Robust and Fast Learning of Sparse Codes with Stochastic Gradient Descent

Kai Labusch, Erhardt Barth, Thomas Martinetz, *Senior Member, IEEE*

***Abstract*—Particular classes of signals, as for example natural images, can be encoded sparsely if appropriate dictionaries are used. Finding such dictionaries based on data samples, however, is a difficult optimization task. In this paper, it is shown that simple stochastic gradient descent, besides being much faster, leads to superior dictionaries compared to the Method of Optimal Directions (MOD) and the K-SVD algorithm. The gain is most significant in the difficult but relevant case of highly overlapping subspaces, i.e. when the data samples are jointly represented by a restricted set of dictionary elements. Moreover, the so-called Bag of Pursuits method is introduced as an extension of Orthogonal Matching Pursuit, and it is shown that it provides an improved approximation of the optimal sparse coefficients and, therefore, significantly improves the performance of the here proposed gradient descent as well as of the MOD and K-SVD approaches. Finally, it is shown how the Bag of Pursuits and a generalized version of the Neural Gas algorithm can be used to derive an even more powerful method for sparse coding. Performance is analyzed based on both synthetic data and the practical problem of image deconvolution. In the latter case, two different dictionaries are learned for sample images of buildings and flowers, respectively. It is demonstrated that the learned dictionaries do indeed adapt to the image class and that they therefore yield superior reconstruction results[1].**

***Index Terms*—sparse coding, neural gas, dictionary learning, matching pursuit, K-SVD, MOD**

I. Introduction

In signal processing and pattern recognition appropriate representations of given data are required. Which representation is appropriate depends on the given problem goal and on the structure of the data. Typical criteria for good representations are coding-efficiency, robustness, invariance, but also more goal-oriented criteria like the resulting classification performance. Besides such criteria, guiding principles for the design and understanding of optimal representations are useful.

One such principle, that also emerged from the neurosciences, is the principle of sparse coding. It has been derived from the observation that when the visual brain represents a particular image, only few of the many visual neurons are active. Later, this observation has been turned into an algorithm that could learn sparse representations of natural images, and the resulting representations resembled receptive fields of neurons in the primary visual cortex [1]. Besides this resemblance, however, it has been shown that the algorithm introduced in [1], which provides a sparse representation by adapting to a particular type of data, can, e.g., improve the performance of systems for hand-written digit recognition [2].

University of Lübeck, Institute for Neuro- and Bioinformatics, Ratzeburger Allee 160, 23562 Lübeck, Germany {*labusch,barth,martinetz*}*@inb.uni-luebeck.de*

[1]An example implementation of the methods that are proposed in this paper can be found at http://www.inb.uni-luebeck.de/tools-demos/ngdl

In a parallel development, ground breaking theoretical insights related to sparse coding and compressed sensing opened a completely new perspective for approaching the problem and made it a now popular topic of signal processing [3], [4], [5], [6], [7], [8], [9], [10], [11]. For a review of these developments see [12].

Many of the new results are based on approaches that assume the existence of a proper dictionary. Most often, however, such a dictionary is not known a priori and has to be learned from the given data. In this paper, we show that stochastic gradient descent can provide such dictionaries with a simple and fast algorithm. We also propose a few refinements that increase the robustness of the algorithm and the quality of the resulting dictionary.

Suppose that we are given data $X = (\mathbf{x}_1, \ldots, \mathbf{x}_L)$, $\mathbf{x}_i \in \mathbb{R}^N$ which we want to represent as a sparse linear combination of some learned dictionary C, i.e., $\mathbf{x}_i = C\mathbf{a}_i$, where $C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$, $\mathbf{c}_l \in \mathbb{R}^N$. Sparsity implies that each coefficient vector $\mathbf{a}_i$ must have only a few non-zero elements, i.e. $\|\mathbf{a}\|_0 \leq k$. The cost function for learning such a dictionary C is given by

$$E_h = \frac{1}{L} \sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \tag{1}$$

with

$$\mathbf{a}_i \quad = \quad \arg\min_{\mathbf{a}} \|\mathbf{x}_i - C\mathbf{a}\| \quad \text{subject to } \|\mathbf{a}\|_0 \leq k \tag{2}$$

denoting the best k-term representation of $\mathbf{x}_i$ in terms of C. The number of dictionary elements M and the maximum number of non-zero entries k are user-defined model parameters. In case of $M > N$, the dictionary is overcomplete.

The nested optimization problem (1) can be solved by alternating between an optimization with respect to the coefficients $\mathbf{a}_i$ and an optimization with respect to the dictionary C. First, the coefficients are determined for some fixed dictionary, and then the coefficients are fixed and the dictionary is updated. This procedure is then repeated many times. Examples of methods that use this two-fold optimization approach are, for instance, the Method of Optimal Directions (MOD) [13], the K-SVD algorithm [14] and the Sparsenet algorithm [1].

Using only data samples generated as a sparse linear combination of some given dictionary, it has been shown that methods such as MOD or K-SVD can be used to reconstruct the dictionary robustly even when the dictionary is highly overcomplete [14]. However, one limitation revealed in our experiments is, that even the K-SVD algorithm, which performed best in [14], can only reconstruct a "true" underlying dictionary for small values of k. The alternative Sparse-Coding-Neural-Gas (SCNG) algorithm proposed in [15] does not suffer

from this deficiency, but, unlike MOD or K-SVD, it is bound to a specific approximation method for the determination of the coefficients, i.e., OOMP (Optimized Orthogonal Matching Pursuit).

In this paper, we propose a simple stochastic-gradient-descent method for learning overcomplete dictionaries. Like with MOD or K-SVD, one can use an arbitrary approximation method for the determination of the coefficients during the learning process. In order to demonstrate the performance of the method, we test it on synthetically generated overcomplete linear combinations of known dictionaries, and compare the resulting performance against MOD and K-SVD. Preliminary results obtained on synthetic data were presented in [16]. Here, we report on computationally more demanding experiments that lead to even better results, and we included new experiments on synthetic data in the analysis which show that our method learns even with a smaller number of given training samples. In order to show that our method actually performs well in a real world application, we also report on new experiments in which we apply the proposed dictionary learning method to the problem of image deconvolution.

The method that is proposed in this paper is fast, since it does not involve a matrix inversion or singular value decomposition in the update of the dictionary. Furthermore, we report on an experiment on synthetic data that indicates that convergence to the "true" solution can be obtained significantly faster compared to other state-of-the-art methods. The proposed method is robust, since it performs well even if the underlying representation is less sparse (large k). Additionally, the method is able to determine the "true" solution, even if only a small number of training samples is given, in contrast to the best performing competing method of our analysis (K-SVD), which does not converge to the "true" solution at all.

II. Estimating the coefficients

Here we discuss some methods that can be used to obtain an approximation of the solution of the optimization problem (2) provided that we are given some fixed dictionary. In general, (2) is a combinatorial problem that is NP-hard [17]. However, a number of approximation methods have been proposed that tackle the problem of finding optimal coefficients $\mathbf{a}_i$ constrained by $\|\mathbf{a}_i\|_0 \leq k$ given fixed C and $\mathbf{x}_i$. Here, we consider a class of greedy methods, e.g., Matching Pursuit (MP) [18], Orthogonal Matching Pursuit (OMP) [19], and Optimized Orthogonal Matching Pursuit (OOMP) [20] that iteratively construct the vector $\mathbf{x}_i$ out of the elements of the dictionary C. It is known from [8], [10] that at least OMP and OOMP solve the optimization problem (2) exactly as long as the mutual coherence of C is small enough with respect to the sparsity of the given signal $\mathbf{x}_i$. The mutual coherence of C is given by

$$\mu(C) = \max_{1 \leq i,j \leq M \; i \neq j} \frac{|\mathbf{c}_i^T \mathbf{c}_j|}{\|\mathbf{c}_i\|_2 \|\mathbf{c}_j\|_2} \; .$$

As long as there is an $\|\mathbf{a}_i\|_0 \leq k$ with $\mathbf{x}_i = C\mathbf{a}_i$ and

$$\|\mathbf{a}_i\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(C)}\right),$$

$\mathbf{a}_i$ is the unique solution of (2) and OMP and OOMP will find it.

A. Matching Pursuit (MP)

We start with a simple approach and consider the Matching Pursuit algorithm (MP). Let $C\mathbf{a}_i^{\mathrm{MP}}$ denote the current approximation of $\mathbf{x}_i$ in MP, and let $\boldsymbol{\epsilon}_i = \mathbf{x}_i - C\mathbf{a}_i^{\mathrm{MP}}$ denote the current residual that still has to be encoded. Initially, $\mathbf{a}_i^{\mathrm{MP}} = 0$ and $\boldsymbol{\epsilon}_i = \mathbf{x}_i$. MP iteratively selects k columns of C by performing the following steps:

1) Select $\mathbf{c}_{l_{\mathrm{win}}}$ by $\mathbf{c}_{l_{\mathrm{win}}} = \arg\max_{\mathbf{c}_l} (\mathbf{c}_l^T \boldsymbol{\epsilon}_i)^2$
2) Set $(\mathbf{a}_i^{\mathrm{MP}})_{l_{\mathrm{win}}} = (\mathbf{a}_i^{\mathrm{MP}})_{l_{\mathrm{win}}} + (\mathbf{c}_{l_{\mathrm{win}}}^T \boldsymbol{\epsilon}_i)$
3) Obtain new residual $\boldsymbol{\epsilon}_i = \mathbf{x}_i - C\mathbf{a}_i^{\mathrm{MP}}$
4) Continue with step 1 until k iterations have been performed

Even if we perform N iterations of MP, i.e., if we select as many dictionary elements as there are input dimensions, it is not guaranteed that we will obtain $C\mathbf{a}_i^{\mathrm{MP}} = \mathbf{x}_i$ and $\boldsymbol{\epsilon}_i = 0$, though the asymptotical convergence of MP for $k \to \infty$ has been proven [18].

B. Orthogonal Matching Pursuit (OMP)

Let $C\mathbf{a}_i^{\mathrm{OMP}}$ denote the current approximation of $\mathbf{x}_i$ in Orthogonal Matching Pursuit. In contrast to MP, this approximation fulfills $C\mathbf{a}_i^{\mathrm{OMP}} = \mathbf{x}_i$ and $\boldsymbol{\epsilon}_i = 0$ after $k \leq N$ iterations [19]. Let U denote the set of indices of those columns of C that already have been used during Orthogonal Matching Pursuit. The number of elements in U, i.e., $|U|$, equals the number of iterations that have been performed so far. The columns of C that are indexed by U are denoted by C^U. Initially, $\mathbf{a}_i^{\mathrm{OMP}} = 0$, $\boldsymbol{\epsilon}_i = \mathbf{x}_i$ and $U = \emptyset$. OMP works as follows:

1) Select $\mathbf{c}_{l_{\mathrm{win}}}$ by $\mathbf{c}_{l_{\mathrm{win}}} = \arg\max_{\mathbf{c}_l, l \notin U} (\mathbf{c}_l^T \boldsymbol{\epsilon}_i)^2$
2) Set $U = U \cup l_{\mathrm{win}}$
3) Solve the optimization problem
$$\mathbf{a}_i^{\mathrm{OMP}} = \arg\min_{\mathbf{a}} \|\mathbf{x}_i - C^U \mathbf{a}\|_2^2$$
4) Obtain current residual $\boldsymbol{\epsilon}_i = \mathbf{x}_i - C\mathbf{a}_i^{\mathrm{OMP}}$
5) Continue with step 1 until k iterations have been performed

C. Optimized Orthogonal Matching Pursuit (OOMP)

An improved variant of the OMP algorithm is Optimized Orthogonal Matching Pursuit (OOMP) [20]. In general, the columns of C are not pairwise orthogonal. Hence, the criterion of OMP that selects the column $\mathbf{c}_{l_{\mathrm{win}}}, l_{\mathrm{win}} \notin U$ of C that is added to U is not optimal with respect to the minimization of the residual that is obtained after the column $\mathbf{c}_{l_{\mathrm{win}}}$ has been added. Therefore, Optimized Orthogonal Matching Pursuit uses a selection criterion that is optimal with respect to the minimization of the norm of the residual obtained: the algorithm runs through all columns of C that have not been used so far and selects the one that yields the smallest residual. Optimized Orthogonal Matching Pursuit works as follows:

1) Select $\mathbf{c}_{l_{\text{win}}}$ such that
$$\mathbf{c}_{l_{\text{win}}} = \arg \min_{\mathbf{c}_l, l \notin U} \min_{\mathbf{a}} \|\mathbf{x} - C^{U \cup l}\mathbf{a}\|$$
2) Set $U = U \cup l_{\text{win}}$
3) Solve the optimization problem
$$\mathbf{a}_i^{\text{OOMP}} = \arg\min_{\mathbf{a}} \|\mathbf{x}_i - C^U\mathbf{a}\|_2^2$$
4) Obtain current residual $\boldsymbol{\epsilon}_i = \mathbf{x}_i - C\mathbf{a}_i^{\text{OOMP}}$
5) Continue with step 1 until k iterations have been performed

The selection criterion of the OOMP algorithm (step 1) involves $M - |U|$ minimization problems, one for each column of C that has not been used so far. In order to reduce the computational complexity of this step, we use an implementation of the OOMP algorithm that employs a temporary dictionary R that has been orthogonalized with respect to C^U. R is obtained by removing the projection of the columns of C onto the subspace spanned by C^U from C and setting the norm of the residuals $\mathbf{r}_l$ to one. The residual $\boldsymbol{\epsilon}_i^U$ is obtained in the same way, i.e., the projection of $\mathbf{x}_i$ to the subspace spanned by C^U is removed from $\mathbf{x}_i$. Initially, $R = (\mathbf{r}_1, \ldots, \mathbf{r}_l, \ldots, \mathbf{r}_M) = C$ and $\boldsymbol{\epsilon}_i^U = \mathbf{x}_i$. In each iteration, the algorithm determines the column $\mathbf{r}_l$ of R with $l \notin U$ that has maximum overlap with respect to the current residual $\boldsymbol{\epsilon}_i^U$:

$$l_{\text{win}} = \arg\max_{l, l \notin U} (\mathbf{r}_l^T \boldsymbol{\epsilon}_i^U)^2 \,. \tag{3}$$

Then, in the construction step, the orthogonal projection with respect to $\mathbf{r}_{l_{\text{win}}}$ is removed from the columns of R and $\boldsymbol{\epsilon}_i^U$:

$$\mathbf{r}_l = \mathbf{r}_l - (\mathbf{r}_{l_{\text{win}}}^T \mathbf{r}_l)\mathbf{r}_{l_{\text{win}}}, \tag{4}$$

$$\boldsymbol{\epsilon}_i^U = \boldsymbol{\epsilon}_i^U - (\mathbf{r}_{l_{\text{win}}}^T \boldsymbol{\epsilon}_i^U)\mathbf{r}_{l_{\text{win}}} \,. \tag{5}$$

After the projection has been removed, l_{win} is added to U, i.e., $U = U \cup l_{\text{win}}$. The columns $\mathbf{r}_l$ with $l \notin U$ may be selected in the subsequent iterations of the algorithm. The norm of these columns is set to unit length. If the stopping criterion $|U| = k$ has been reached, the final entries of $\mathbf{a}_i^{\text{OOMP}}$ can be obtained by recursively collecting the contribution of each column of C during the construction process, taking into account the normalization of the columns of R in each iteration.

III. Hard-competitive stochastic gradient dictionary learning

So far, we have considered the case where a dictionary C is given. Now, we want to solve the problem of learning an optimal dictionary $C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$ from the training data $\mathbf{x}_1, \ldots, \mathbf{x}_L$ provided that we know the number of dictionary elements M as well as the dimension k of the subspaces that cover the training data. We propose a very simple way of minimizing (1). Suppose that we are given some method such as MP, OMP, or OOMP that provides an estimation $\mathbf{a}_t$ of the coefficients of a given sample $\mathbf{x}_t$ that is randomly selected from the training data at time steps $t = 0, \ldots, t_{\max}$. At each time step, we update the dictionary C according to the gradient of (1) with respect to C:

$$\Delta C = \alpha_t (\mathbf{x}_t - C\mathbf{a}_t)\mathbf{a}_t{}^T \tag{6}$$

The learning rate

$$\alpha_t = \alpha_0 \left(\frac{\alpha_{\text{final}}}{\alpha_0}\right)^{\frac{t}{t_{\max}}} \tag{7}$$

decreases exponentially. α_0 denotes the initial and α_{final} the final learning rate.

After each update, the column vectors of C are renormalized to one. Then a new training sample $\mathbf{x}_{t+1}$ is selected, the coefficients $\mathbf{a}_{t+1}$ are re-determined, and the next update for C can be performed. This simple procedure is fast, since it does not involve a singular value decomposition or a matrix inversion. Furthermore it uses only one sample in each learning step and is therefore even applicable for online-learning. It also does not require storage of a large set of training samples.

Note, that (6) can be seen as the pattern-by-pattern variant of the dictionary update rule of the Sparsenet algorithm [1] which is

$$\Delta C = \eta \, (X - CA) \, A^T \tag{8}$$

where X is a matrix that contains the training samples that are considered in the current batch update and A are the coefficients that have been estimated with respect to these training samples. Here, η is a constant learning rate.

IV. Experiments with hard-competitive stochastic gradient descent

In the experiments we use synthetic data that actually can be represented as sparse linear combinations of some dictionary. We perform the experiments in order to asses two questions: (i) How good is the target function (1) minimized? (ii) Is it possible to obtain the true underlying dictionary only from the given data?

In the following $C^{\text{true}} = (\mathbf{c}_1^{\text{true}}, \ldots, \mathbf{c}_{50}^{\text{true}}) \in \mathbb{R}^{20 \times 50}$ denotes a synthetic dictionary. Each entry of C^{true} is uniformly chosen in the interval $[-0.5, 0.5]$. Furthermore, we set $\|\mathbf{c}_l^{\text{true}}\| = 1$. Using such a dictionary, we create a training set $X = (\mathbf{x}_1, \ldots, \mathbf{x}_{1500})$, $\mathbf{x}_i \in \mathbb{R}^{20}$ where each training sample $\mathbf{x}_i$ is a sparse linear combination of the columns of the dictionary:

$$\mathbf{x}_i = C^{\text{true}}\mathbf{b}_i \,. \tag{9}$$

We choose the coefficient vectors $\mathbf{b}_i \in \mathbb{R}^{50}$ such that they contain k non-zero entries. The selection of the position of the non-zero entries in the coefficient vectors is performed according to three different data generation scenarios:

- **Random dictionary elements:** In this scenario all combinations of k dictionary elements are possible. Hence, the position of the non-zero entries in each coefficient vector $\mathbf{b}_i$ is uniformly chosen in the interval $[1, \ldots, 50]$.
- **Independent Subspaces:** In this case the training samples are located in a small number of k-dimensional subspaces. We achieve this by defining $\lfloor 50/k \rfloor$ groups of dictionary elements, each group containing k randomly

selected dictionary elements. The groups do not intersect, i.e., each dictionary element is at most member of one group. In order to generate a training sample, we uniformly choose one group of dictionary elements and obtain the training sample as a linear combination of the dictionary elements that belong to the selected group.

- **Dependent subspaces:** In this case, similar to the previous scenario, the training samples are located in a small number of k-dimensional subspaces. In contrast to the previous scenario, the subspaces do highly intersect, i.e., the subspaces share basis vectors. In order to achieve this, we uniformly select $k-1$ dictionary elements. Then, we use $50-k+1$ groups of dictionary elements where each group consists of the $k-1$ selected dictionary elements plus one further dictionary element. Again, in order to generate a training sample, we uniformly choose one group of dictionary elements and obtain the training sample as a linear combination of the dictionary elements that belong to the selected group.

The value of the non-zero entries is always chosen uniformly in the interval $[-0.5, 0.5]$. Finally the data was scaled such that the mean variance was equal to 1.

We apply MOD, K-SVD and the stochastic gradient descent method that is proposed in this paper to the training data. In case of MOD and K-SVD, we use the implementations that are provided by the authors of [14].

Let $C^{\text{learned}} = (\mathbf{c}_1^{\text{learned}}, \ldots, \mathbf{c}_{50}^{\text{learned}})$ denote the dictionary that has been learned by one of these methods on the basis of the training samples. In order to measure the performance of the methods with respect to the minimization of the target function, we consider

$$E_h = \frac{1}{1500} \sum_{i=1}^{1500} \|\mathbf{x}_i - C^{\text{learned}} \mathbf{a}_i\|_2^2 \tag{10}$$

where $\mathbf{a}_i$ is obtained from the optimized orthogonal matching pursuit algorithm. In order to asses if the true dictionary can be reconstructed from the training data, we consider the mean maximum overlap between each element of the true dictionary and the learned dictionary:

$$MMO = \frac{1}{50} \sum_{l=1}^{50} \max_{k=1,\ldots,50} |(\mathbf{c}_l^{\text{true}})^T \mathbf{c}_k^{\text{learned}}| \,. \tag{11}$$

k, the number of non-zero entries is varied from 1 to 11. For the stochastic gradient descent method, we perform 100×1500 update steps, i.e., 100 learning epochs. For MOD and K-SVD, we perform 100 learning iterations, each iteration using 1500 training samples. Note, that this leads to the same computational demand for all the methods used. We repeat all experiments 50 times and report the mean result over all experiments. For all dictionary learning methods, i.e., MOD, K-SVD, and the stochastic gradient descent, optimized orthogonal matching pursuit is used in order to obtain the dictionary coefficients during learning.

The results of this experiment are depicted in Figure 1. In case of the random dictionary elements scenario (see (a) and (d)) the stochastic gradient approach clearly outperforms MOD and K-SVD. From the mean maximum overlap (see (d)) it can be seen that almost all dictionary elements are well reconstructed with up to 6 non-zero coefficients in the linear combinations. If the dictionary elements cannot be reconstructed any more, i.e., for $k > 6$, the mean representation error E_h starts to grow (see (a)). In case of the independent subspaces ((b) and (e)) and dependent subspaces scenario ((c) and (f)) the stochastic gradient method also outperforms MOD and K-SVD in terms of minimization of the representation error (see (b) and (c)), whereas in terms of dictionary reconstruction performance only in the intersecting subspaces scenario a clear performance gain compared to MOD and K-SVD can be seen (see (c) and (f)). This might be caused by the fact that in case of the independent subspaces scenario it is sufficient to find dictionary elements that span the subspaces where the data is located in order to minimize the target function, i.e., the scenario does not force the method to find the true dictionary elements in order to minimize the target function.

V. A BAG OF PURSUITS

In order to further improve performance, we now consider an extension of OOMP that not only determines a single approximation of the best coefficients but that determines K_{user} good approximations of the best coefficients. This method is called "bag of pursuits" (BOP), since it performs a sequence of optimzed orthogonal matching pursuits. The algorithm starts with $U_n^j = \emptyset$, $R_0^j = (\mathbf{r}_1^{0,j}, \ldots, \mathbf{r}_M^{0,j}) = C$ and $\boldsymbol{\epsilon}_0^j = \mathbf{x}$. The set U_n^j contains the indices of those columns of C that have been used during the j-th pursuit with respect to $\mathbf{x}$ up to the n-th iteration. R_n^j is a temporary matrix that has been orthogonalized with respect to the columns of C that are indexed by U_n^j. $\mathbf{r}_l^{n,j}$ is the l-th column of R_n^j. $\boldsymbol{\epsilon}_n^j$ is the residual in the n-th iteration of the j-th pursuit with respect to $\mathbf{x}$.

In iteration n, the algorithm looks for that column of R_n^j whose inclusion in the linear combination leads to the smallest residual $\boldsymbol{\epsilon}_{n+1}^j$ in the next iteration of the algorithm, i.e., that has the maximum overlap with respect to the current residual. Hence, with

$$\mathbf{y}_n^j = \left(\frac{{\mathbf{r}_1^{n,j}}^T \boldsymbol{\epsilon}_n^j}{\|\mathbf{r}_1^{n,j}\|}, \ldots, \frac{{\mathbf{r}_l^{n,j}}^T \boldsymbol{\epsilon}_n^j}{\|\mathbf{r}_l^{n,j}\|}, \ldots, \frac{{\mathbf{r}_M^{n,j}}^T \boldsymbol{\epsilon}_n^j}{\|\mathbf{r}_M^{n,j}\|} \right) \tag{12}$$

it looks for $l_{\text{win}}(n,j) = \arg\max_{l, l \notin U_n^j} (\mathbf{y}_n^j)_l^2$. Then, the orthogonal projection of R_n^j to $\mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}$ is removed from R_n^j

$$R_{n+1}^j = R_n^j - \frac{\mathbf{r}_{l_{\text{win}}(n,j)}^{n,j} ({R_n^j}^T \mathbf{r}_{l_{\text{win}}(n,j)}^{n,j})^T}{{\mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}}^T \mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}} \,. \tag{13}$$

Furthermore, the orthogonal projection of $\boldsymbol{\epsilon}_n^j$ to $\mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}$ is removed from $\boldsymbol{\epsilon}_n^j$

$$\boldsymbol{\epsilon}_{n+1}^j = \boldsymbol{\epsilon}_n^j - \frac{{\boldsymbol{\epsilon}_n^j}^T \mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}}{{\mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}}^T \mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}} \mathbf{r}_{l_{\text{win}}(n,j)}^{n,j} \,. \tag{14}$$

The algorithm stops if $\|\boldsymbol{\epsilon}_n^j\| = 0$ or $n = k$. The j-th approximation of the coefficients of the best k-term approximation, i.e., $\mathbf{a}^j$, can be obtained by recursively tracking

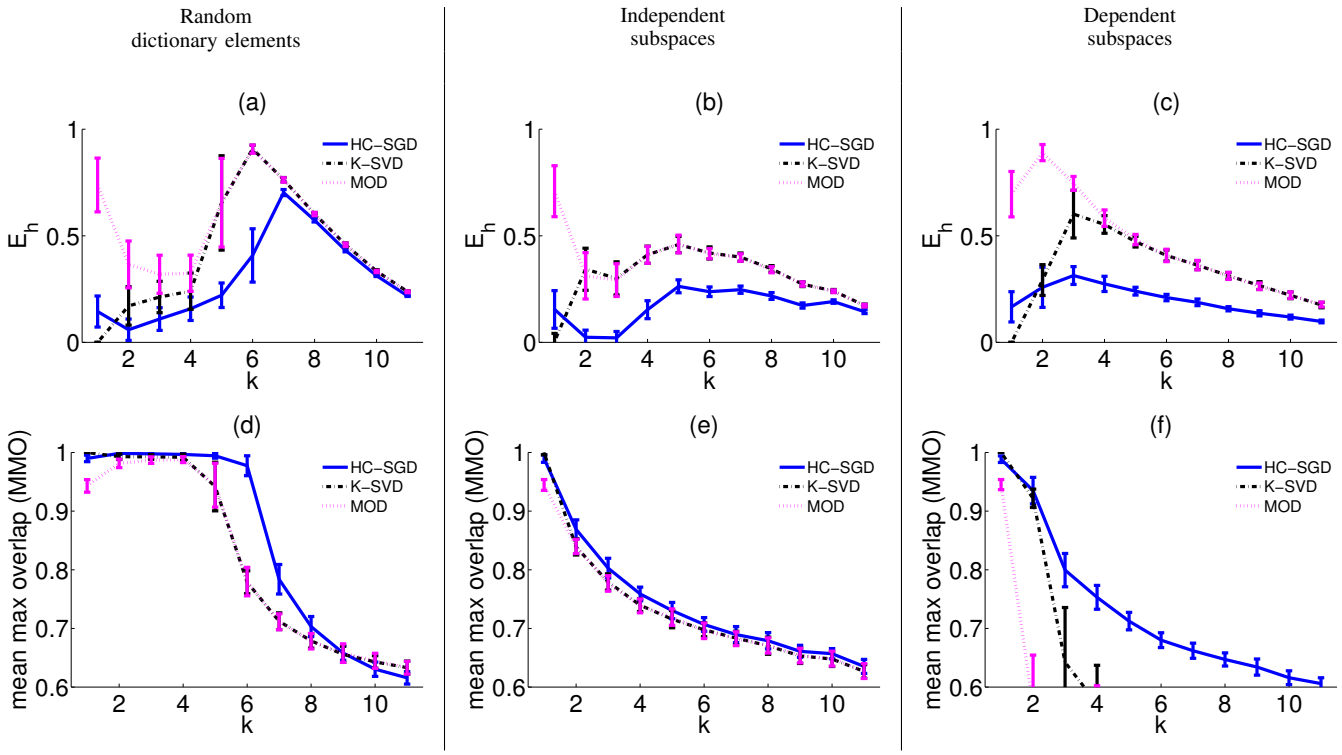

Fig. 1. Experimental results for hard-competitive stochastic gradient descent. The figures show the obtained final reconstruction error E_h and the obtained final mean maximum overlap (MMO) between learned and "true" dictionary. See text for details. HC-SGD: hard-competitive stochastic gradient descent ($\alpha_0 = 0.1$, $\alpha_{\text{final}} = 10^{-3}$). MOD and K-SVD: 100 learning iterations each using 1500 training samples. During learning, for all methods the coefficients were obtained from OOMP. All experiments were repeated 50 times.

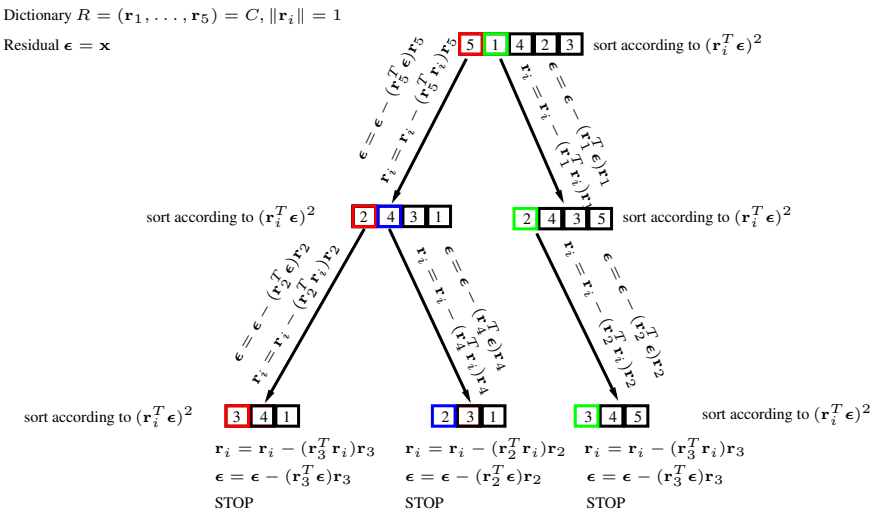

Fig. 2. The Figure depicts the tree-like search procedure of the BOP method if the constraint $\|\mathbf{a}\|_0 \leq 3$ is in place. In this example $K_{\text{user}} = 3$ holds, i.e., the method determines three different solutions. The method starts by sorting the dictionary elements according to their overlap with respect to the residual (root of the tree). The dictionary element that has the largest overlap, i.e., element 5, is selected. All other dictionary elements as well as the residual are orthogonalized with respected to dictionary element 5. This procedure is repeated (elements 2 and 3 are selected) until at most three dictionaries elements have been used. Now, the second solution is determined. Among all overlaps that have been computed so far, the largest one is selected (element 1 at root level). Again, a sequence of orthogonalizations is performed, until three dictionary elements have been used (elements 2 and 3 are selected). The third solution is obtained by repeating the entire procedure again. Solution 1: 5,2,3 Solution 2: 1,2,3 Solution 3: 5,4,2.

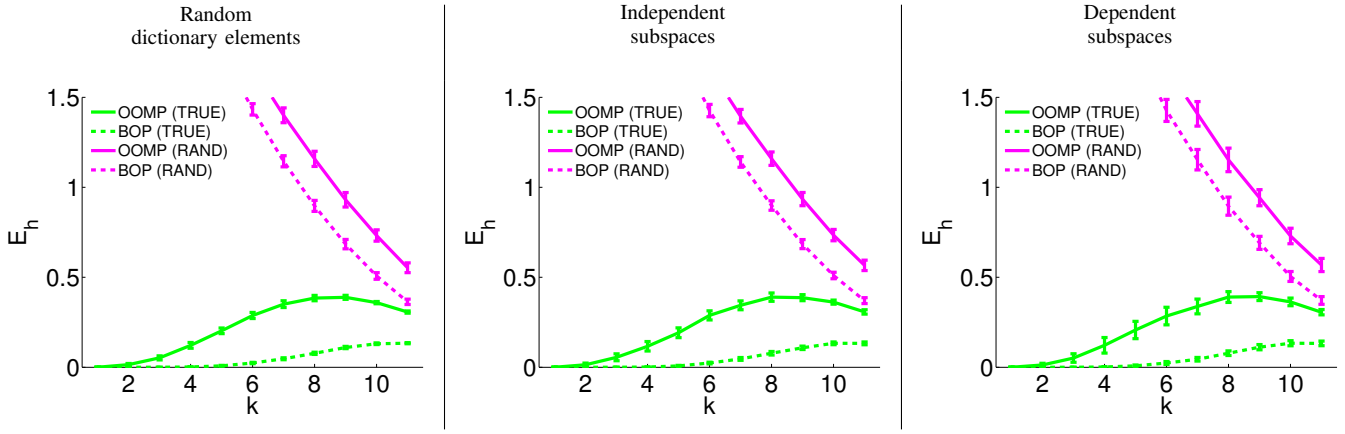

Fig. 3. Comparison of the mean reconstruction error E_h on the data that has been described in Section IV. Both sparse approximation methods, i.e, OOMP and BOP, were provided with the "true" dictionary the data was generated from. It can be seen that if the "true" dictionary is known, OOMP finds the "true" solution only for $k = 1$ whereas BOP with $K_{user} = 50$ finds the correct solution up to $k = 5$. For comparison purposes we also show the mean reconstruction error both methods achieve with a random dictionary. All experiments were repeated 50 times.

the contribution of each column of C that has been used during the iterations of pursuit j. In order to obtain a set of approximations $\mathbf{a}^1, ..., \mathbf{a}^{K_{\text{user}}}$, where K_{user} is chosen by the user, we want to conduct K_{user} matching pursuits. To obtain K_{user} different pursuits, we implement the following function:

$$Q(l,n,j) = \begin{cases} 0: & \text{If there is no pursuit among all pursuits that have been performed with respect to } \mathbf{x} \text{ that is equal to the } j\text{-th pursuit up to the } n\text{-th iteration where in that iteration column } l \text{ has been selected}^2 \\ 1: & \text{else .} \end{cases} \quad (15)$$

Then, while a pursuit is performed, we track all overlaps $\mathbf{y}_n^j$ that have been computed during that pursuit. For instance, if $\mathbf{a}^1$ has been determined, we have $\mathbf{y}_0^1, \ldots, \mathbf{y}_n^1, \ldots, \mathbf{y}_{s_1-1}^1$ where s_1 is the number of iterations of the 1st pursuit with respect to $\mathbf{x}$. In order to find $\mathbf{a}^2$, we now look for the largest overlap in the previous pursuit that has not been used so far

$$n_{\text{target}} = \arg \max_{n=0,\ldots,s_1-1} \max_{l,Q(l,n,j)=0} (\mathbf{y}_n^1)_l^2 \quad (16)$$

$$l_{\text{target}} = \arg \max_{l} (\mathbf{y}_{n_{\text{target}}}^1)_l^2 \, . \quad (17)$$

We replay the 1st pursuit up to iteration n_{target}. In that iteration, we select column l_{target} instead of the previous winner and continue with the pursuit, until the stopping criterion has been reached. If m pursuits have been performed, among all previous pursuits, we look for the largest overlap that has not been used so far:

$$j_{\text{target}} = \arg \max_{j=1,\ldots,m} \max_{n=0,\ldots,s_j-1} \max_{l,Q(l,n,j)=0} (\mathbf{y}_n^j)_l^2 \quad (18)$$

$$n_{\text{target}} = \arg \max_{n=0,\ldots,s_{j_{\text{target}}}-1} \max_{l,Q(l,n,j_{\text{target}})=0} (\mathbf{y}_n^{j_{\text{target}}})_l^2 \quad (19)$$

$$l_{\text{target}} = \arg \max_{l,Q(l,n_{\text{target}},j_{\text{target}})=0} (\mathbf{y}_{n_{\text{target}}}^{j_{\text{target}}})_l^2 \, . \quad (20)$$

[2]This means, that two pursuits are considered to be equal if in both pursuits the same set of dictionary elements has been selected up to the n-th iteration. The order in which the elements have been selected does not matter.

We replay pursuit j_{target} up to iteration n_{target}. In that iteration, we select column l_{target} instead of the previous winner and continue with the pursuit until the stopping criterion has been reached. We repeat this procedure until K_{user} pursuits have been performed. A schematic view of the BOP method is shown in Figure 2.

Since in this paper overcomplete dictionaries are considered, the size of the dictionary can become very large. If the dictionary is very large, the orthogonalization of the entire temporary dictionary with respect to the element of the dictionary that has been used in the current step, i.e., (13), can be computationally expensive. A computationally less demanding strategy is to omit the orthogonalization of the dictionary, i.e., to perform the same tree like search procedure where OMP solutions are computed instead of OOMP solutions.

VI. Experiments using BOP

First, we tested how far BOP improves OOMP. For that purpose, we again used our synthetical data from Section IV, however, this time the dictionary needs not to be learned but the true dictionary is assumed to be known. Then, only the best coefficients for reconstruction have to be determined. This is done with OOMP and, for comparison, with BOP. The result with respect to the reconstruction error both methods achieve is shown in Figure 3. OOMP succeeds in finding the correct coefficients only in case $k < 2$. Perfect reconstruction for $k = 1$ can be expected from the theoretical result in [8], [10], since the average mutual coherence of our synthetical dictionaries is 0.7 ± 0.05. In contrast, BOP is able to obtain perfect reconstruction even up to $k = 4$ or almost $k = 5$. Since the average mutual coherence is the same in all three data generation scenarios, the performance of BOP and OOMP is almost equal for all the scenarios, if the "true" dictionary is known or a random dictionary is used for the coefficient estimation. This is in contrast to the experiments where the dictionary has to be learned from the training data, and where the scenario has an influence on the ability of the learning method to determine the "true" dictionary (as can be seen for instance in Figure 1).

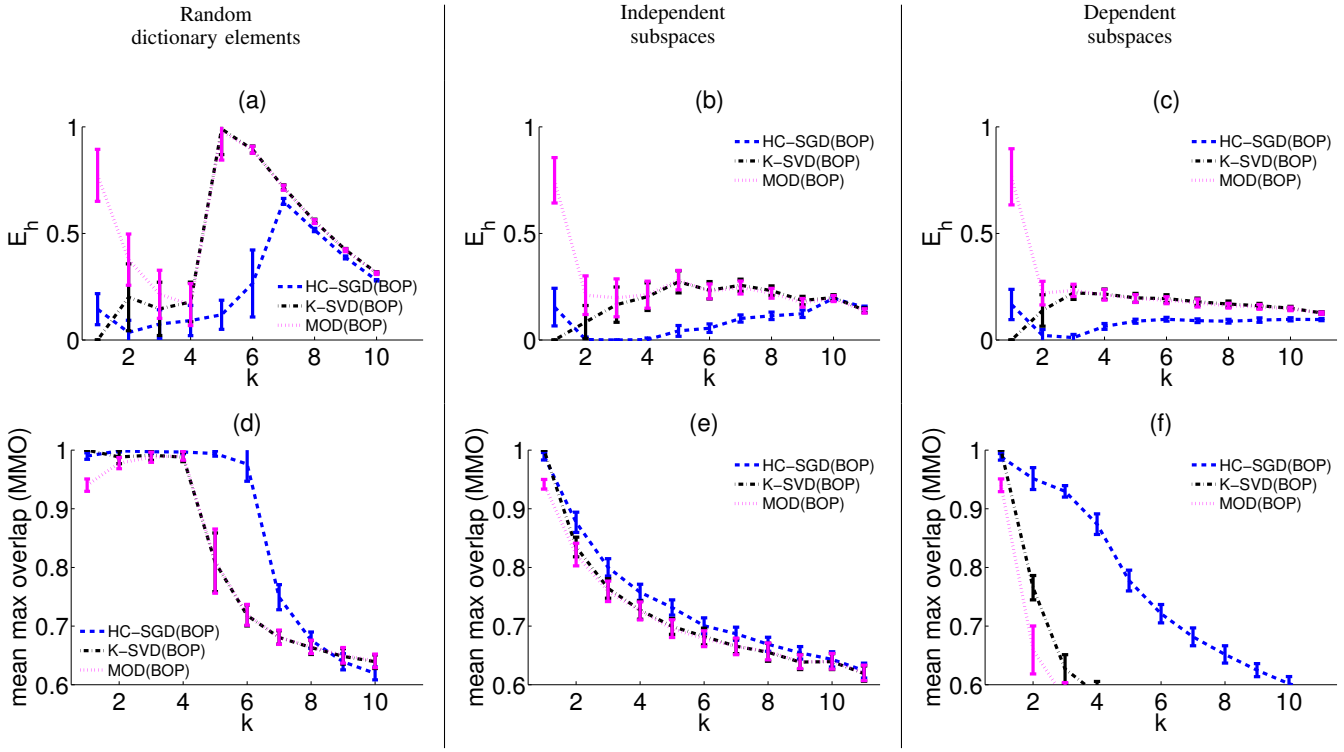

Fig. 4. Experimental results for MOD, K-SVD, and hard-competitive stochastic gradient descent. All methods employed the BOP algorithm with $K_{\text{user}=50}$ in order to estimate the dictionary coefficients in the learning process. See text for details. HC-SGD(BOP): hard-competitive stochastic gradient descent ($\alpha_0 = 0.1$, $\alpha_{\text{final}} = 10^{-3}$). MOD and K-SVD: 100 learning iterations, each iteration using 1500 training samples. During learning for all methods the coefficients were obtained from the BOP method with $K_{\text{user}} = 50$. All experiments were repeated 50 times.

In Figure 4, we show the results we obtain with BOP in the experiment described in Section IV. Now, for all methods the dictionary coefficients were obtained from the best pursuit out of $K_{\text{user}} = 50$ pursuits that were performed according to the "Bag of Pursuits" approach described in Section V. Compared to the results of the first experiment (see Figure 1) it can be seen that the computationally more demanding method for the approximation of the best coefficients leads to a significantly improved performance of MOD, K-SVD and the stochastic gradient descent with respect to the minimization of the representation error E_h (see (a)-(c)). The most obvious improvement can be seen in case of the dependent subspaces scenario where also the dictionary reconstruction performance significantly improves (see (c) and (f)). In the random dictionary elements (see (a) and (d)) and independent subspaces scenario (see (b) and (e)) there are only small improvements with respect to the reconstruction of the true dictionary.

VII. Soft-competitive stochastic gradient descent

In the experiments in Section VI, we have shown that the learning performance can be significantly improved by using the best coefficients from the set of solutions that are provided by the BOP method. So far, only the best pursuit from a set of K_{user} pursuits has been used in the stochastic gradient descent. If a sparse approximation method such as BOP is used that subsequently improves on a set of intermediate solutions towards the final solution, it is desireable not only to use the final solution in the learning process but also to employ the information that might be contained in the set of intermediate solutions for the dictionary learning process. If possible, this should be implemented such that the use of the intermediate solutions does not introduce additional computational demand.

We now propose a soft-competitive learning strategy that uses a set of approximations in order to solve the optimization problem (1). The soft-competitive approach is motivated from the domain of vector quantization. Vector quantization can be understood as a special case of sparse coding where the coefficients of the dictionary are constrained according to $\|\mathbf{a}_i\|_0 = 1$ and $\|\mathbf{a}_i\|_2 = 1$. In vector quantization the coefficients are chosen such that $(\mathbf{a}_i)_m = 1$, $(\mathbf{a}_i)_l = 0\ \forall l \neq m$ where $m = \arg\min_l \|\mathbf{c}_l - \mathbf{x}_i\|_2^2$. Many vector quantization algorithms consider only the winner for learning, i.e., the dictionary vector $\mathbf{c}_m$ for which $(\mathbf{a}_i)_m = 1$ holds. As a consequence of that type of hard-competitive learning scheme, problems such as bad quantization, initialization sensitivity, or slow convergence can arise.

In order to remedy these problems, soft-competitive vector quantization methods such as the NG algorithm [21], [22] have been proposed. In the Neural Gas algorithm all possible encodings are considered in each learning step, i.e., $\mathbf{a}_i^1, \ldots, \mathbf{a}_i^M$ with $(\mathbf{a}_i^j)_j = 1$. Then, the encodings are sorted according to their reconstruction error

$$\|\mathbf{x}_i - C\mathbf{a}_i^{j_0}\| \leq \cdots \leq \|\mathbf{x}_i - C\mathbf{a}_i^{j_p}\| \leq \cdots \leq \|\mathbf{x}_i - C\mathbf{a}_i^{j_M}\| \,. \tag{21}$$

In contrast to the hard-competitive approaches, in each learning iteration, every codebook vector $\mathbf{c}_l$ is updated. The update is weighted according to the rank of the encoding that uses the

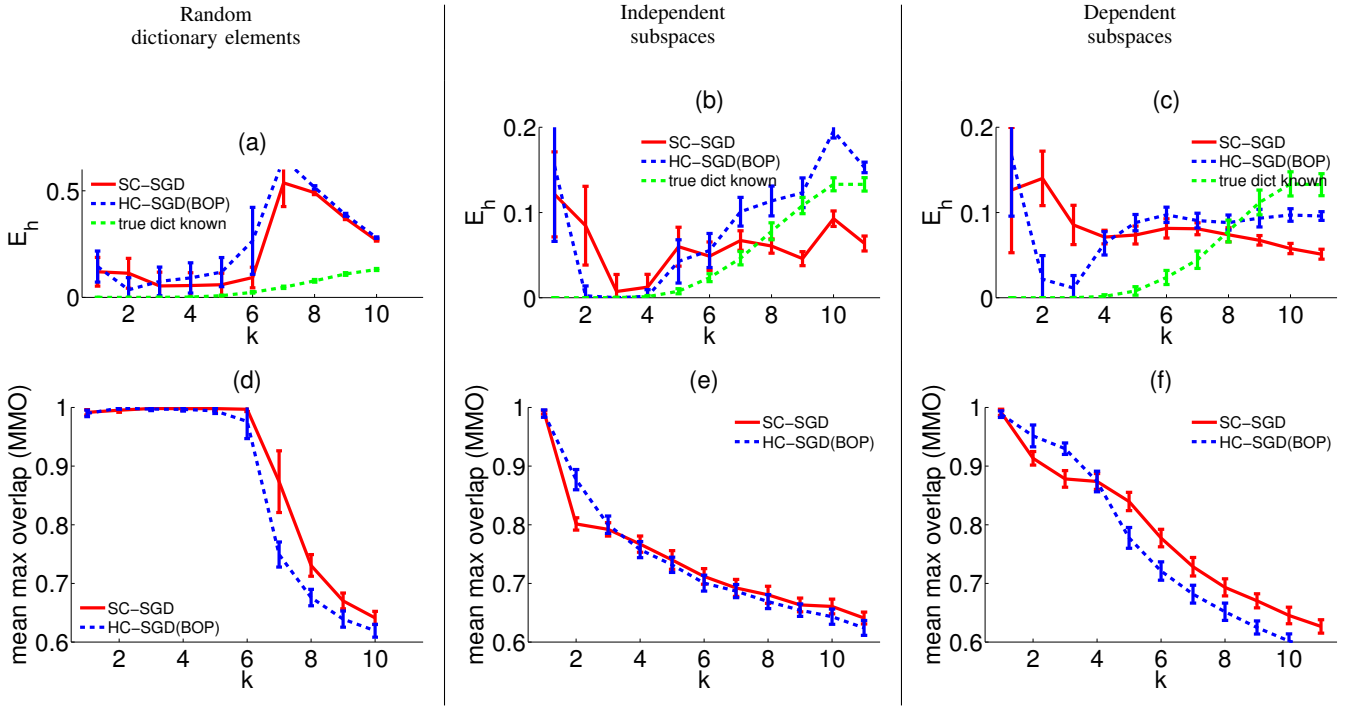

Fig. 5. Experimental comparison of hard- and soft-competitive stochastic gradient descent. See text for details. HC-SGD(BOP): hard-competitive stochastic gradient descent ($\alpha_0 = 0.1$, $\alpha_{\text{final}} = 10^{-3}$). SC-SGD: soft-competitive stochastic gradient descent ($\alpha_0 = 0.1$, $\alpha_{\text{final}} = 10^{-3}$, $\lambda_0 = 50$, $\lambda_{\text{final}} = 0.1$). All experiments were repeated 50 times. In the random dictionary elements and dependent subspaces scenarios further performance improvements with respect to the dictionary reconstruction can be observed in the soft-competitive case. For $k > 6$ a significant decrease of the mean reconstruction error is obtained. In the dependent and independent subspace scenarios, for $k > 8$ the mean reconstruction error obtained from the soft-competitive approach is even smaller than the mean reconstruction error obtained by using the "true" dictionary. This indicates that in these scenarios the methods are not forced to learn the true dictionary but learn a dictionary that spans the subspaces where the data is located. Obviously there is a solution that is more incoherent than the true dictionary that still spans these subspaces. All experiments were repeated 50 times.

codebook vector $\mathbf{c}_l$. It has been shown in [22] that this type of update is equivalent to a gradient descent on a well-defined cost function. Due to the soft-competitive learning scheme the NG algorithm shows robust convergence to close to optimal distributions of the codebook vectors over the data manifold.

Here we want to apply this ranking approach to the learning of sparse codes. Similar to the NG algorithm, for each given sample $\mathbf{x}_i$, we consider all K possible coefficient vectors $\mathbf{a}_i^j$, i.e., encodings that have at most k non-zero entries. The elements of each $\mathbf{a}_i^j$ are chosen such that $\|\mathbf{x}_i - C\mathbf{a}_i^j\|$ is minimal. We order the coefficients according to the representation error that is obtained by using them to approximate the sample $\mathbf{x}_i$

$$\|\mathbf{x}_i - C\mathbf{a}_i^{j_0}\| < \cdots < \|\mathbf{x}_i - C\mathbf{a}_i^{j_p}\| < \cdots < \|\mathbf{x}_i - C\mathbf{a}_i^{j_K}\| \,. \tag{22}$$

If there are coefficient vectors that lead to the same reconstruction error

$$\|\mathbf{x}_i - C\mathbf{a}_i^{m_1}\| = \|\mathbf{x}_i - C\mathbf{a}_i^{m_2}\| = \cdots = \|\mathbf{x}_i - C\mathbf{a}_i^{m_V}\| \,, \tag{23}$$

we randomly pick one of them and do not consider the others. Note that we need this due to theoretical considerations while in practice this situation almost never occurs. Let $\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C) = p$ denote the number of coefficient vectors $\mathbf{a}_i^m$ with $\|\mathbf{x}_i - C\mathbf{a}_i^m\| < \|\mathbf{x}_i - C\mathbf{a}_i^j\|$. Introducing the neighborhood $h_{\lambda_t}(v) = e^{-v/\lambda_t}$, we consider the following modified error function

$$E_s = \sum_{i=1}^{L}\sum_{j=1}^{K} h_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C))\|\mathbf{x}_i - C\mathbf{a}_i^j\|_2^2 \tag{24}$$

which becomes equal to (1) for $\lambda_t \to 0$. In order to minimize (24), we consider the gradient of E_s with respect to C, which is

$$\frac{\partial E_s}{\partial C} = -2\sum_{i=1}^{L}\sum_{j=1}^{K} h_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C))(\mathbf{x}_i - C\mathbf{a}_i^j)\mathbf{a}_i^{jT} + R \tag{25}$$

with

$$\begin{aligned} R \;=\; & \sum_{i=1}^{L}\sum_{j=1}^{K} h'_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C)) \\ & \cdot \frac{\partial \text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C)}{\partial C}\|\mathbf{x}_i - C\mathbf{a}_i^j\|_2^2 \,. \end{aligned} \tag{26}$$

In order to show that $R = 0$, we adopt the proof given in [22] to our setting. With $\mathbf{e}_i^j = \mathbf{x}_i - C\mathbf{a}_i^j$, we write $\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C)$ as

$$\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C) = \sum_{m=1}^{K} \theta((\mathbf{e}_i^j)^2 - (\mathbf{e}_i^m)^2) \tag{27}$$

where $\theta(x)$ is the Heaviside step function. The derivative of the Heaviside step function is the delta distribution $\delta(x)$ with $\delta(x) = 0$ for $x \neq 0$ and $\int \delta(x)dx = 1$. Therefore, we can

write

$$R = 2\sum_{i=1}^{L}\sum_{j=1}^{K} h'_{\lambda_t}(\mathrm{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C))(\mathbf{e}_i^j)^2 \cdot \sum_{m=1}^{K}((\mathbf{e}_i^m)(\mathbf{a}_i^m)^T - (\mathbf{e}_i^j)(\mathbf{a}_i^j)^T)\delta((\mathbf{e}_i^j)^2 - (\mathbf{e}_i^m)^2) \quad (28)$$

Each term of (28) is non-vanishing only for those $\mathbf{a}_i^j$ for which $(\mathbf{e}_i^j)^2 = (\mathbf{e}_i^m)^2$ is valid. Since we explicitly excluded this case, we obtain $R = 0$. Hence, we can perform a stochastic gradient descent on (24) with respect to C by applying $t = 0, \ldots, t_{\max}$ updates of C using the gradient based learning rule

$$\Delta C = \alpha_t \sum_{j=1}^{K} h_{\lambda_t}(\mathrm{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C))(\mathbf{x}_i - C\mathbf{a}_i^j)\mathbf{a}_j^{i\,T} \quad (29)$$

for a randomly chosen $\mathbf{x}_i \in X$ where

$$\lambda_t = \lambda_0 \left(\frac{\lambda_{\mathrm{final}}}{\lambda_0}\right)^{\frac{t}{t_{\max}}} \quad (30)$$

is an exponentially decreasing neighborhood-size. Again, α_t is an exponentially decreasing learning rate (see equation (7)). After each update, the column vectors of C are renormalized to one. Then the $\mathbf{a}_i^j$ are re-determined and the next update for C can be performed.

So far, for each training sample $\mathbf{x}_i$, all possible coefficient vectors $\mathbf{a}_i^j$, $j = 1, \ldots, K$ with $\|\mathbf{a}_i^j\|_0 \leq k$ have been considered. K grows exponentially with M and k. Therefore, this approach is not applicable in practice. However, since in (24) all those contributions in the sum for which the rank is larger than the neighborhood-size λ_t can be neglected, we actually do not need all possible coefficient vectors. We only need the first best ones with respect to the reconstruction error. These are directly provided by the BOP method, at least approximately.

VIII. Experiments soft-competitive stochastic gradient descent

In the third experiment, we employed soft-competitive learning in the stochastic gradient descent, i.e., the coefficients corresponding to each of the $K_{\mathrm{user}} = 50$ pursuits were used in the update step according to (29) with initial neighbourhood-size $\lambda_0 = 50$ and final neighbourhood-size $\lambda_{\mathrm{final}} = 0.1$. The results of this experiment are depicted in Figure 5. It can be seen that for less sparse scenarios, i.e. $k > 6$, the soft-competitive learning further improves the performance. Particularly in case of the dependent subspaces scenario a significant improvement in terms of dictionary reconstruction performance can be seen for $k > 4$ (see (f)). For very sparse settings, i.e. $k \leq 4$, the hard-competitive approach seems to perform better than the soft-competitive variant. Again, in case of the independent subspaces only the representation error decreases (see (b)) whereas no performance gain for the dictionary reconstruction can be seen (see (e)). Again, this might be caused by the fact that in case of the subspace scenario it is sufficient to learn dictionary elements that span the subspaces in order to minimize the target function.

Finally, we evaluated the influence of both, the number of given training data and performed training iterations/epochs, on the ability of the methods to estimate the underlying "true" dictionary. We again applied the soft-competitive stochastic gradient descent approach and K-SVD to the synthetic data, but now we varied the number of given training samples and the number of training iterations/epochs. We only used K-SVD for the comparison, since it performed best among the non-gradient methods in the previous experiments. The number of non-zero coefficients in the linear combinations was set to 5. The results of the experiments are shown in Figure 6. In the random dictionary elements scenario and the dependent subspaces scenario, the dictionary reconstruction performance of the stochastic gradient descent method can be significantly improved by an increase of the number of given training samples (see (a) and (c)). In contrast to this, the performance of the K-SVD method increases only marginally in these cases (see (d) and (f)). Furthermore, in the random dictionary elements scenario with 1000 training samples, with the stochastic gradient descent method the dictionary is obtained after approximately 40 training epochs. For the gradient descent method, the convergence to the "true" solution can be seen even if only 500 training samples are given (see (a)). In contrast to that, K-SVD does not converge at all to the "true" solution (see (d)).

In the independent subspaces scenario, the dictionary reconstruction performance of the gradient descent method can also be improved by an increase of the number of given training samples (see (b)). However, in this case, the results indicate that it is not possible to estimate the dictionary with arbitrary accuracy, i.e., due to the structure of the subspaces. At some point, the method has learned a set of dictionary elements that span the subspaces and a further increase of the number of training samples does not lead to a better estimation of the "true" dictionary. Again, convergence to the "true" solution cannot be seen for the K-SVD method as well (see (e)).

IX. Application to image deconvolution

In the following experiments, we demonstrate how to deconvolve a given image by employing dictionaries that have been learned from a set of training images that are similar to the image that is to be deconvolved.

Let a vector $\mathbf{q}_i$ from the set of vectors $\mathbf{q}_1, \ldots, \mathbf{q}_{N_q}$, $\mathbf{q}_i \in \mathbb{R}^D$ be a patch of size $d \times d$, $d^2 = D$ at position i in the image I_q. The image I_q has been obtained by applying a known convolution operator Q to the primal image I_p:

$$\mathbf{q}_i = Q\mathbf{p}_i \; i = 1, \ldots, N_q \; . \quad (31)$$

N_q denotes the number of valid patches of size $d \times d$ of the given image I_q. For computational reasons, we only considered those patches in the experiments that are located at even coordinates. Q is a matrix that describes the convolution operator. A vector $\mathbf{p}_i \in \mathbb{R}^N$ is a patch of size $n \times n$, $n^2 = N$ at position i in the primal image I_p that is to be reconstructed. In order to compute the primal image from the convolved image the operator Q has to be inverted. However, if $D < N$ holds, generally Q cannot be inverted. Nevertheless, the problem

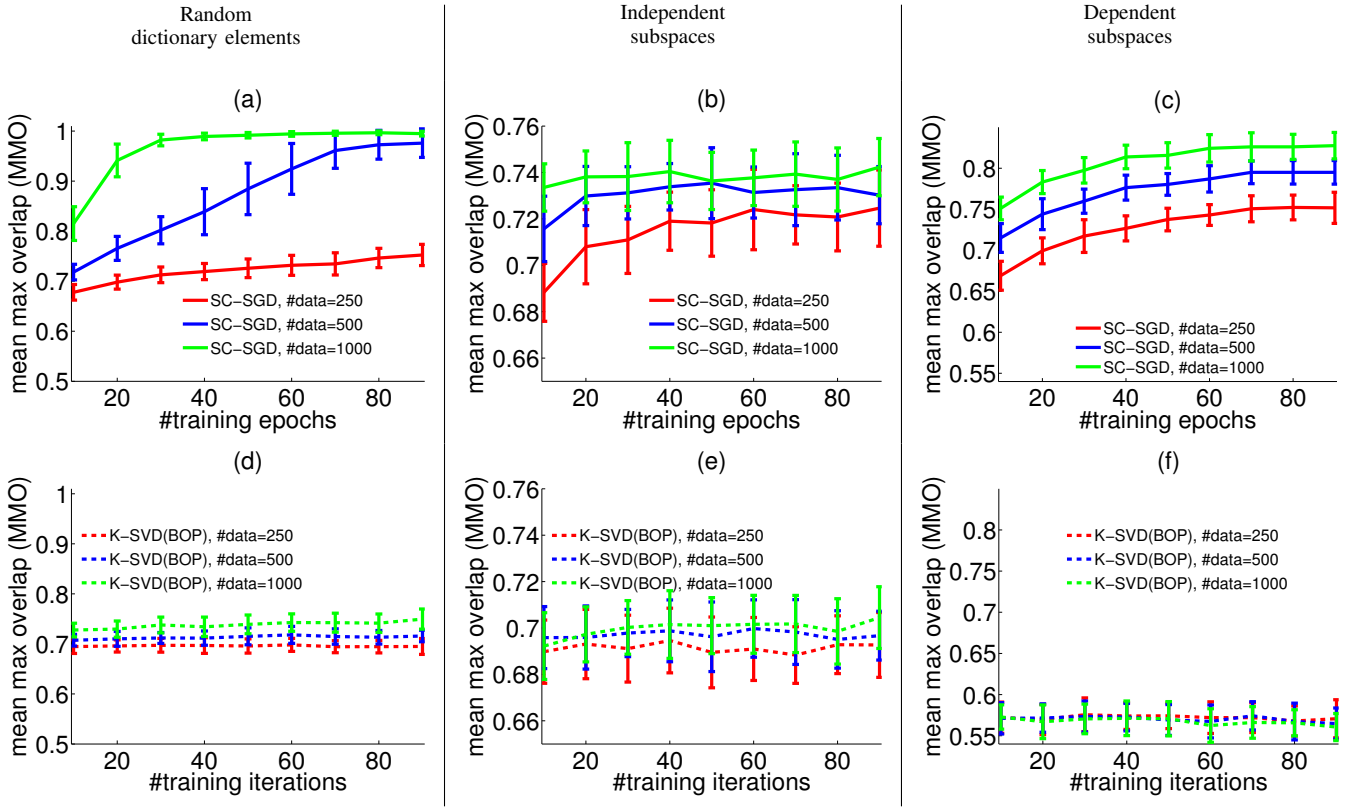

Fig. 6. Experimental results for soft-competitive stochastic gradient descent and K-SVD. The figures show the mean maximum overlap between the learned and the "true" underlying dictionary. k, the number of non-zero entries, was set to 5 in all three scenarios. Both methods employed the BOP algorithm with $K_{\text{user}=50}$ in order to estimate the dictionary coefficients in the learning process. We varied the size of the given training set and the number of training iterations/epochs. It can be seen that soft-competitive stochastic gradient descent is able to determine the "true" dictionary even for small training sets. In all scenarios SC-SGD convergences significantly faster to significantly better results than K-SVD. SC-SGD: soft-competitive stochastic gradient descent ($\alpha_0 = 0.1$, $\alpha_{\text{final}} = 10^{-3}$, $\lambda_0 = 50$, $\lambda_{\text{final}} = 0.1$). All experiments were repeated 50 times.

can be approached in a sparse approximation framework by exploiting sparseness properties of the primal image patches $\mathbf{p}_i$.

The hypothesis that enables us to convert the problem into a sparse approximation problem is that the patches of the primal image can be represented as a sparse linear combination of some dictionary C:

$$\mathbf{p}_i = C\mathbf{a}_i \Rightarrow \mathbf{q}_i = QC\mathbf{a}_i \tag{32}$$

with $C \in \mathbb{R}^{N\times M}$ and $\|\mathbf{a}_i\|_0 \leq k \ll M$. In order to estimate $\mathbf{p}_i$ on the basis of $\mathbf{q}_i$, we consider the following sparse approximation problem

$$\hat{\mathbf{a}}_i = \arg\min_{\mathbf{a}} \|\mathbf{q}_i - QC\mathbf{a}\|_2 \text{ subject to } \|\mathbf{a}\|_0 \leq k \tag{33}$$

where $\hat{\mathbf{p}}_i = C\hat{\mathbf{a}}_i$ is the approximation of $\mathbf{p}_i$. To obtain the approximation $\hat{\mathbf{p}}_i$, one has to solve two problems: (i) choose an appropriate dictionary C and (ii) solve the optimization problem (33).

In the experiments, we consider two images that have been blurred using a Gaussian convolution kernel of size 7×7 with standard deviation 2.33. The first image depicts a mediaeval building of the city of Brugge (see Figure 7). It is also termed house image in the following. The second image depicts a closeup photograph of a flower (see Figure 9). It is termed flower image in the following. The blurred versions of the images are also depicted in Figure 7 and 9.

As already mentioned above, we use dictionaries that have been learned from image data. To obtain suitable dictionaries, we take training sets of images that are similar to the convolved images. The first training set, which is termed town set in the following, consists of photographs of the city of Brugge (see Figure 8). These images mainly depict mediaeval buildings. The second training set, which is termed flowers set in the following, is a collection of closeup photographs of flowers (see Figure 10).

All images are linearly scaled such that the pixel values are in the interval $[0, 1]$. For each image class, from the respective training images we extract a set of training patches consisting of 150.000 patches of size 16×16, i.e., $\mathbf{x}_1, \ldots, \mathbf{x}_{150000}$, $\mathbf{x}_i \in \mathbb{R}^{256}$. The training patches are selected randomly but with a variance within each patch of at least 0.1. Then we apply the hard- and soft-competitive stochastic gradient descent (Sections III and VII) in order to learn overcomplete dictionaries $C \in \mathbb{R}^{256\times 1681}$ that are optimized to encode the respective training set using at most k non-zero entries per sample. In the hard- and soft-competitive case, we use the BOP method with $K_{\text{user}} = 10$ in order to determine the dictionary coefficients during learing. We evaluate different choices of k, i.e., $k = 12, 14, 16, 18, 20$. We perform only one learning epoch, i.e., a one-pass run over the entire training set. For comparison purposes the deconvolution experiments were repeated using an overcomplete Haar-wavelet frame which is

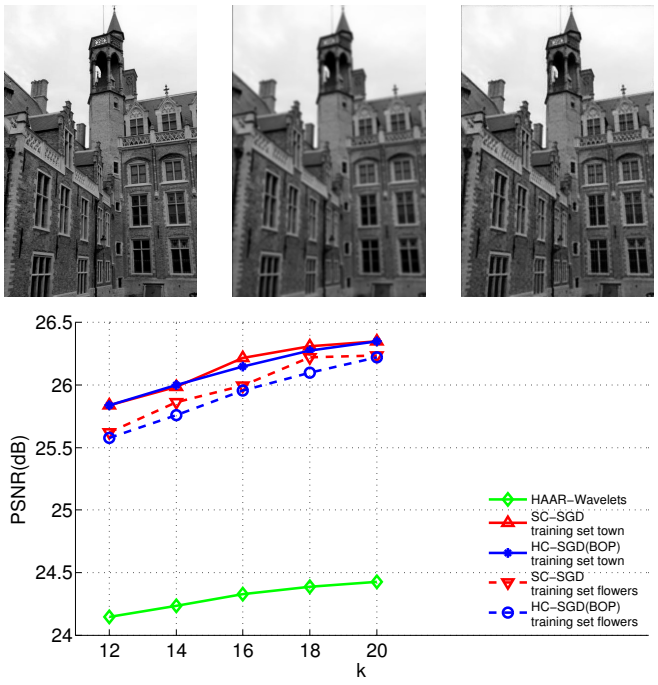

Fig. 7. **Top Left:** primal image, **Top middle:** blurred image (PSNR: 23.3dB), **Top right:** best deconvolution result (PSNR: 26.3dB, SC-SGD, $k = 20$, training set town, dictionary is depicted in figure 8). **Bottom:** PSNR of deconvoluted image with respect to primal image for different choices of k and methods. The larger k is, the better the deconvolution becomes. Best results are obtained if the town set (see figure 8) is used for dictionary training instead of the flowers set. See text for details.

depicted in Figure 11.

In order to solve the optimization problem (33), we first apply the (known) convolution operator Q to the learned dictionary C. Each dictionary element can be interpreted as a 2-dimensional patch. These dictionary patches are blurred using the same operater that was applied to the house and flower image. The learned dictionary elements correspond to 16×16 image patches. Applying a 7×7 gaussian filter to these dictionary elements leads to convolved dictionary elements of size 10×10 since it does not make sense to add a padding region. Hence, the estimation of the coefficients took place by using blurred image patches and dictionary patches of size 10×10. The coefficients are determined using the BOP method ($K_{\text{user}} = 10$). This means that for each tile $\mathbf{p}_i$ of the primal image a set of approximations, i.e., $\hat{\mathbf{p}}_i^j = C\mathbf{a}_i^j \;\; j = 1, \ldots, K_{\text{user}}$, is determined. Instead of just using the best solution according to the reconstruction error, we select from this set the approximation $\hat{\mathbf{p}}_i^{j*}$ that interferes the least with the estimation of the primal image in the neighbourhood of position i.

The interference of an approximation $\hat{\mathbf{p}}_i^j$ with respect to its neighbourhood is computed by considering the set of approximations of those tiles of the primal image that overlap position i. The tiles of the primal image that overlap with position i are denoted as $\mathbf{p}_k \in N(i)$ in the following. Furthermore, let $\hat{\mathbf{p}}_k^l,\; l = 1, \ldots, K_{\text{user}},\; \mathbf{p}_k \in N(i)$ denote the set of approximations of these tiles that have been obtained from the BOP method. Additionally, let $O(i,k)$ and $O(k,i)$ be matrices that implement two mappings into a lower dimensional subspace such that the vector representations of the two tiles at position i and k only contain the corresponding pixels of the tiles that are located in their overlapping region after the mapping has been applied. Now, $I(i,j)$, i.e., the interference of the j-th approximation at position i, is defined as the sum of the distances of the closest approximations of all overlapping tiles in the neighbourhood of position i:

$$I(i,j) = \sum_{\mathbf{p}_k \in N(i)} \min_{l=1,\ldots,K_{\text{user}}} \|O(i,k)\hat{\mathbf{p}}_i^j - O(k,i)\hat{\mathbf{p}}_k^l\|_2 \; . \quad (34)$$

For each tile, we sweep through the set of its approximations and select the one that leads to the smallest interference:

$$j^* = \arg\min_j I(i,j) \; . \quad (35)$$

The final estimation of the pixel value of the primal image at position i is obtained as the mean of the approximations of all tiles that overlap with position i. For each tile from the set of

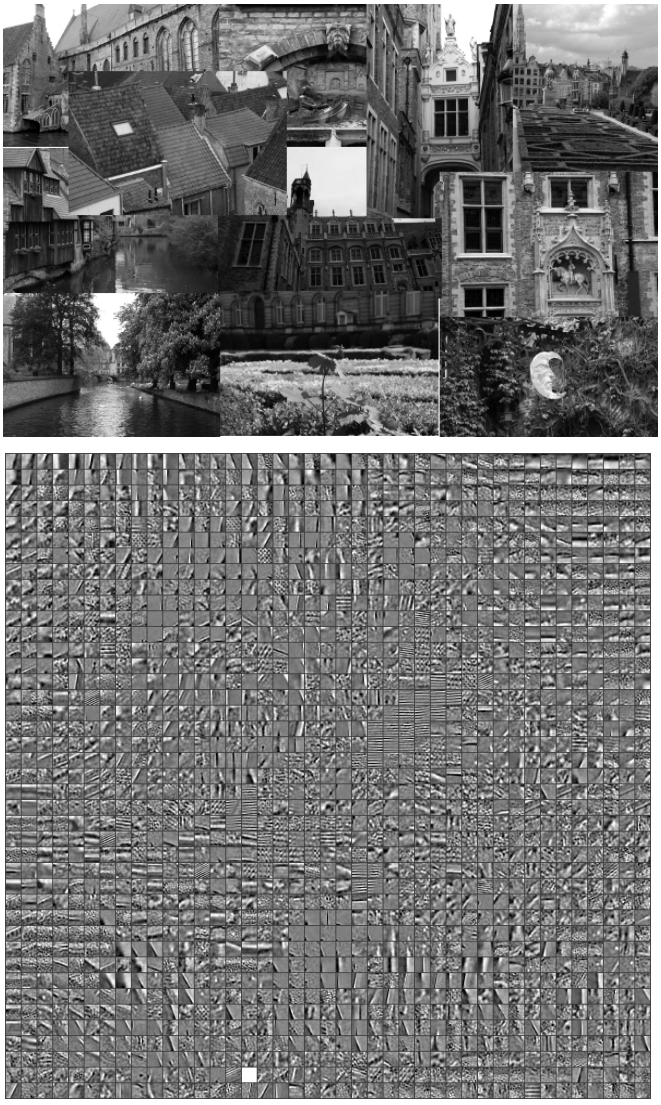

Fig. 8. **Top:** Town training set. **Bottom:** Best performing dictionary obtained from this set of images (SC-SGD, $K_{\text{user}} = 10, \alpha_0 = 10^{-1}, \alpha_{\text{final}} = 10^{-3}, \lambda_0 = 50, \lambda_{\text{final}} = 0.1,\; k = 20$). The dictionary consists of 1681 elements of size 16×16. The 2D-arrangment was obtained from a Kohonen-Map.

approximations that has been obtained from the BOP method the least interfering one is chosen accoding to (35).

A. Deconvolution results

The deconvolution results for the house and flower images are depicted in Figure 7 and 9 respectively. In order to compute a PSNR with respect to the original image all results were scaled to the interval $[0, 1]$ and centered such that the PSNR was maximized. Furthermore, the images were cut such that artifacts at the borders did not influence the PSNR measurement.

In both cases, i.e. house and flower, the PSNR of the sparse reconstruction improves compared to the PSNR of the blurred image. In case of the house image, it can be improved from $23.3dB$ to $26.3dB$ whereas in case of the flower image it can be improved from $25.6dB$ to $33.3dB$. Figure 7 depicts the PSNR values for the house image that are obtained with $k = 12, 14, 16, 18, 20$ and dictionaries learned by hard-competitive as well as soft-competitive stochastic gradient descent. Figure 7 also depicts results that are obtained if a dictionary that has been learned on the flowers training set is used in order to reconstruct the house image. In Figure 9 the same kind of results for the flower image are depicted. Additionally, both figures show the results that have been obtained by using an overcomplete Haar-wavelet frame.

It can be seen that for both images the learned dictionaries outperform the Haar-wavelet frame. The larger the number of non-zero entries k becomes, the better the obtained performance is. Both figures show that the obtained performance depends on the set of training images that have been used for learning the dictionaries. In both cases the dictionaries that have been obtained from the set of images that are similar to the image that is subject to deconvolution outperform those dictionaries that have been learned from images that do not look similar to the convolved image. Due to computational reasons the number of pursuits performed in the BOP method was small ($K_{\text{user}} = 10$) given the size of the dictionary and the number of non-zero coefficients. Hence, the hard-competitive and soft-competitive method are quite similar. Therefore, also the performance of hard-competitive and soft-competitive learning is rather similar though the soft-competitive version more often slightly outperforms the hard-comeptitive approach than the other way around. The PSNRs obtained for the house image range from $25.5dB$ to $26.5dB$, whereas the PSNRs obtained in case of the flower image range from $31dB$ to $33dB$. Therefore, the same absolute variation in the reconstruction quality leads to a higher variance in the PNSR curves in case of the flower image.

The best performing dictionaries that have been learned from the town- and the flowers training set are depicted in Figure 8 and 10, respectively. It can be seen that there is a clear difference between the two dictionaries, e.g., the dictionary obtained from the town set contains more higher frequency components.

X. Conclusions

We have proposed a novel algorithm for dictionary learning that uses stochastic-gradient-descent on a cost function for sparse coding. Results obtained with typical synthetic benchmark data show that this simple and fast method is competitive, and in many cases even superior to computationally more intensive state-of-the-art methods such as MOD or K-SVD, both in terms of how well the representation error is minimized and how well the dictionary is reconstructed.

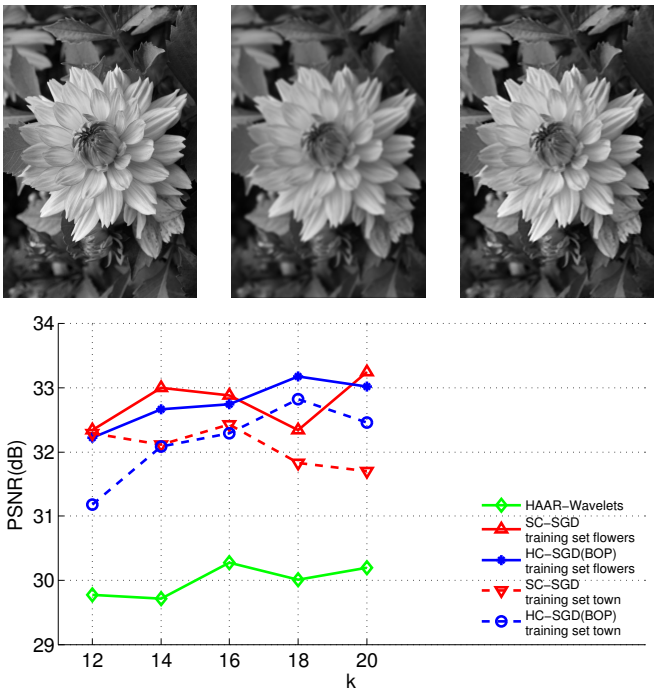

Fig. 9. **Top left:** primal image, **Top middle:** blurred image (PSNR: 25.6dB), **Top right:** best deconvolution result (PSNR: 33.3dB, SC-SGD, $k = 20$,$\alpha_0 = 10^{-1}$, $\alpha_{\text{final}} = 10^{-2}$, $\lambda_0 = 10$, $\lambda_{\text{final}} = 10^{-2}$, training set flowers, dictionary depicted in Figure 10). **Bottom:** PSNR of deconvoluted image with respect to primal image for different choices of k and methods. Best results are obtained if the flowers set (see Figure 10) is used for dictionary training instead of the town set. See text for details.

Moreover, we introduced the Bag of Pursuits (BOP) method in order to obtain an even better estimation of the best k-term approximation of the given data. The BOP can be computationally expensive, but we have shown that both the approximation of the data and the learned dictionaries are significantly improved with BOP. Moreover, the BOP method and a generalization of the Neural-Gas approach have been used to derive the soft-competitive stochastic gradient descent algorithm as a novel method for learning complete and overcomplete dictionaries for sparse coding. This method has further improved our results. One important observation is that the performance of the methods proposed here degrades much slower than that of MOD and K-SVD as a function of the degree of sparseness, which makes the proposed methods more robust and more versatile. This increased applicability is further facilitated by the fact that our methods learn fast and online.

Finally, we have applied the proposed methods to the practical problem of image deconvolution. We have done this for two different classes of images, namely buildings and flowers. The dictionaries have been learned separately for each class. We have then used both dictionaries to invert image

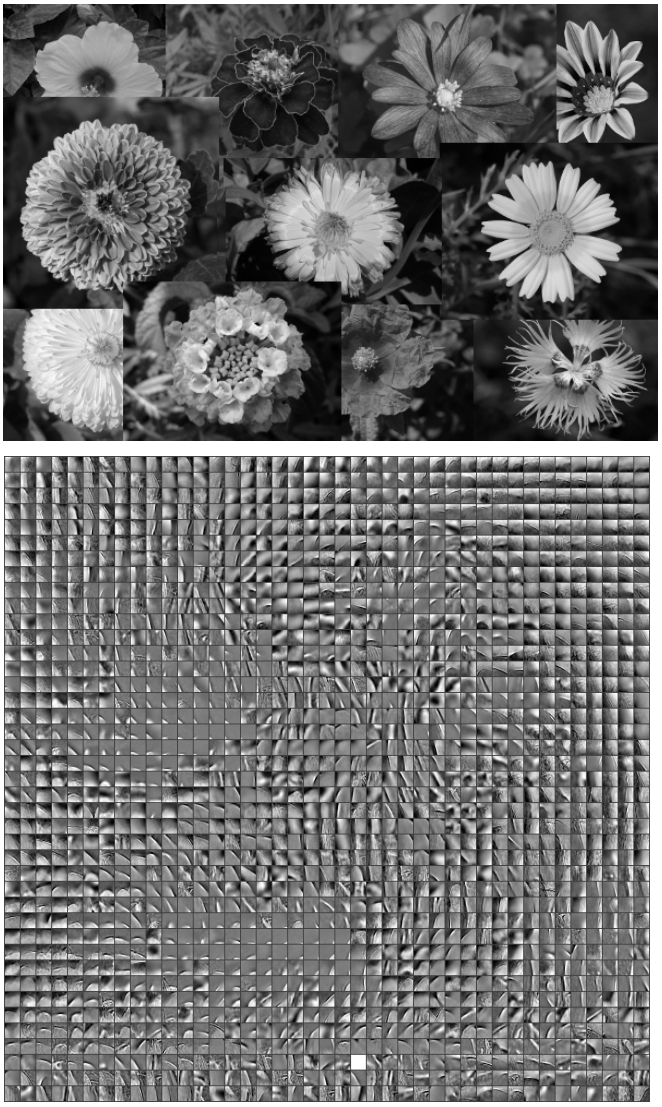

Fig. 10. **Top:** Flowers training set. **Bottom:** Best performing dictionary obtained from this set of images (SC-SGD, $K_{\mathrm{user}} = 10, \alpha_0 = 10^{-1}, \alpha_{\mathrm{final}} = 10^{-2}, \lambda_0 = 10, \lambda_{\mathrm{final}} = 10^{-2}$, $k = 20$). The dictionary consists of 1681 elements of size 16×16. The 2D-arrangment was obtained using a Kohonen-Map.

blur. When using the buildings dictionary for deblurring the buildings, results are significantly better than when using the flowers dictionary. Conversely, the flowers dictionary yielded better results on the flowers images. We have therefore shown that our method is able to learn dictionaries that adapt to a particular class of images. Moreover, all results obtained with our learned dictionaries are clearly better than those obtained with the Haar-wavelets that we use as a reference.

Overall, we have proposed a number of improvements that make the learning of sparse dictionaries faster and more robust. The quality of the resulting dictionaries has also been improved. This has been demonstrated on a comprehensive set of synthetic data and by applying the methods to the real problem of image deconvolution.

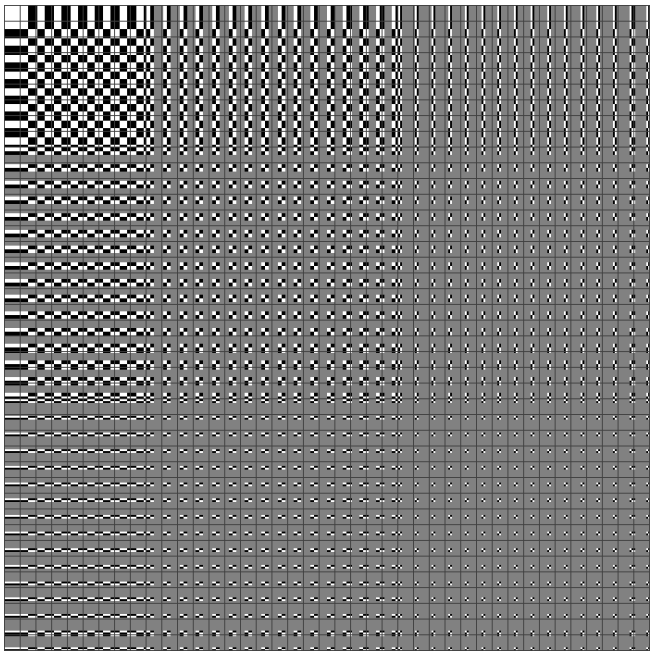

Fig. 11. The Haar-wavelet frame that has been used for comparison purposes.

ACKNOWLEDGMENT

The authors would like to thank Christoph Feilke, who helped to perform the deconvolution experiments. Furthermore, the authors would like to thank the reviewers for their comments which helped to improve the paper.

REFERENCES

[1] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, no. 381, pp. 607–609, 1996.

[2] K. Labusch, E. Barth, and T. Martinetz, "Simple Method for High-Performance Digit Recognition Based on Sparse Coding," *IEEE Transactions on Neural Networks*, vol. 19, no. 11, pp. 1985–1989, 2008.

[3] D. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2845–2862, 1999.

[4] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001. [Online]. Available: http://dx.doi.org/10.1137/S003614450037906X

[5] M. Elad and A. Bruckstein, "A generalized uncertainty principle and sparse representation in pairs of bases," *IEEETrans. Inform. Theory*, vol. 48, pp. 2558–2567, 2002.

[6] D. Donoho and M. Elad, "Optimally sparse representation in general (non-orthogonal) dictionaries via l1 minimization," *Proc. Natl. Acad. Sci.*, vol. 100, pp. 2197–2202, 2003.

[7] R. Gribonval and M. Nielsen, "Sparse decompositions in unions of bases," *IEEETrans. Inform. Theory*, vol. 49, pp. 3320–3325, 2003.

[8] J. A. Tropp, "Greed is good: algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[9] E. Candes and T. Tao, "Decoding by linear programming," *IEEETrans. Inform. Theory*, vol. 51, pp. 4203–4215, 2005.

[10] D. L. Donoho, M. Elad, and V. N. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise." *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 6–18, 2006.

[11] E. Candes, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Comm. Pure Appl. Math.*, vol. 59, pp. 1207–1223, 2006.

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

[13] K. Engan, S. O. Aase, and J. Hakon Husoy, "Method of optimal directions for frame design," in *ICASSP '99: Proceedings of the Acoustics, Speech, and Signal Processing, 1999. on 1999 IEEE International Conference*. Washington, DC, USA: IEEE Computer Society, 1999, pp. 2443–2446.

[14] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, vol. 54, no. 11, pp. 4311–4322, 2006.

[15] K. Labusch, E. Barth, and T. Martinetz, "Sparse Coding Neural Gas: Learning of Overcomplete Data Representations," *Neurocomputing*, vol. 72, no. 7-9, pp. 1547–1555, 2009.

[16] ——, "Bag of Pursuits and Neural Gas for Improved Sparse Coding," in *Proceedings of the 19th International Conference on Computational Statistics*, G. Saporta, Ed. Springer, 2010, pp. 327–336.

[17] G. Davis, S. Mallat, and M. Avellaneda, "Greedy adaptive approximation," *J. Constr. Approx.*, vol. 13, pp. 57–89, 1997.

[18] S.Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3397–3415, 1993.

[19] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition," *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems,*, November 1993.

[20] L. Rebollo-Neira and D. Lowe, "Optimized orthogonal matching pursuit approach," *IEEE Signal Processing Letters*, vol. 9, no. 4, pp. 137–140, 2002.

[21] T. Martinetz and K. Schulten, "A "Neural-Gas Network" Learns Topologies," *Artificial Neural Networks*, vol. I, pp. 397–402, 1991.

[22] T. Martinetz, S. Berkovich, and K. Schulten, ""Neural-gas" Network for Vector Quantization and its Application to Time-Series Prediction," *IEEE-Transactions on Neural Networks*, vol. 4, no. 4, pp. 558–569, 1993.

Kai Labusch studied computer science at the University of Lübeck, where he graduated 2004. He now works as research assistant at the Institute for Neuro- and Bioinformatics of the University of Lübeck, where he pursues a PhD degree.

Erhardt Barth is a professor at the Institute for Neuro- and Bioinformatics at the University of Lübeck, where he leads the research on human and machine vision. He received his Ph.D. in electrical and communications engineering from the Technical University of Munich and has conducted research at the Universities of Melbourne and Munich, the Institute for Advanced Study in Berlin, and the NASA Vision Science and Technology Group in California. In 2000 Dr. Barth received the Schloessmann Award from the Max-Planck Society.

Thomas Martinetz (M'91-SM'04) studied Physics and Mathematics in Munich and Cologne and got his PhD with the Theoretical Biophysics Group at the Beckman Institute of the University of Illinois at Urbana-Champaign. From 1991 to 1996 he led the project Neural Networks for automation control at the Corporate Research Laboratories of the Siemens AG in Munich. From 1996 to 1999 he was Professor for Neural Computation at the Ruhr-University of Bochum and head of the Center for Neuroinformatics. Thomas Martinetz is Chairman of the German Chapter of the European Neural Network Society.