provides an alternative way of supervised dictionary learning by combining CSC framework with sparse representation classification method (SRC) in [16]. They train one set of dictionaries specific for each class and each image is matched to the class with smallest reconstruction residual at test phase. Although the label information is encoded in the dictionary, the classifier itself is nonparametric and cannot be generalized to larger scale. Moreover, the dictionaries could not be updated batch-by-batch in an online learning fashion.

Our contributions. This paper describes an algorithm to learn the convolutional dictionaries in convolutional sparse coding problems in a supervised way. Inspired by [14], the

⋆{yzeng004,e160075,egbhuang}@ntu.edu.sg

sparse code is regarded as a function of the input vector and the dictionary. The optimality condition is applied which shows that under mild condition the task objective is differentiable with the dictionaries. We show close form solution of the gradients so that the supervised loss could be optimized using back-propagation method. Compared to previous results [15], our method is an end-to-end algorithm with no separate learning stage, and we also demonstrate superior performance on benchmarking dataset.

2. DATA DRIVEN CONVOLUTIONAL SPARSE CODING

2.1. Problem Formulation

Given an input vector x, the convolutional sparse coding algorithm computes the sparse code α for each x with parameters $\mathbf{D}_l$:

$$\alpha(x, \mathbf{D}_l) = \underset{\{\alpha_i\}}{\operatorname{argmin}} \frac{1}{2}\|x - \sum_{i=1}^{m} d_i * \alpha_i\|_2^2 + \lambda_1 \sum_{i=1}^{m} \|\alpha_i\|_1 + \frac{\lambda_2}{2} \sum_{i=1}^{m} \|\alpha_i\|_2^2. \quad (2)$$

Here $x \in \mathbb{R}^{\mathrm{N}}$ is the input vector, $\alpha_i \in \mathbb{R}^{\mathrm{N}}$ is the convolutional sparse code, $d_i \in \mathbb{R}^{\mathrm{n}}$ is the filter, α is the concatenation of sparse codes and $\mathbf{D}_l = [d_1, \ldots, d_m] \in \mathbb{R}^{\mathrm{n}\times\mathrm{m}}$ is the concatenation of the filters. Notice we use the *elastic net* [17] formulation, which is important for the stability of backward computation and will be discussed later. After encoding using (2), the supervised learning problem is formulated as the following loss objective $\mathcal{L}$ under expectation of joint sample space $\mathcal{Y} \times \mathcal{X}$:

$$\min_{\mathbf{D}_l, \mathbf{W}} \mathcal{L}(\mathbf{D}_l, \mathbf{W}) = \mathbb{E}_{\mathcal{Y},\mathcal{X}}\Big[l_s(y, \mathbf{W}, \alpha(x, \mathbf{D}_l))\Big] \quad \text{s.t.} \quad \mathbf{D}_l \in \mathcal{D}_l, \mathbf{W} \in \mathcal{W}, \quad (3)$$

where $y \in \mathcal{Y}$ is the label, $x \in \mathcal{X}$ is the sample. l_s defines a *supervised loss* and $\mathbf{W}$ is the collection of parameters for this loss. $\mathcal{D}_l$ and $\mathcal{W}$ are the constraint sets for $\mathbf{D}_l$ and $\mathbf{W}$ respectively. In this way the label information $y \in \mathcal{Y}$ is back-propagated directly from (3) to $\mathbf{D}_l$ using stochastic gradient descent. Our main objective is to minimize (3) to learn a task specific dictionary.

2.2. Forward Encoding

The form of (2) is actually hard to optimize. [5, 6, 7] propose to solve (2) (when $\lambda_2 = 0$) using ADMM. Here instead, we adopt the *slice representation* of the convolution [8]. Assuming a circular boundary, the convolution operation is equivalent to a matrix multiplication:

$$\sum_{i=1}^{m} d_i * \alpha_i = \sum_{i=1}^{m} \mathbf{D}_i \alpha_i = \mathbf{D}\alpha, \quad (4)$$

where $\mathbf{D}$, α are the concatenation of circulant matrices $\mathbf{D}_i$ and sparse codes α_i respectively. It is noticed in [8] that rearranging columns of $\mathbf{D}$ gives rise to a more compact representation:

$$\mathbf{D}\alpha = \sum_{i=1}^{N} \mathbf{R}_i^{\mathrm{T}} \mathbf{D}_l \tilde{\alpha}_i, \quad (5)$$

where $\{\tilde{\alpha}_i\}_{i=1}^{N}$ is the rearragement of $\{\alpha_i\}_{i=1}^{m}$. $\mathbf{R}_i \in \mathbb{R}^{\mathrm{n}\times\mathrm{N}}$ is a *patch extractor* that extracts the i-th patch from the input vector, and its transpose $\mathbf{R}_i^{\mathrm{T}}$ aggregates the patch to reconstruct the vector.[1]

Using such representation, (2) could be solved iteratively using ISTA [18] as described in [8]:

$$\alpha_i^{k+1} = \mathcal{S}_{\lambda_1/c}\Big((1 - \frac{\lambda_2}{c})\alpha_i^k + \frac{1}{c}\mathbf{D}_l^{\mathrm{T}}\mathbf{R}_i(x - \sum_i \mathbf{R}_i^{\mathrm{T}}\mathbf{D}_l\alpha_i)\Big), \forall i. \quad (6)$$

Notice the additional coefficient $(1 - \lambda_2/c)$, which is due to the L_2 regularization term. $c > \lambda_{\max}(\mathbf{D}^{\mathrm{T}}\mathbf{D})$ to ensure convergence in ISTA.

2.3. Backward Update

To update the dictionary $\mathbf{D}_l$ we need to calculate the derivative of (3) with respect to $\mathbf{D}_l$. The differentiability property is proved in Section 4 of [14], where they consider the ordinary sparse coding problem. We summarize the result into the following lemma:

Lemma 1 *For ordinary elastic net problem* $\alpha(x, \mathbf{D}) = \operatorname{argmin} \frac{1}{2}\|x - \mathbf{D}\alpha\|_2^2 + \lambda_1\|\alpha\|_1 + \frac{\lambda_2}{2}\|\alpha\|_2^2$*, if* $\lambda_2 > 0$ *and (i)* $\mathcal{X}$ *and* $\mathcal{Y}$ *are compact, (ii)* l_s *is twice continuously differentiable, (iii)* $p_{\mathcal{Y},\mathcal{X}}(y, \cdot)$ *is continuous for any* $y \in \mathcal{Y}$*, then the function* $\mathcal{L}$ *in* (3) *is differentiable, and*

$$\nabla_{\mathbf{D}}\mathcal{L}(\mathbf{D}, \mathbf{W}) = \mathbb{E}_{\mathcal{Y},\mathcal{X}}\Big[-\mathbf{D}\beta\alpha^{\mathrm{T}} + (x - \mathbf{D}\alpha)\beta^{\mathrm{T}}\Big], \quad (7)$$

where $\beta \in \mathbb{R}^{\mathrm{Nm}}$ *is of the same size as* α *satisfying*

$$\beta_\Lambda = \big(\mathbf{D}_\Lambda^{\mathrm{T}}\mathbf{D}_\Lambda + \lambda_2\mathbf{I}\big)^{-1}\nabla_{\alpha_\Lambda} l_s(y, \mathbf{W}, \alpha), \quad (8)$$

and $\beta_{\Lambda^C} = 0$*.* $\Lambda = \{j \,|\, j \in \{1, \ldots, Nm\},\, \alpha[j] \neq 0\}$ *is the nonzero index set of* α*.*

□

Notice that the full matrix $\mathbf{D}$ is parameterized by $\mathbf{D}_l$, thus $\nabla_{\mathbf{D}_l}\mathcal{L}(\mathbf{D}_l, \mathbf{W})$ could be computed from $\nabla_{\mathbf{D}}\mathcal{L}(\mathbf{D}, \mathbf{W})$ by the chain rule:

[1] Since we are always using the slice decomposition (5), in the following we use α_i instead of $\tilde{\alpha}_i$ to simplify notation.

Algorithm 1: Sliced-based Supervised Dictionary Learning for Convolutional Sparse Coding

Data: Sample space $\mathcal{Y} \times \mathcal{X}$, initial values $\mathbf{W}$ and $\mathbf{D}_l$, iteration T, learning rate η.

for t *in* $1 : T$ **do**

Sample data $(y_t, x_t) \in \mathcal{Y} \times \mathcal{X}$;

Solve α_t from (2) using ISTA (6);

Compute loss $l_s(y_t, \mathbf{W}, \alpha_t)$;

Compute β_t using (8) and back-propagated error $\nabla_\alpha l_s(y_t, \mathbf{W}, \alpha_t)$;

Compute the derivative $\nabla_{\mathbf{D}_l}\mathcal{L}(\mathbf{D}_l, \mathbf{W})$ using (11);

Compute the derivative $\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{D}_l, \mathbf{W})$ using back-propagation;

Update weights:

$$\begin{aligned} \mathbf{W} &\leftarrow \Pi_{\mathcal{W}}\big(\mathbf{W} - \eta\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{D}_l, \mathbf{W})\big) \\ \mathbf{D}_l &\leftarrow \Pi_{\mathcal{D}_l}\big(\mathbf{D}_l - \eta\nabla_{\mathbf{D}_l}\mathcal{L}(\mathbf{D}_l, \mathbf{W})\big) \end{aligned} \tag{12}$$

$$\begin{aligned} &\frac{\partial \mathcal{L}}{\partial \mathrm{d}_{ik}}(\mathbf{D}_l, \mathbf{W}) \\ &= \mathrm{tr}\left[\nabla_{\mathbf{D}}\mathcal{L}(\mathbf{D}, \mathbf{W})^{\mathrm{T}}\frac{\partial \mathbf{D}}{\partial \mathrm{d}_{ik}}\right] \\ &= \mathbb{E}\left[\mathrm{tr}\Big[(-\mathbf{D}\beta\alpha^{\mathrm{T}} + (x - \mathbf{D}\alpha)\beta^{\mathrm{T}})^{\mathrm{T}}\frac{\partial \mathbf{D}}{\partial \mathrm{d}_{ik}}\Big]\right] \\ &= \mathbb{E}\left[-\beta^{\mathrm{T}}\mathbf{D}^{\mathrm{T}}\frac{\partial \mathbf{D}}{\partial \mathrm{d}_{ik}}\alpha + (x - \mathbf{D}\alpha)^{\mathrm{T}}\frac{\partial \mathbf{D}}{\partial \mathrm{d}_{ik}}\beta\right]. \end{aligned} \tag{9}$$

Using the parametric representation of $\mathbf{D}\alpha$ (and similarly for $\mathbf{D}\beta$),

$$\frac{\partial \mathbf{D}}{\partial \mathrm{d}_{ik}}\alpha = \sum_{j=1}^{N}\mathbf{R}_j^{\mathrm{T}}\frac{\partial \mathbf{D}_l}{\partial \mathrm{d}_{ik}}\alpha_j = \sum_{j=1}^{N}\mathbf{R}_j^{\mathrm{T}}\mathbf{E}_{ik}\alpha_j, \tag{10}$$

where $\mathbf{E}_{ik} \in \mathbb{R}^{\mathrm{n}\times\mathrm{m}}$ is 1 at (i, k) and 0 otherwise, the final result is

$$\begin{aligned} \nabla_{\mathbf{D}_l}\mathcal{L}(\mathbf{D}_l, \mathbf{W}) = \mathbb{E}\Big[&-\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{R}_i\mathbf{R}_j^{\mathrm{T}}\mathbf{D}_l\beta_j\alpha_i^{\mathrm{T}} \\ &+ \sum_{i=1}^{N}\mathbf{R}_i\Big(x - \sum_{j=1}^{N}\mathbf{R}_j^{\mathrm{T}}\mathbf{D}_l\alpha_j\Big)\beta_i^{\mathrm{T}}\Big]. \end{aligned} \tag{11}$$

Here β is solved by (8), and $\beta_i \in \mathbb{R}^{\mathrm{m}}$ is the i-th slice of β. The form of (11) is very similar to (7). When the derivative is computed, $\mathbf{D}_l$ could be updated using SGD. The main algorithm is summarized in Algorithm 1.

2.4. Discussion on Computational Issues

The heaviest computation falls on equation (8), where we have to compute the inverse of a large matrix for every iteration, and the complexity is $\mathcal{O}(p^3)$ for $p = |\Lambda|$. The same problem exists for the original CSC problem, but since $\mathbf{D}$ has a banded circulant structure, this problem could be solved efficiently by FFT [5, 7]. On the other hand, because of the fact that the value of Λ is *data dependent* and varies for each sample and each iteration, the block circulant structure of $\mathbf{D}^{\mathrm{T}}\mathbf{D}$ is not preserved, and our backward update (12) could not benefit from this Fourier trick. Although there are other methods (exact or approximate) for solving a positive definite system, empirically we find none could beat this complexity by a large margin when sparsity is high ($p \ll Nm$), and they often bring heavy overheads which makes it harder to optimize. Thus in our implementation we simply use Cholesky factorization to solve (8).

An alternative way of accelerating the gradient computation is to *reduce the context size*, i.e. cropping the full images into smaller patches for encoding. Such approach is very similar to traditional patch-based sparse coding methods, except that we compute CSC code on each image patch instead of solving SC problem. Intuitively, pixels that are far away within the same image is less relevant to each other than those that are near, so it is reasonable to assume that they are independent. When such cropping strategy is applied, ideally the time complexity for matrix inverse will be reduced by k if the image is cropped to k^2 patches.

3. EXPERIMENTS

In this section we show experimental results on MNIST [19] hand written digit classification. For all our experiments the convolutional sparse code is solved via (6) with the kernel size of 5×5, which is then max-pooled to a spatial resolution of 4×4. The output is used as a feature for an ordinary fully connected network with softmax loss. We fix $\lambda_1 = 0.1$ and $\lambda_2 = 1$ for all our experiments. These values are picked empirically instead of cross validation. We use a learning rate of 0.01 and reduce by a factor of 0.1 when the loss plateaus. The model is learned end-to-end using Caffe [20].

3.1. Training Versus Encoding

We first demonstrate the effectiveness of the proposed algorithm by comparing classification results on MNIST with or without training the dictionary using Algorithm 1. We initialize the values of the dictionary from a Gaussian distribution with mean 0 and standard derivation of 0.01, which is independent of the data distribution. We set the number of feature maps for our dictionary to be 128 for both experiments, and iterate over the training set for 2100 iterations with batch size 100. To evaluate how reducing context size in Section 2.4

Table 1. Comparison of different methods and training data settings on MNIST dataset

method	settings	accuracy (%)
Ours (small context)	all samples	96.99
Ours (no update)	all samples	98.27
Ours (supervised)	all samples	**98.72**
CSC+SVM[2] [12]	300 training samples	83.1
CSCC [15]	300 training samples	84.5
Ours (supervised)	300 training samples	**90.2**

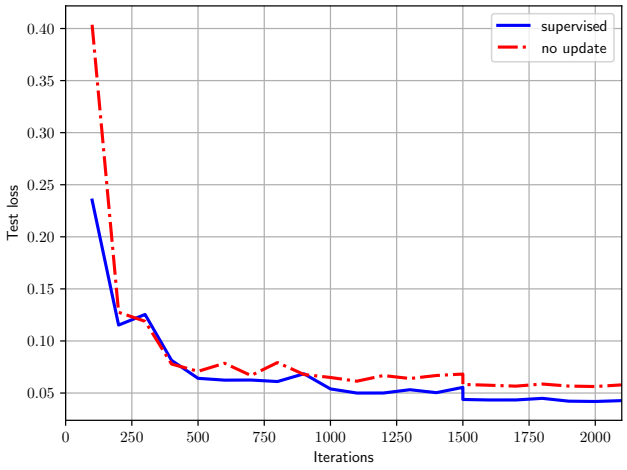

Fig. 1. Effect of supervised training compared to randomly initialized case on MNIST

influences accuracy, we use the same experimental settings except for a smaller context size of 7×7.

The results could be seen in Table 1 where all samples are used. We notice that even if the dictionary is not updated, the encoded features are still reaching to an accuracy of 98.27%. This phenomenon is not surprising and has been observed in [9] on the CIFAR-10 dataset. They find that proper encoding has a larger influence on sparse feature learning than an unsupervised dictionary learning method such as K-Means. Our proposed supervised CSC method increases the accuracy to a slightly better result of 98.72%. We also show in Fig 1 that the loss is systematically lower that that of a random initialization case, indicating that dictionary learning is also a critical step for sparse feature learning.

We also observe the accuracy for small context case. The performance is worse than the random dictionary case in full context scenario. Considering that MNIST is a rather clean and structural dataset, we hypothesize that the relations between patches are important as well. We also observe the update speed increase by a factor between 2 and 3 with smaller context size, which approximately follows our estimate of a factor of 4 (for patch size reducing from 28 to 7).

[2]This accuracy result is adopted from [15].

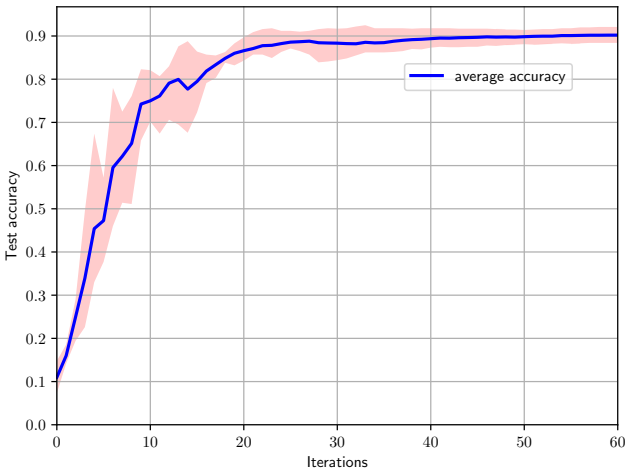

Fig. 2. Accuracy on MNIST using 300 training samples

3.2. Compare with Previous Results

We also compare with other results on MNIST using CSC. We note results reported in [15], where they learn one dictionary for each class. For MNIST they use 15 filters, 30 training samples and 100 test samples per class. For fair comparison we use the same number of training and testing samples, but only 100 filters. The learning rate is fixed at 0.01 and the training process stops early at 60 iterations since the training set size is very small. Other parameter settings are the same as Section 3.1. The final result is averaged over five runs, which is shown in Fig 2. The red zone is the accuracy for each iteration.

We compare to results reported in [15] in Table 1. Our supervised algorithm reaches to an accuracy of over 90%, beating previous ones by a large margin with fewer filters and smaller kernels.

4. CONCLUSION

We have presented in this paper a supervised formulation of dictionary learning for convolutional sparse coding. Such supervision allows the dictionaries to be fine-tuned for specific tasks, and the whole framework could be learned end-to-end using stochastic gradient descent. Experiments on MNIST hand written digit classification demonstrate that such method is more effective in recognition problem than previous CSC methods. Future work would be focusing on more efficient backward update methods. In addition, a hybrid method combining reconstruction capacity and the power of supervision for convolutional sparse coding is also desirable.

5. REFERENCES

[1] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031–1044, June 2010.

[2] Adam Coates and Andrew Y. Ng, *Learning Feature Representations with K-Means*, pp. 561–580, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[3] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[5] Hilton Bristow, Anders Eriksson, and Simon Lucey, "Fast convolutional sparse coding," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[6] Felix Heide, Wolfgang Heidrich, and Gordon Wetzstein, "Fast and flexible convolutional sparse coding," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[7] B. Wohlberg, "Efficient convolutional sparse coding," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 7173–7177.

[8] 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, Nov 2017.

[9] Adam Coates and Andrew Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, USA, 2011, ICML'11, pp. 921–928, Omnipress.

[10] Koray Kavukcuoglu, Pierre Sermanet, Y lan Boureau, Karol Gregor, Michael Mathieu, and Yann L. Cun, "Learning convolutional feature hierarchies for visual recognition," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds., pp. 1090–1098. Curran Associates, Inc., 2010.

[11] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 2528–2535.

[12] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2018–2025.

[13] J. A. Bagnell and David M. Bradley, "Differentiable sparse coding," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., pp. 113–120. Curran Associates, Inc., 2009.

[14] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, April 2012.

[15] B. Chen, J. Li, B. Ma, and G. Wei, "Convolutional sparse coding classification model for image classification," in *2016 IEEE International Conference on Image Processing (ICIP)*, Sept 2016, pp. 1918–1922.

[16] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, Feb 2009.

[17] Hui Zou and Trevor Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[18] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

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

[20] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, New York, NY, USA, 2014, MM '14, pp. 675–678, ACM.