# A DIVIDE-AND-CONQUER DICTIONARY LEARNING ALGORITHM AND ITS PERFORMANCE ANALYSIS

*Subhadip Mukherjee and Chandra Sekhar Seelamantula*

Department of Electrical Engineering, Indian Institute of Science, Bangalore 560012, India
Emails: subhadip@ee.iisc.ernet.in, chandra.sekhar@ieee.org

**ABSTRACT**

We address the problem of learning a sparsifying synthesis dictionary over large datasets that occur in numerous signal and image processing applications, such as inpainting, super-resolution, etc. We develop a dictionary learning algorithm that exploits the similarity of the training examples to reduce the training time. Training datasets containing correlated examples typically occur in image processing applications, as the datasets contain the patches extracted from natural images as training vectors. Our algorithm employs a divide-and-conquer approach, where one leverages the correlation within the training examples to segment the dataset into clusters containing similar examples, and learn local dictionaries for each of them. This constitutes the *divide* step of the algorithm. In the *conquer* step, a global dictionary is trained using the atoms of the local dictionaries as the training examples. We analyze the run-time complexity and the representation error of the proposed divide-and-conquer dictionary learning algorithm, and compare the performance with the batch and online dictionary learning algorithms, both on synthesized dataset and natural images. The analysis reveals that the proposed algorithm has an asymptotic complexity that is linear and logarithmic in the number of training examples, corresponding to sequential and parallel implementations, respectively.

***Index Terms*—** Dictionary learning, sparsity, big-data, divide-and-conquer.

## 1. INTRODUCTION

The problem of learning a sparsifying data-adaptive synthesis dictionary has gained attention in recent years [1–6]. Other than the synthesis model, learning analysis dictionaries [7, 8] and transforms [9–11] for promoting sparsity has also been studied. Algorithms based on dictionary learning find applications in areas such as image super-resolution [12], blind source separation [13], document detection [14, 15], face recognition [16], image classification [17] and restoration [18], etc. The basic idea behind the approach is to learn a dictionary on a set of signals that are likely to occur in a particular application. Dictionaries for data representation are expected to meet two desirable, yet competing criteria: namely, adaptability and fast implementation. Data-independent dictionaries, such as discrete cosine transform, discrete Fourier transform, wavelet etc., offer efficient implementation due to the dyadic structure, but since they are not optimized for a given data, they are limited in their ability to sparsify. On the contrary, learned dictionaries are data-adaptive, but, unlike their data-independent counterparts, a structure has to be imposed to facilitate fast implementation. To bridge the two seemingly contradictory aspects, Rubinstein et al. [19] proposed the idea of a *sparse dictionary*, where the dictionary atoms are themselves sparse over another dictionary. Therefore, the dictionary $D$ can be written as $D = \Phi A$, where the columns of $A$ are $s_1$-sparse. The training examples can be written as $Y = DX = \Phi AX$, where $X$ has $s_2$-sparse columns. This model is also referred to as the *double sparsity* model [20]. It can be observed that, under this model, $Y$ admits a $s$-sparse representation in $\Phi$, where $s \leq s_1 s_2$.

In many image processing applications, the dataset over which the dictionary is trained is large, thus rendering batch processing impracticable due to time and storage constraints. One approach to tackle the problem is to use a sequential update of the dictionary, where only one training example is revealed to the algorithm at a time. Mairal et al. developed a stochastic-approximation-based online dictionary learning algorithm [21, 22], which is scalable to large datasets. However, it is not possible to exploit the hidden structure in the dataset using a sequential algorithm, as it processes the examples in the same order in which they are presented. A scalable online algorithm for dictionary learning was developed by Cherian et al. [23] in the context of sparse noise removal. Mackey et al. [24] proposed a divide-and-conquer approach for noisy matrix factorization. Recently, the problem of distributed dictionary learning over sensor networks has been addressed in [25, 26]. The idea of codeword clustering and hierarchical sparse coding to expedite dictionary learning was proposed in [27]. This technique involves clustering of dictionary atoms, but, unlike ours, does not exploit the correlation of the examples in the dataset to achieve fast learning.

*Our Contribution*: Often the training examples tend to be clustered, which can be modeled by imposing a Gaussian mixture model (GMM) distribution on them, that is, by assuming that the examples are i.i.d., and follow the distribution $p_Y(y) = \sum_{t=1}^{L} \frac{\alpha_t e^{-\frac{1}{2}(y-\mu_t)^T \Sigma_t^{-1}(y-\mu_t)}}{\sqrt{2\pi}\,|\Sigma_t|^{\frac{1}{2}}}$, where $\sum_{t=1}^{L} \alpha_t = 1$. In such scenarios, we propose to learn a global dictionary $\Phi$ in two stages: (i) learn local dictionaries, each for one cluster and (ii) combine them into a global one. Assume that the local dictionary for the $t^{\text{th}}$ cluster $\mathcal{C}_t$ is $D_t$, whose columns admit a sparse representation in a global dictionary $\Phi$. Consequently, it follows that $Y_t = D_t X_t = \Phi A_t X_t = \Phi B_t$, where $B_t = A_t X_t$, and the columns of $Y_t$ are in $\mathcal{C}_t$. Since the examples in a particular cluster have a high degree of correlation, it is possible to represent them using a dictionary sparsely. Then, we leverage the assumption that the atoms of the local dictionaries are sparse over the global dictionary $\Phi$, to learn $\Phi$ using the columns of $D_t$s. The resulting divide-and conquer dictionary learning (DCDL) algorithm to learn a sparsifying global synthesis dictionary $\Phi$ is presented in Algorithm 1, where

one employs only one level of decomposition of the dataset. The complexity and error bound for DCDL are also analyzed.

## 2. PROBLEM STATEMENT AND THE ALGORITHM

Let us assume that the training dataset $T$ contains $n$ examples given by $T = \{y_i\}_{i=1}^{n} \in \mathbb{R}^m$. Our objective is to design a scalable algorithm to learn a dictionary $D \in \mathbb{R}^{m\times k}$, $m < k$, such that $D$ represents each $y_i$ with an $s$-sparse coefficient vector $x_i$, that is, there exist a small constant $\epsilon > 0$ and an integer $s$ such that $\|y_i - Dx_i\|_2 \leq \epsilon$, with $x_i$ satisfying $\|x_i\|_0 \leq s$ for all $i$. Let $Y \in \mathbb{R}^{m\times n}$ denote the matrix formed by stacking the training vectors $y_i$ on the columns. We assume that the dataset $T$ possesses a special structure, in particular, that the training examples exhibit high correlation and form clusters in the $m$-dimensional Euclidean space $\mathbb{R}^m$.

We propose a divide-and-conquer algorithm to solve the problem. First, we segment the training dataset $T$ into $L$ clusters, each containing similar vectors, and train a local dictionary on each of them. The choice of $L$ affects the reduction in training time. Our analysis in Section 2.3 reveals the optimum choice of $L$ as a function of the data-size $n$. Corresponding to two extreme values of $L$, namely $L = 1$ and $L = n$, the training time is observed to be the same as that of the standard K-SVD-based batch dictionary learning. Clustering of data is performed using the standard $K$-means algorithm, where $\ell_1$ distance metric is used as the measure of similarity, in order to achieve robustness to outliers. Let the dictionary trained on the $t^{\text{th}}$ cluster, $1 \leq t \leq L$, be denoted by $D^{(t)} \in \mathbb{R}^{m\times k_t}$, where the number of atoms in the local dictionaries satisfies $k_t \leq k$. In order to obtain a global dictionary $D$ from the local dictionaries, we form a new dataset by stacking the dictionaries $D^{(t)}$ on the columns, and training a dictionary over the dataset $D$ thus constructed. It turns out that the global dictionary $\Phi$ represents the entire dataset with the desired sparsity level, provided that the sparsity is chosen appropriately for the subproblems. The procedure is summarized in Algorithm 1. The parameters $s_1$ and $s_2$ in Steps 2 and 3 of the algorithm are chosen such that $s = s_1 s_2$, where $s$ is the overall sparsity level.

Note that in Algorithm 1, the given dataset $T$ is first clustered, and the local dictionaries are trained directly on the clusters using K-SVD. However, the process can be repeated recursively, where the local dictionaries are trained using the DCDL algorithm, by further segmenting the clusters. In fact, one can continue applying the DCDL algorithm recursively, until one arrives at a dataset containing only two examples. At that stage, a direct technique could be used for training. However, reasonably good savings in time and storage could be obtained by applying a hierarchical clustering on the dataset up to some finite level. Note that the asymptotic complexity expressions hold for the case where the DCDL algorithm is applied until one ends up with a dataset containing only two examples.

### 2.1. Sequential and Parallel Implementation of DCDL

We consider two possible implementations of the proposed DCDL algorithm, namely, parallel and sequential, depending on how the local dictionaries are learned. Since there is no data-dependency in learning the local dictionaries, they can be trained in parallel, in the case where computational time constraint is tighter than the storage constraint. On the other hand, one might choose to train the local dictionaries sequentially, resulting in more training time and less storage requirement compared with the parallel implementation. We derive the worst-case asymptotic time complexities for either cases in Section 2.3. For sequential implementation, the worst-case run-time of the DCDL algorithm scales linearly with respect to the data-size $n$. On the contrary, parallel implementation of the DCDL technique results in asymptotically logarithmic run-time in $n$.

**Algorithm 1** DCDL algorithm to learn a dictionary $\Phi \in \mathbb{R}^{m\times k}$ from a database $Y \in \mathbb{R}^{m\times n}$, such that $\Phi$ represents each column of the data matrix $Y$ using an $s$-sparse coefficient vector.

1. *Cluster the training dataset*: Decompose $Y$ into $L$ non-overlapping clusters $\{Y_t\}_{t=1}^{L}$ of size $m \times n_t$.
2. *Train a dictionary on each dataset*: Learn a dictionary $D_t \in \mathbb{R}^{m\times k_t}$, $k_t \leq k$, to represent the columns of $Y_t$ as $Y_t = D_t X_t$, with a sparsity level of $s_1 < s$.
3. *Merge the dictionaries*: Construct a new data-matrix $D = [D_1, D_2, \cdots, D_L] \in \mathbb{R}^{m\times\sum_{t=1}^{L} k_t}$. Learn a dictionary $\Phi \in \mathbb{R}^{m\times k}$, which represents the columns of $D$ with sparsity $s_2 = \frac{s}{s_1}$.

### 2.2. Bound on Representation Error

Before moving on to the complexity analysis of DCDL, we quantify the loss in performance in terms of the data representation error as a result of the two step approach. In the following development, we use $P$ to indicate a matrix formed by concatenating submatrices $\{P_t\}_{t=1}^{L}$. The overall squared error $e$ on the dataset $T$ is given by

$$\begin{aligned} e(T) &= \sum_{t=1}^{L} \|Y_t - \Phi B_t\|_F^2 = \sum_{t=1}^{L} \|D_t X_t - \Phi A_t X_t + Y_t - D_t X_t\|_F^2 \\ &\leq 2\sum_{t=1}^{L} \|D_t - \Phi A_t\|_F^2 \|X_t\|_F^2 + 2\sum_{t=1}^{L} \|Y_t - D_t X_t\|_F^2 , \end{aligned}$$

where the inequality follows from the fact that $\|P+Q\|_F^2 \leq 2\left(\|P\|_F^2 + \|Q\|_F^2\right)$, for two matrices $P$ and $Q$, and the sub-multiplicative property of the Frobenius norm. Further simplification yields $e(T) \leq 2M\|D - \Phi A\|_F^2 + 2\sum_{t=1}^{L} \|Y_t - D_t X_t\|_F^2$, where $M = \sup_{1\leq t\leq L} \|X_t\|_F$ is a constant. Now, let us assume that the mean-squared errors (MSEs) during the divide and conquer steps are bounded, that is, $\frac{L}{n}\|Y_t - D_t X_t\|_F^2 \leq \epsilon_1$ for all $t$, and $\frac{1}{L}\|D - \Phi A\|_F^2 \leq \epsilon_2$. This, in turn, implies that $e(T) = \|Y - \Phi B\|_F^2 \leq 2ML\epsilon_2 + 2\epsilon_1 n$. Consequently, if the number of components in the GMM satisfies $L = o(n)$, we have $\lim_{n\to\infty} \frac{1}{n} e(T) = \lim_{n\to\infty} \frac{1}{n}\|Y - \Phi B\|_F^2 = 2\epsilon_1$. Therefore, if the number of segments $L$ grows strictly slower that $n$, the overall MSE in two-stage sparse coding asymptotically remains bounded from above, as long as the MSE over each cluster and the MSE during the conquer step are bounded.

### 2.3. Asymptotic Complexity Analysis

The computational complexity of the DCDL algorithm for the sequential and parallel implementations is derived and compared with that of the batch and online learning schemes. The definitions of the asymptotic notations $\mathcal{O}$, $\Theta$, $\Omega$, and $o$ can be found in [28].

#### *2.3.1. Complexity of batch learning*

*Sparse coding*: For a dataset of size $n$, the sparse coding step using *orthogonal matching pursuit* (OMP) [29] requires $\Theta(smkn)$ computations, where $s$ is the sparsity and $m$ is the signal dimension.

*Dictionary update using K-SVD*: For a dictionary of size $m \times k$, one computes the SVD of an $m \times n$ matrix $k$ times, which requires $\Theta\left(km^2n + kn^3\right)$ operations. Therefore, the total computation time required for each iteration of the K-SVD is given by

$$T = cnk\left(sm + m^2 + n^2\right), \tag{1}$$

for some constant $c > 0$. We assume that $m$, $s$, and $k$ are constants and do not scale with $n$. Moreover, we assume that the iterations are repeated a fixed number of times, say $I_1$. Consequently, we have that the complexity of K-SVD is given by $T_{K-\text{SVD}} = \Theta\left(n^3\right)$. As the complexity scales as $n^3$, it is not suitable for large datasets.

#### 2.3.2. Complexity of the online learning algorithm [21]

After a new training example is revealed, one needs to solve a sparse coding problem, requiring $\Theta\left(smk\right)$ computations. This step is followed by updating the dictionary, where one needs to repeat an iterative process certain number of times, each pass of which involves the multiplication of an $m \times k$ matrix by a $k$-length vector, entailing $\Theta\left(mk\right)$ operations. Thus, for $n$ examples, the overall computation needed is given by $\Theta\left(n\right)$, if one assumes that $m$, $k$, and $s$ are much smaller compared to $n$.

#### 2.3.3. Complexity of sequential and parallel versions of DCDL

Using the expression in (1), the complexities of the clustering, splitting, and the merging steps are given below:

$$\begin{aligned}
T_{\text{cluster}} &= c_0 LJmn, \text{where the iteration count } J \text{ is a constant,} \\
T_{\text{split}} &= \sum_{t=1}^{L}\left[c_2 n_t k_1\left(s_1 m + m^2 + n_t^2\right)\right], \\
&= c_1 c_2 n k_1\left[s_1 m + m^2 + c_1^2\frac{n^2}{L^2}\right], \text{if } n_t = c_1\frac{n}{L}, \text{ and} \\
T_{\text{merge}} &= c_3\left(k_1 L\right)k\left(s_2 m + m^2 + k_1^2 L^2\right).
\end{aligned}$$

Therefore, the overall complexity of the divide-and-conquer algorithm is given by

$$\begin{aligned}
T_{\text{DC}} &= T_{\text{cluster}} + T_{\text{split}} + T_{\text{merge}} \\
&= c_0\left(LJmn\right) + c_1 c_2 n k_1\left[s_1 m + m^2 + c_1^2\frac{n^2}{L^2}\right] \\
&+ c_3\left(k_1 L\right)k\left(s_2 m + m^2 + k_1^2 L^2\right).
\end{aligned}$$

Let $L = \Theta\left(n^\alpha\right)$, where $0 < \alpha < 1$. It is easy to observe that the resulting complexity of the algorithm is $T_{\text{DC}} = \Theta\left(n^\tau\right)$, where $\tau = \max\left\{3\alpha, 3 - 2\alpha, 1 + \alpha\right\}$. Clearly, $\tau$ attains a minimum when $\alpha = 0.6$. Again, in principle, each local dictionary and the global one during merging can be trained using a divide-and-conquer algorithm. Therefore, if the local dictionaries are trained sequentially, the overall complexity expression can be written as

$$T(n) = n^{0.6}T\left(n^{0.4}\right) + T\left(n^{0.6}\right), \text{ for } n \geq 2. \tag{2}$$

The solution to the recurrence in (2) is given by $T(n) = \Theta(n)$, as proved in Proposition 1.

**Proposition 1** *For $0 < \alpha < 1$, the solution to the recurrence*

$$T(n) = n^{1-\alpha}T\left(n^\alpha\right) + T\left(n^{1-\alpha}\right), n = 2, 3, \cdots \tag{3}$$

is given by $T(n) = \Theta(n)$.

*Proof*: We prove the proposition by solving the recurrence directly using the substitution method. Let us assume that there exist constants $a, b > 0$ such that $T(n) \leq an - b$, for $n \geq 2$. Substituting in (3), we get $T(n) \leq n^{1-\alpha}\left(an^\alpha - b\right) + \left(an^{1-\alpha} - b\right) \overset{(i)}{\leq} an - b$, where the inequality $(i)$ holds for all $n$ if the constants $a$ and $b$ are chosen such that $b > a$. To handle the base case, we assume that $n \geq 2$, since it makes little sense to train a dictionary where the dataset contains only one example. Consequently, we have to choose two constants $a$ and $b$ such that they satisfy $0 < a < b$ and $T(2) \leq 2a - b$, for the induction hypothesis to work for all $n \geq 2$. To that end, we set $b = 1.5a$, and choose $a$ satisfying $a \geq 2T(2)$. Hence, $T(n) = \mathcal{O}(n)$ solves (2), indicating that the asymptotic complexity can be brought down to a level where it scales at most linearly with the data-size. From (3), it is also easy to verify that $T(n) = \Omega(n)$, because, assuming $T(n) \geq cn$ for $c > 0$ yields

$$T(n) \geq n^{1-\alpha}\left(cn^\alpha\right) + cn^{1-\alpha} \geq cn, \text{ for all } n.$$

Combining $T(n) = \mathcal{O}(n)$ and $T(n) = \Omega(n)$, we get $T(n) = \Theta(n)$. Thus, the worst-case asymptotic run-time complexity of the DCDL algorithm is linear in datasize $n$.

Alternatively, if one chooses to train the local dictionaries in parallel, the asymptotic complexity is given by the recurrence

$$T(n) = T\left(n^{0.4}\right) + T\left(n^{0.6}\right), n \geq 2, \tag{4}$$

having a solution $T(n) = \Theta(\log n)$, as shown in Proposition 2.

**Proposition 2** *For $0 < \alpha < 1$, the solution to the recurrence*

$$T(n) = T\left(n^\alpha\right) + T\left(n^{1-\alpha}\right), n = 2, 3, \cdots \tag{5}$$

is given by $T(n) = \Theta(\log n)$.

*Proof*: To establish the upper-bound on $T(n)$, it suffices to show that there exists a constant $a$ such that $T(n) \leq a \log n$. As in Proposition 1, the proof goes by substitution. Assuming $T(n) \leq a \log n$, we get

$$T(n) \leq a\alpha \log n + a(1 - \alpha)\log n = a \log n.$$

Similarly, one can argue that there exists a constant $b$ such that $T(n) \geq b \log n$, thus proving that $T(n) = \Theta(\log n)$.

However, if one chooses to segment the dataset only once, and learn the local dictionaries directly using K-SVD without further decomposition of the clusters, the complexity of both sequential and parallel implementations is given by $T(n) = \mathcal{O}\left(n^{1.8}\right)$. The savings in computation over the batch K-SVD increase as one employs deeper decomposition of the dataset, albeit at the cost of sacrificing the performance in terms of representation accuracy. As the number of levels in the decomposition increases, the run-time of the sequential DCDL algorithm approaches that of the online technique.

| Performance metrics | Batch K-SVD | Online dictionary learning | DCDL Sequential | DCDL Parallel |
|---|---|---|---|---|
| MSE (dB) | $-16.57$ | $-14.71$ | $-15.20$ | $-15.20$ |
| Time ($\times 10^4$s) | 1.79 | 0.095 | 1.35 | 0.203 |

**Table 1**. Performance comparison of the DCDL algorithm with batch K-SVD and the online dictionary learning technique on synthesized signals drawn from a GMM distribution.

| Images | Batch K-SVD | Online dictionary learning | DCDL parallel |
|---|---|---|---|
|  | 41.33 $(4.82 \times 10^3)$ | 40.65 $(1.61 \times 10^3)$ | 41.67 $(2.70 \times 10^2)$ |
|  | 40.78 $(4.37 \times 10^3)$ | 39.43 $(1.63 \times 10^3)$ | 39.61 $(2.36 \times 10^2)$ |
|  | 39.73 $(4.88 \times 10^3)$ | 36.97 $(1.80 \times 10^3)$ | 39.54 $(2.33 \times 10^2)$ |

**Table 2**. Recovery PSNR (in dB) obtained using different algorithms for three images. The time taken for training the dictionaries (in seconds) is indicated within parentheses. The PSNR values obtained using a sparsity level of $s = 9$ in the overcomplete DCT dictionary are given by $37.65$, $38.95$, and $37.60$ dB, respectively.

## 3. EXPERIMENTAL VALIDATION

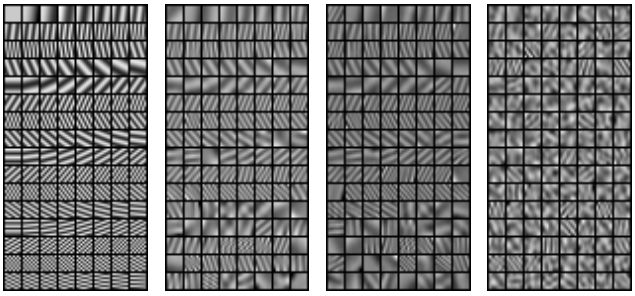
The experiments are performed using MATLAB 2011, running on a system with a $3.2$ GHz core i5 processor and $8$ GB internal memory.

### 3.1. Synthesized Dataset

To validate the performance of the DCDL algorithm and compare the run-time with that of the batch K-SVD and the online dictionary learning (ODL) algorithms [21], we conduct experiments on a synthesized dataset with $n = 10^5$ examples. The training examples are drawn from a GMM distribution, with $L = n^{0.6} = 10^3$, and $\alpha_t = \frac{1}{L}$, for all $t$. The training vectors are of dimension $m = 30$ and the size of the global dictionary is chosen to be $m \times k$, where $k = 80$. The size of the local dictionaries learnt by the DCDL algorithm in the divide stage is taken as $m \times k_t$, where $k_t = k_1 = 50$ for all $t$. The value of sparsity during the divide and conquer stages are set at $s_1 = 4$ and $s_2 = 3$, respectively, so that the overall sparsity of the training examples is given by $s \leq 12$. One level of decomposition is employed in the DCDL algorithm, where the dataset is first segmented into $L$ clusters, and the local dictionaries over the clusters are trained directly using the K-SVD algorithm, without further decomposition. In the merging stage, we accumulate the atoms of the local dictionaries, and train a global dictionary using them, by employing the K-SVD algorithm. The MSE obtained using the algorithms, and the corresponding run-times are reported in Table 1. We observe that the DCDL algorithm outperforms the online learning technique in terms of MSE on the training dataset by approximately $0.5$ dB. However, the DCDL algorithm falls short of the batch K-SVD algorithm by nearly $1.4$ dB. In terms of run-time, the sequential version of the DCDL algorithm is better compared with the competing batch technique, although it uses lesser storage as it involves processing of datasets having smaller size. On the contrary, the parallel implementation of the DCDL algorithm results in a remarkable improvement in run-time, almost by a factor of $10$, in comparison with the batch K-SVD. The training time taken in the parallel implementation is almost $2.14$ times than that required in the online algorithm. Further reduction in training time could be achieved by employing more levels of decomposition, as argued in Section 2.3.

### 3.2. Image Sparsification

Patches of size $8 \times 8$ are extracted from images, with a shift of two pixels in either directions. Subsequently, the patches are vectorized and the mean is subtracted before using them for training. The number of patches thus extracted from an image of size $512 \times 512$ is given by $259^2 \approx 6.4 \times 10^4$. The database of image patches is divided into $L = 50$ segments. The size of the global dictionary to be learnt is $64 \times 128$, whereas the local dictionaries learnt by the DCDL algorithm are of size $64 \times 96$. The sparsity levels during the divide and conquer steps are taken as $s_1 = 3$ and $s_2 = 3$, respectively, to achieve an overall sparse representation of every patch in the global dictionary with sparsity $s = 9$. Following dictionary training, the image is formed by computing the average of overlapping patches reconstructed with a sparsity level of $s = 9$ in the trained dictionaries. The performance of the algorithms is assessed in terms of the recovery peak signal-to-noise ratio (PSNR), defined as $\text{PSNR} = 20 \log_{10} \left( \frac{255\sqrt{P}}{\|I - \hat{I}\|_F} \right)$ dB, where $I$ and $\hat{I}$ denote the actual and the reconstructed images, respectively, and $P$ is the number of pixels in the image.



(a) DCT dictionary (b) Batch K-SVD (c) DCDL (d) Online method

**Fig. 1**. Dictionaries trained on the *Barbara* image.

The recovery PSNR and the training time required by the algorithms for three different images are shown in Table 2. We observe that the reconstruction PSNR of the DCDL algorithm is on par with the batch K-SVD technique, although the training time required is remarkably smaller. To facilitate visual comparison, the dictionary atoms trained on the *Barbara* image using the algorithms are shown in Fig. 1, where the atoms are shown using blocks of size $8 \times 8$. We observe that the DCDL algorithm learns similar atoms as the batch algorithm for representing the image patches, although in significantly less (reduction by a factor of 7, approximately) time. Similarity of the dictionaries trained using the batch K-SVD and the DCDL algorithm is also reflected in the recovery PSNR values.

## 4. CONCLUSIONS

We have developed an algorithm for dictionary learning based on the divide-and-conquer approach, that exploits the similarity of the training examples to achieve reduction in training time. The global dictionary is learned by using the local dictionaries trained on clusters of similar examples. We have established that, for one level of decomposition of the dataset, the MSE obtained using the DCDL algorithm is bounded, as the number of training examples grow to infinity. The asymptotic run-time of the sequential and the parallel implementations of the DCDL algorithm is shown to be linear and logarithmic, respectively, in the size of the training set. Experimentally, we have demonstrated that the proposed DCDL algorithm requires significantly lesser training time in comparison with the batch learning technique, without incurring significant loss (approximately $0.5 - 1$ dB) in performance measured in terms of MSE.

## 5. REFERENCES


[1] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, Jun. 2010.

[2] I. Tosić and P. Frossard, "Dictionary learning," *IEEE Signal Process. Mag.*, vol. 28, no. 2, pp. 27–38, Mar. 2011.

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

[4] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[5] S. Arora, R. Ge, and A. Moitra, "New algorithms for learning incoherent and overcomplete dictionaries," *arXiv:1308.6273v5*, May 2014.

[6] W. Dai, T. Xu, and W. Wang, "Simultaneous codeword optimization (SimCO) for dictionary update and learning," *IEEE Trans. on Signal Process.*, vol. 60, no. 12, pp. 6340–6353, Dec. 2012.

[7] R. Rubinstein, T. Peleg, and M. Elad, "Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model," *IEEE Trans. Signal Process.*, vol. 61, no. 3, pp. 661–677, Feb. 2013.

[8] J. Dong, W. Wang, and W. Dai, "Analysis SimCO: A new algorithm for analysis dictionary learning," in *Proc. IEEE Intl. Conf. on Accoust., Speech, and Signal Process.*, pp. 7193–7197, May 2014.

[9] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Trans. on Signal Process.*, vol. 61, no. 5, pp. 1072–1086, Mar. 2013.

[10] S. Ravishankar, B. Wen, and Y. Bresler, "Online sparsifying transform learning–Part I: Algorithms," *IEEE J. Selected Topics in Signal Process.*, vol. 9, no. 4, pp. 625–636, Jun. 2015.

[11] E. M. Eksioglu and O. Bayir, "K-SVD meets transform learning: Transform K-SVD," *IEEE Signal Process. Lett.*, vol. 21, no. 3, pp. 347–351, Mar. 2014.

[12] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. on Image Process.*, vol. 19, no. 11, pp. 2861–2873, Nov. 2010.

[13] V. Abolghasemi, S. Ferdowsi, and S. Sanei, "Blind separation of image sources via adaptive dictionary learning," *IEEE Trans. on Image Process.*, vol. 21, no. 6, pp. 2921–2930, Jun. 2012.

[14] S. P. Kasiviswanathan, H. Wang, A. Banerjee, and P. Melville, "Online $\ell_1$-dictionary learning with application to novel document detection," in *Proc. Advances in Neural Info. Proc. Systems*, vol. 3, pp. 2258–2266, 2012.

[15] S. P. Kasiviswanathan, "Fast online $\ell_1$ dictionary learning algorithm for novel document detection," in *Proc. IEEE Intl. Conf. on Accoust., Speech, and Signal Process.*, pp. 8585–8589, 2013.

[16] Q. Zhang and B. Li, "Discriminative K-SVD for dictionary learning in face recognition," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (CVPR), pp. 2691–2698, 2010.

[17] L. Li, S. Li, and Y. Fu, "Learning low-rank and discriminative dictionary for image classification," *Elsevier Signal Process.*, vol. 32, no. 10, pp. 814–823, Oct. 2014.

[18] C. Bao, J. Cai, and H. Ji, "Fast sparsity-based orthogonal dictionary learning for image restoration," in *Proc. IEEE Intl. Conf. on Computer Vision*, pp. 3384–3391, 2013.

[19] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1553–1564, Mar. 2010.

[20] N. Ai, J. Peng, X. Zhu, and X. Feng, "SISR via trained double sparsity dictionaries," *Multimedia Tools and Applications*, vol. 74, issue 6, pp. 1997–2007, Mar. 2015.

[21] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proc.* $26^{th}$ *Annual Intl. Conf. on Machine Learning*, pp. 689–696, 2009.

[22] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[23] A. Cherian, S. Sra, and N. Papanikolopoulos, "Denoising sparse noise via online dictionary learning," *in Proc. IEEE Intl. Conf. on Accoust., Speech, and Signal Process.*, pp. 2060–2063, May 2011.

[24] L. Mackey, A. Talwalkar, and M. I. Jordan, "Distributed matrix completion and robust factorization," *Journal of Machine Learning Research*, vol. 16, pp. 913–960, 2015.

[25] P. Chainais and C. Richard, "Distributed dictionary learning over a sensor network," *arXiv:1304.3568v1*, Apr. 2013.

[26] J. Liang, M. Zhang, X. Zeng, and G. Yu, "Distributed dictionary learning for sparse representation is sensor networks," *IEEE Trans. on Image Process.*, vol. 23, no. 6, pp. 2528–2541, Jun. 2014.

[27] T. Xu, W. Dai, and W. Wang, "Fast dictionary learning algorithm via codeword clustering and hierarchical sparse coding," in *Proc.* $9^{th}$ *IMA Intl. Conf. on Math. in Signal Process.*, Dec. 2012.

[28] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* $3^{rd}$ *edition*, The MIT Press, 2005.

[29] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. on Info. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.