# Task-Driven Dictionary Learning based on Convolutional Neural Network Features

Tom Tirer
*School of Electrical Engineering*
*Tel Aviv University*
Tel Aviv, Israel
tirer.tom@gmail.com

Raja Giryes
*School of Electrical Engineering*
*Tel Aviv University*
Tel Aviv, Israel
raja@tauex.tau.ac.il

***Abstract*—Modeling data as a linear combination of a few elements from a learned dictionary has been used extensively in the recent decade in many fields, such as machine learning and signal processing. The learning of the dictionary is usually performed in an unsupervised manner, which is most suitable for regression tasks. However, for other purposes, e.g. image classification, it is advantageous to learn a dictionary from the data in a supervised way. Such an approach has been referred to as *task-driven dictionary learning*. In this work, we integrate this approach with deep learning. We modify this strategy such that the dictionary is learned for features obtained by a convolutional neural network (CNN). The parameters of the CNN are learned simultaneously with the task-driven dictionary and with the classifier parameters.**

## I. Introduction

In the recent decade many signal and image processing applications have benefited remarkably from adopting low dimensional models for the signals of interest. The most popular form of this low dimensional modeling assumes that signals admit a sparse representation in a given dictionary - this is referred to as the "sparsity model" [1]. A crucial factor in the success of sparse representation methods is the choice of the dictionary, which should provide a faithful encoding for a given signal, but also be able to discriminatively represent different signals.

Using off-the-shelf dictionaries, such as redundant wavelets or discrete cosine transform, may suffice for restoration problems of general images, but will not be effective enough for specific tasks, such as clasification of images from a certain domain. Indeed, the dictionary learning (DL) approach, in which the dictionary is learned from the training data has led to state-of-the-art results in many practical applications, such as image reconstruction and image classification.

This work is supported by ERC StG SPADE.

There are two main strategies for dictionary learning: unsupervised DL and supervised DL. Unsupervised DL methods, which conveniently spare any need for data labeling, have been widely applied to image processing tasks, such as image denoising and compression [2], [3], [4]. Two popular unsupervised DL algorithms are K-SVD [5] and online DL [6]. However, without using additional label information of training data, unsupervised DL methods may not acquire discriminative representations for signals that belong to different classes. Therefore, while these methods are powerful for data reconstruction, they may not be a good choice for classification tasks. In contrast, supervised DL do use available class labels of training samples. This class discrimination information is exploited while learning the dictionary, and thus leads to better classification performance [7], [8].

In this work we build on the task-driven dictionary learning framework proposed in [7]. We use this framework for multiclass classification. However, we modify the original algorithm such that the dictionary is learned for features obtained by a convolutional neural network (CNN) [9]. The parameters of the CNN are learned simultaneously with the task-driven dictionary and with the classifier parameters. The proposed method outperforms the original algorithm that learns a linear transform of the input data. In addition, in the case of a small number of training examples, it can get better performance than the full CNN, from which we extract the features.

Our effort is related to other recent works that combines neural network training with low parsimonious data models. For example, in [10] it has been shown that a CNN may be interpreted as a convolutional sparse coding process, which has led to theoretical guarantees for the uniqueness of outputs of each layer of the

ISBN 978-90-827970-1-5 

network. In [11] a novel deep network has been proposed that encodes features of hidden layers with nonnegative sparse representation. In [12] the network weights were decomposed as a sparse composition of a fixed set of filters. In [13], low-rank based loss has been proposed to improve the performance of neural networks.

## II. Background

For the sake of completeness, we briefly describe the unsupervised DL and the (supervised) task-driven DL. Classical unsupervised dictionary learning techniques consider a finite training set of signals $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m \times n}$ and minimize the empirical cost function

$$\min_{\boldsymbol{D}} \frac{1}{n} \sum_{i=1}^{n} \ell_u(\boldsymbol{x}_i, \boldsymbol{D}) \tag{1}$$

with respect to a dictionary $\boldsymbol{D} \in \mathbb{R}^{m \times p}$, where each column of $\boldsymbol{D}$ represents a dictionary element. $\ell_u$ (the subscript $u$ denotes "unsupervised") is a loss-function such that $\ell_u(\boldsymbol{x}, \boldsymbol{D})$ should be small if $\boldsymbol{D}$ is "good" at representing the signal $\boldsymbol{x}$ in a sparse fashion. Alternatively, it is more likely that one is interested in the minimization of the expected cost

$$\min_{\boldsymbol{D}} \mathbb{E}_{\boldsymbol{x}}\{\ell_u(\boldsymbol{x}, \boldsymbol{D})\}, \tag{2}$$

where the expectation is taken relative to the (unknown) probability distribution $p(\boldsymbol{x})$ of the data. This kind of cost functions is usually minimized using stochastic gradient descent (SGD) methods.

We define $\ell_u(\boldsymbol{x}, \boldsymbol{D})$ as the optimal value of the elastic-net sparse coding problem [14]

$$\ell_u(\boldsymbol{x}, \boldsymbol{D}) \triangleq \min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{D}\boldsymbol{\alpha}\|_2^2 + \lambda_1 \|\boldsymbol{\alpha}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}\|_2^2, \tag{3}$$

where $\lambda_1$ and $\lambda_2$ are regularization parameters. Setting $\lambda_2 = 0$ yields the $\ell_1$ sparse decomposition problem (also known as basis pursuit denoising [15] or Lasso [16]), while using $\lambda_2 > 0$ results in a more stable solution. To prevent the $\ell_2$-norm of $\boldsymbol{D}$ from being arbitrarily large, which would lead to arbitrarily small values of $\boldsymbol{\alpha}$, it is common to constrain $\boldsymbol{D}$ to the following convex set of matrices

$$\mathcal{D} \triangleq \{\boldsymbol{D} \in \mathbb{R}^{m \times p} \text{ s.t. } \forall j \in \{1, \ldots, p\} \ \|\boldsymbol{d}_j\|_2 \leq 1\}. \tag{4}$$

where $\{\boldsymbol{d}_j\}$ denote the columns of $\boldsymbol{D}$.

We turn now to describe the task-driven DL formulation. Given a dictionary $\boldsymbol{D}$, a vector $\boldsymbol{x}$ can be represented as a sparse vector $\boldsymbol{\alpha}^*(\boldsymbol{x}, \boldsymbol{D})$ defined as the solution of (3). We assume that each signal $\boldsymbol{x} \in \mathcal{X}$ is associated with a variable $\boldsymbol{y} \in \mathcal{Y}$, which we want to predict from $\boldsymbol{x}$ (e.g., a label in classification tasks). We can now use the sparse vector $\boldsymbol{\alpha}^*(\boldsymbol{x}, \boldsymbol{D})$ as a feature representation of a signal $\boldsymbol{x}$ in the following minimization formulation:

$$\min_{\boldsymbol{W} \in \mathcal{W}} \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\{\ell_s(\boldsymbol{y}, \boldsymbol{W}, \boldsymbol{\alpha}^*(\boldsymbol{x}, \boldsymbol{D}))\} + \frac{\nu}{2}\|\boldsymbol{W}\|_F^2, \tag{5}$$

where $\boldsymbol{W}$ are model parameters which we want to learn, $\mathcal{W}$ is a convex set, and $\nu$ is a regularization parameter. In this equation, $\ell_s$ (the subscript $s$ denotes "supervised") is a convex loss function that measures how well one can predict $\boldsymbol{y}$ by observing $\boldsymbol{\alpha}^*(\boldsymbol{x}, \boldsymbol{D})$ given the model parameters $\boldsymbol{W}$. For instance, it can be the square, logistic, or hinge loss. The expectation is taken with respect to the unknown probability distribution $p(\boldsymbol{x}, \boldsymbol{y})$ of the data.

Finally, the task-driven dictionary learning formulation consists of jointly learning $\boldsymbol{W}$ and $\boldsymbol{D}$ by solving

$$\min_{\boldsymbol{D} \in \mathcal{D}, \boldsymbol{W} \in \mathcal{W}} \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\{\ell_s(\boldsymbol{y}, \boldsymbol{W}, \boldsymbol{\alpha}^*(\boldsymbol{x}, \boldsymbol{D}))\} + \frac{\nu}{2}\|\boldsymbol{W}\|_F^2. \tag{6}$$

Note that the function in the expectation is non-convex and non-smooth. The non-convexity is typical to DL problems, and handled by alternating minimization: performing sparse coding with a fixed dictionary, and updating the dictionary using the obtained representations. The non-smoothness is mitigated by the expectation operator together with properties of $\boldsymbol{\alpha}^*(\boldsymbol{x}_i, \boldsymbol{D})$, obtained by the elastic-net formulation with $\lambda_2 > 0$, as detailed in [7].

The formulation (6) is further extended in [7] by including a linear transform of the input data, represented by a matrix $\boldsymbol{Z}$, which allows to increase or decrease the dimension of the features $\boldsymbol{x}$. Thus, we have

$$\min_{\boldsymbol{D} \in \mathcal{D}, \boldsymbol{W} \in \mathcal{W}, \boldsymbol{Z} \in \mathcal{Z}} \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\{\ell_s(\boldsymbol{y}, \boldsymbol{W}, \boldsymbol{\alpha}^*(\boldsymbol{Z}\boldsymbol{x}, \boldsymbol{D}))\} + \frac{\nu_1}{2}\|\boldsymbol{W}\|_F^2 + \frac{\nu_2}{2}\|\boldsymbol{Z}\|_F^2. \tag{7}$$

where $\nu_1$ and $\nu_2$ are two regularization parameters, and $\mathcal{Z}$ is a convex set.

The optimization of (7) can be carried out using the projected stochastic gradient descent scheme, presented in Algorithm 1.

## III. The Proposed Method

Inspired by the excellent performance of recent deep-learning-based methods at different tasks [17], [18], [19], [20], we turn to extend the task-driven DL such that the dictionary is learned for features obtained by a convolutional neural network (CNN) [9], instead of a simple linear transform of the input data. In more details, instead

ISBN 978-90-827970-1-5 © EURASIP 2018

**Algorithm 1:** SGD algorithm for task-driven DL

**Input:** $p(\boldsymbol{x},\boldsymbol{y})$ (a way to draw i.i.d samples), $\lambda_1,\lambda_2,\nu_1,\nu_2$ (regularization parameters), $\boldsymbol{D}\in\mathcal{D},\boldsymbol{W}\in\mathcal{W},\boldsymbol{Z}\in\mathcal{Z}$ (initializations), $T$ (number of iterations), $t_0,\rho$ (learning rate parameters).

**for** $t=1$ *to* $T$ **do**

Draw $(\boldsymbol{x},\boldsymbol{y})$ from $p(\boldsymbol{x},\boldsymbol{y})$;

Sparse coding: compute $\boldsymbol{\alpha}^*(\boldsymbol{Z}\boldsymbol{x},\boldsymbol{D})$ using elastic-net;

$\Lambda=\{j\in\{1,\ldots,p\}:\boldsymbol{\alpha}^*[j]\neq 0\}$;

$\boldsymbol{\beta}^*_\Lambda=(\boldsymbol{D}_\Lambda^T\boldsymbol{D}_\Lambda+\lambda_2\boldsymbol{I})^{-1}\nabla_{\boldsymbol{\alpha}_\Lambda}\ell_s(\boldsymbol{y},\boldsymbol{W},\boldsymbol{\alpha}^*)$;

$\boldsymbol{\beta}^*_{\Lambda^C}=\boldsymbol{0}$;

$\rho_t=\min(\rho,\rho\frac{t_0}{t})$;

$\boldsymbol{W}=\Pi_{\mathcal{W}}[\boldsymbol{W}$
$-\rho_t\left(\nabla_{\boldsymbol{W}}\ell_s(\boldsymbol{y},\boldsymbol{W},\boldsymbol{\alpha}^*)+\nu_1\boldsymbol{W}\right)]$;

$\boldsymbol{Z}=\Pi_{\mathcal{Z}}[\boldsymbol{Z}-\rho_t\left(\boldsymbol{D}\boldsymbol{\beta}^*\boldsymbol{x}^T+\nu_2\boldsymbol{Z}\right)]$;

$\boldsymbol{D}=\Pi_{\mathcal{D}}[\boldsymbol{D}$
$-\rho_t\left(-\boldsymbol{D}\boldsymbol{\beta}^*\boldsymbol{\alpha}^{*T}+(\boldsymbol{Z}\boldsymbol{x}-\boldsymbol{D}\boldsymbol{\alpha}^*)\boldsymbol{\beta}^{*T}\right)]$;

**end**

**Output:** $\boldsymbol{D}\in\mathcal{D},\boldsymbol{W}\in\mathcal{W},\boldsymbol{Z}\in\mathcal{Z}$.

of training the dictionary for sparse representations of the linearly transformed $\boldsymbol{x}$:

$$\tilde{\boldsymbol{x}}_{linear}\triangleq\boldsymbol{Z}\boldsymbol{x}, \tag{8}$$

we train it for sparse representations of the features

$$\tilde{\boldsymbol{x}}_{CNN}\triangleq\boldsymbol{f}_{CNN}(\boldsymbol{x};\boldsymbol{z}), \tag{9}$$

where $\boldsymbol{f}_{CNN}$ is a non-linear function of $\boldsymbol{x}$, obtained by common building blocks of CNNs, such as convolutional layers and max-pooling layers. $\boldsymbol{z}$ is an $N_{\boldsymbol{z}}\times 1$ vector that denotes all the parameters of the network, e.g. the weights of the convolutional layers. The proposed architecture is demonstrated in Fig. 1. Note that the parameters of the CNN layers are not fixed. They are learned simultaneously with the task-driven dictionary $\boldsymbol{D}$ and with the model parameters $\boldsymbol{W}$. Thus, the formulation in (7) is extended to

$$\min_{\boldsymbol{D}\in\mathcal{D},\boldsymbol{W}\in\mathcal{W},\boldsymbol{z}\in\mathcal{Z}}\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\left\{\ell_s\left(\boldsymbol{y},\boldsymbol{W},\boldsymbol{\alpha}^*\left(\tilde{\boldsymbol{x}}(\boldsymbol{x};\boldsymbol{z}),\boldsymbol{D}\right)\right)\right\}+\frac{\nu_1}{2}\|\boldsymbol{W}\|_F^2+\frac{\nu_2}{2}\|\boldsymbol{z}\|_2^2. \tag{10}$$

Note the general dependency of $\boldsymbol{\alpha}^*$ on $\boldsymbol{z}$ through $\tilde{\boldsymbol{x}}$, which equals to $\tilde{\boldsymbol{x}}_{linear}$ when linear transformation is used and to $\tilde{\boldsymbol{x}}_{CNN}$ when a CNN is employed.

We restrict ourselves to standard static neural networks, which have bounded results given that their inputs are bounded [21]. Therefore, the function $\boldsymbol{\alpha}^*(\tilde{\boldsymbol{x}}_{CNN},\boldsymbol{D})$

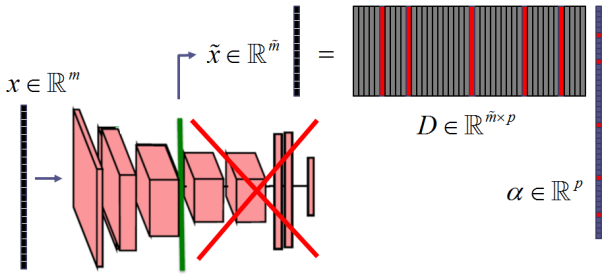


Fig. 1: Proposed architecture for joint dictionary and CNN learning.

is still uniformly Lipschitz, and the differentiability proofs in [7] are still valid. However, the update rule for $\boldsymbol{z}$ has to be modified to our general case

$$\boldsymbol{z}=\Pi_{\mathcal{Z}}[\boldsymbol{z}-\rho_t\left(\nabla_{\boldsymbol{z}}\ell_s+\nu_2\boldsymbol{z}\right)]. \tag{11}$$

Following the computation technique of [22], we have

$$\nabla_{\boldsymbol{z}}\ell_s=\frac{\partial\tilde{\boldsymbol{x}}}{\partial\boldsymbol{z}}\boldsymbol{D}\boldsymbol{\beta}^*, \tag{12}$$

where we used the chain rule, and $\boldsymbol{\beta}^*$ is defined exactly as in Algorithm 1. As a sanity check, in the case $\tilde{\boldsymbol{x}}=\boldsymbol{Z}\boldsymbol{x}$ (where $\boldsymbol{Z}\in\mathbb{R}^{\tilde{m}\times m}$), we have

$$\frac{\partial(\boldsymbol{Z}\boldsymbol{x})}{\partial\mathrm{vec}(\boldsymbol{Z})}=\boldsymbol{x}\otimes\boldsymbol{I}_{\tilde{m}}, \tag{13}$$

where $\otimes$ is the Kronecker product and $\boldsymbol{I}_{\tilde{m}}$ is the identity matrix of size $\tilde{m}\times\tilde{m}$. Therefore, we have

$$\nabla_{\boldsymbol{z}}\ell_s=(\boldsymbol{x}\otimes\boldsymbol{I}_{\tilde{m}})\boldsymbol{D}\boldsymbol{\beta}^*=(\boldsymbol{x}\otimes\boldsymbol{D})\boldsymbol{\beta}^*=\mathrm{vec}(\boldsymbol{D}\boldsymbol{\beta}^*\boldsymbol{x}^T), \tag{14}$$

which is the vectorization of $\partial\ell_s/\partial\boldsymbol{Z}=\boldsymbol{D}\boldsymbol{\beta}^*\boldsymbol{x}^T$ that appears in [7]. Exploiting the fact that the optimization algorithm is based on stochastic gradient descent, we use backpropagation [23] to calculate $\nabla_{\boldsymbol{z}}\ell_s$, with the vector $\boldsymbol{D}\boldsymbol{\beta}^*$ being used as the derivatives of the network w.r.t. the feature layer's output.

## IV. Application to Digits Classification

We consider here a classification task using the MNIST handwritten digits dataset [24], which contains 70,000 $28\times 28$ images, 60,000 of them for training and 10,000 for testing.

We use a single dictionary and a multiclass loss function. This strategy has less computational cost compared to other strategies such as one-versus-all and one-versus-one. Thus, in our setting, $\mathcal{Y}=\{1,\ldots,10\}$. Given a vector $\boldsymbol{x}$, we want to learn (jointly with $\boldsymbol{D}$ and $\boldsymbol{z}$) the parameters $\boldsymbol{W}\in\mathbb{R}^{p\times 10}$ of a linear model to predict $y\in\mathcal{Y}$, using the sparse representation $\boldsymbol{\alpha}^*(\tilde{\boldsymbol{x}},\boldsymbol{D})$ as

ISBN 978-90-827970-1-5 

features. Each column of $\boldsymbol{W}$ is associated with a specific class, i.e. $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{10}]$, and the maximal entry of $\boldsymbol{W}^T\boldsymbol{\alpha}^*$ determines the predicted class. The loss function that we use is the categorical cross-entropy loss (also known as softmax loss), which is given by

$$\ell_s(y, \boldsymbol{W}, \boldsymbol{\alpha}^*) = -\log\left(\frac{\mathrm{e}^{\boldsymbol{w}_y^T\boldsymbol{\alpha}^*}}{\sum_{\tilde{y}\in\mathcal{Y}} \mathrm{e}^{\boldsymbol{w}_{\tilde{y}}^T\boldsymbol{\alpha}^*}}\right). \tag{15}$$

Note that the derivatives required for Algorithm 1 can be computed easily. Let

$$P_y \triangleq \frac{\mathrm{e}^{\boldsymbol{w}_y^T\boldsymbol{\alpha}^*}}{\sum_{\tilde{y}\in\mathcal{Y}} \mathrm{e}^{\boldsymbol{w}_{\tilde{y}}^T\boldsymbol{\alpha}^*}}, \tag{16}$$

then, it can be shown that

$$\frac{\partial \ell_s(y, \boldsymbol{W}, \boldsymbol{\alpha}^*)}{\partial \alpha^*[i]} = \sum_{\tilde{y}\in\mathcal{Y}} \left(-w_y[i] + w_{\tilde{y}}[i]\right) P_{\tilde{y}}, \tag{17}$$

$$\frac{\partial \ell_s(y, \boldsymbol{W}, \boldsymbol{\alpha}^*)}{\partial w_{\overline{y}}^*[i]} = \alpha^*[i]\left(P_{\overline{y}} - \delta[\overline{y} - y]\right), \tag{18}$$

where $\delta[\cdot]$ denotes the Kronecker delta.

We compare the performance of three methods:

- full-CNN: an 8-layers CNN from [25].
- TDDL-LIN: task-driven DL with $\tilde{\boldsymbol{x}} = \boldsymbol{Z}\boldsymbol{x}$.
- TDDL-CNN: task-driven with $\tilde{\boldsymbol{x}} = \boldsymbol{f}_{CNN}(\boldsymbol{x}; \boldsymbol{z})$.

For the full-CNN, we use the exact 8-layers CNN architecture and training scheme (including the optimization algorithm) supplied in the MatConvNet toolbox [25] for the MNIST dataset.

For the task-driven methods, similar to [7], each image is normalized to zero mean and unit norm. Also, it is worthwhile to decrease the dimension of the input vectors in order to reduce the runtime. For TDDL-LIN we use a $196 \times 784$ matrix $\boldsymbol{Z}$, which is initialized with entries independently drawn from the standard normal distribution $\mathcal{N}(0, 1)$. For TDDL-CNN we keep only the first 4 layers of the full CNN, and reduce the number of filters in the third layer. The first layer is a convolutional layer with 20 filters of size $5 \times 5 \times 1$ and stride 1. The second layer is a $2 \times 2$ max-pooling with stride 2. The third layer is a convolutional layer with 12 filters of size $5 \times 5 \times 20$ and stride 1. The fourth layer is a $2 \times 2$ max-pooling with stride 2. For an input image of size $28 \times 28$, this results with a $192 \times 1$ feature vector. We initialize the weights with standard normal weights, and the biases with zeros. Note that the linear transform of TDDL-LIN has 153,664 learnable parameters, while the non-linear transform of TDDL-CNN has only 6,532 learnable parameters.

TABLE I: Accuracy rates of different classification methods

| Train set size | full-CNN | TDDL-LIN | TDDL-CNN |
|---|---|---|---|
| 60K | **0.990** | 0.971 | 0.984 |
| 1K | 0.917 | 0.898 | **0.928** |

For both TDDL-based methods, we use an $\tilde{m} \times 400$ dictionary $\boldsymbol{D}$, which is initialized in the same manner. From each class, we take 40 input features (i.e. 40 $\tilde{\boldsymbol{x}}$ vectors, using the fixed initial value of $\boldsymbol{z}$ or $\boldsymbol{Z}$) to initialize an online dictionary learning (ODL) algorithm applied only on training samples from the same class. The resulted 40 atoms from each class are then collected into $\boldsymbol{D}$. This strategy has led to better performance and more discriminative sparse representations. With this initial dictionary $\boldsymbol{D}$ at hand, we obtain an initial value of $\boldsymbol{W}$ by optimizing (5) (again, $\boldsymbol{z}$ is fixed), and get a triplet of parameters $(\boldsymbol{D}, \boldsymbol{W}, \boldsymbol{z})$, which are used to initialize the task-driven DL algorithm.

We use $\lambda_1 = 0.8$ and $\lambda_1 = 4$ for TDDL-LIN and TDDL-CNN, respectively. These choices lead to an average sparsity of 25-35 for $\boldsymbol{\alpha}^*$ at the end of the task-driven DL (note that after ODL alone, the sparsity is about 10, but it grows during the TDDL, as the classification error reduces, making the representation more discriminative). The rest of the parameters are $\lambda_2 = 0$ (though the proof for differentiability [7] requires $\lambda_2 > 0$, the algorithm still converged with $\lambda_2 = 0$), $\nu_1 = 0.1$, $\nu_2 = 0$, $t_0 = 4,000$ and $\rho = 0.05$. The mini-batches size is 100.

We perform two experiments. In the first one, we use all the training data as is (i.e. 60,000 training samples) with $T = 40,000$. The second experiment reflects an extreme case, in which only 100 training samples are given per class (i.e. 1,000 training samples). The latter experiment is repeated 20 times with $T = 1,000$, each time for a different subset of 100 training samples. Table I summarizes the accuracy rates of the examined methods on the test set.

Fig. 2 presents the model parameters $\boldsymbol{W}$ ($\boldsymbol{w}_y$ for each $y \in \mathcal{Y}$) that were learned for TDDL-CNN in the first experiment. Note that the classifier emphasizes different indices of $\boldsymbol{\alpha}^*$ (i.e., using different dictionary atoms) for each class, a fact which implies that the dictionary is discriminative.

From the results in Table I, it can be concluded that TDDL-CNN outperforms TDDL-LIN. It can be also observed that TDDL-CNN outperforms the full-CNN when the labeled data is limited. This demonstrates that it may be possible to improve CNN performance by adding a data-model prior in the learning process.

ISBN 978-90-827970-1-5 © EURASIP 2018

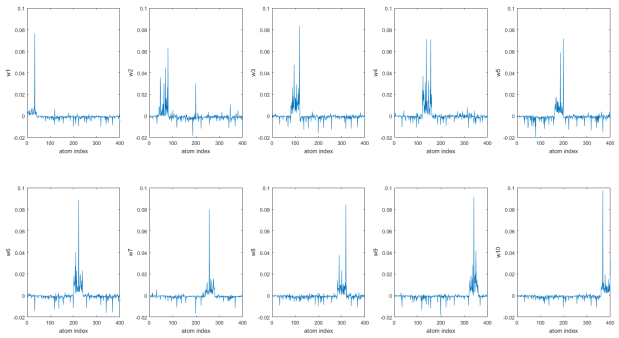

Fig. 2: The learned $\boldsymbol{w}_y$, $y \in \mathcal{Y}$, for TDDL-CNN (first exp.).

## V. Conclusion

In this work we have proposed a CNN based task-driven dictionary learning, where the parameters of the CNN are optimized simultaneously with the dictionary and a given classifier. Our method clearly outperforms task-driven dictionary learning that is restricted to a linear transform of the input data. Moreover, our work shows the potential of improving the performance of deep neural networks in the presence of a small number of training examples using the sparsity prior. It also leads to the following question: Is it possible to use the unlabeled data to improve the dictionary and further increase the performance gap between TDDL-CNN and full-CNN? One approach is using the semi-supervised formulation in [7] combined with a CNN. Yet, our efforts in this direction have led to a small improvement (e.g. the accuracy of TDDL-CNN for MNIST with 1K training samples increased from 0.928 to 0.935). Presumably, the limited improvement follows from the fact that a plain unsupervised dictionary learning makes the dictionary less discriminative. Thus, we leave the semi-supervised learning task to a future work. Such a future work may examine also other deep architectures and loss functions.

## References


[1] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM review*, vol. 51, no. 1, pp. 34–81, 2009.

[2] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Imag. Proc.*, vol. 15, no. 12, pp. 3736–3745, 2006.

[3] O. Bryt and M. Elad, "Compression of facial images using the K-SVD algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, pp. 270–282, 2008.

[4] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *ICCV*, 2009.

[5] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[6] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *ICML*, 2009.

[7] 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, 2012.

[8] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent K-SVD: Learning a discriminative dictionary for recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2651–2664, 2013.

[9] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[10] V. Papyan, Y. Romano, and M. Elad, "Convolutional neural networks analyzed via convolutional sparse coding," *Journal of Machine Learning Research*, vol. 18, no. 83, pp. 1–52, 2017.

[11] X. Sun, N. M. Nasrabadi, and T. D. Tran, "Supervised deep sparse coding networks," *arXiv:1701.08349*, 2017.

[12] Q. Qiu, X. Cheng, R. Calderbank, and G. Sapiro, "Dcfnet: Deep neural network with decomposed convolutional filters," *arXiv:1802.04145*, 2018.

[13] J. Lezama, Q. Qiu, P. Muse, and G. Sapiro, "OLÉ: Orthogonal low-rank embedding, a plug and play geometric loss for deep learning," in *CVPR*, 2018.

[14] H. Zou and T. 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.

[15] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[16] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[18] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 30–42, 2012.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2014.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[21] J. Bruna, A. Szlam, and Y. LeCun, "Signal recovery from pooling representations," in *ICML*, 2014.

[22] J. Yang, K. Yu, and T. Huang, "Supervised translation-invariant sparse coding," in *CVPR*, 2010.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[24] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[25] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for Matlab," in *ACM international conference on Multimedia*, pp. 689–692, 2015.


ISBN 978-90-827970-1-5 © EURASIP 2018