Projet de Recherche et d'Innovation Master
(PRIM)

Rapport Final

Convolutional Sparse Coding pour l'apprentissage de représentation sur des signaux et des images

Basé sur la papier scientifique

Fast and Flexible Convolutional Sparse Coding

(Felix Heide, Wolfgang Heidrich, Gordon Wetzstein)

Etudiants
Thai-Chau TRUONG
Quang-Huy DINH

Professeur
Alexandre GRAMFORT

February 14, 2016

Abstract

In this project, we study and implement one of the newest methods in the field of convolutional sparse coding in [6]. To test the performance of the method, we do experiments on grey-scaled images and electroencephalography (EEG) signals. Experimental results show that this method works well on these types of data.

Contents

1 Theoretical Framework

1.1 Review of the convolutional sparse coding (CSC) problem

Before the use of convolution operator to find a sparse coding representation, there existed patch-based approaches that solve the following problem

$$\begin{aligned} \underset{D,z}{\text{argmin}} \quad & \frac{1}{2}\sum_{i=1}^{N} \|x_i - Dz_i\|_2^2 + \beta \|z_i\|_1 \\ \text{subject to} \quad & \|d_k\|_2^2 \leq 1 \ \forall k \in \{1, \dots, K\} \end{aligned}$$

where x_i is the original data, z_i is the sparse feature map of each data instance and $D = [d_1, d_2, ..., d_K]$ is the set of filters (or kernels) on all dataset. The parameter β controls the trade-off between the sparsity of the codes and the reconstruction error. When β is large, the function favors the minimization of the L1-regularized term of the code and therefore the output is sparser. On the contrary, when β is small, the algorithm focuses more on minimizing the difference between the original and the reconstructed signal and we have a more accurate reconstruction but lower sparsity. The constraint on the L2-norm of the filters prevents them from absorbing too much energy of the objective function. Without this constraint, the weights of the filters could be arbitrarily large and this diminishes the corresponding sparse feature maps. This problem is also known as a LASSO problem and has many application such as perceptual classification [11], signal reconstruction [5], on-line learning [9].
However, despite the low computational complexity, this approach has some significant drawback. First, the learned common features are not diverse. The filters do not cover a wide range of available structures in the dataset. Second, this method produces redundant linearly dependent filters, which means that there exist filters that are shifted versions of each other.
To overcome the disadvantages of patch-based approaches, the convolution operator is used. And the sparse coding problem has the new following form

$$\begin{aligned} \underset{d,z}{\text{argmin}} \quad & \frac{1}{2}\left\| x - \sum_{k=1}^{K} d_k * z_k \right\|_2^2 + \beta \sum_{k=1}^{K} \|z_k\|_1 \\ \text{subject to} \quad & \|d_k\|_2^2 \leq 1 \ \forall k \in \{1, \dots, K\} \end{aligned} \tag{1}$$

In this new problem, the element-wise multiplication operator is replaced by the convolution operator. All parameters have the same roles as those of patch-based approaches. To handle the issue of high computational complexity of the convolution, Bristow et al. (in [2] and [4]) exploit the Parseval's theorem to transform the optimization problem (1) into the following problem

$$\begin{aligned} \underset{d,z}{\text{argmin}} \ & \frac{1}{2}\left\| \hat{x} - \sum_{k=1}^{K} \hat{d}_k \odot \hat{z}_k \right\|_2^2 + \beta \sum_{k=1}^{K} \|t_k\|_1 \\ \text{subject to} \ & \|s_k\|_2^2 \leq 1 \forall k \in \{1, \dots, K\} \\ & s_k = S\Phi^T \hat{d}_k \forall k \in \{1, \dots, K\} \\ & t_k = z_k \forall k \in \{1, \dots, K\} \end{aligned} \tag{2}$$

where $\hat{x}$, $\hat{d}_k$, $\hat{z}_k$ are the discrete Fourier transform (DFT) transform of x, d and z respectively. ϕ is the DFT matrix. S is the projection matrix that limits the filters to their corresponding small size in the spatial domain. By this reformulation, the computationally expensive original problem is reformulated as more computationally efficient new problem with element-wise multiplication. To solve this, one popular optimization algorithm is to combine the coordinate descent and Alternating Direction Method of Multipliers (ADMM) algorithm ([1], [3] and [8]) to update the codes and the filters separately but consecutively. The approaches using convolution operator are empirically proven to capture more structures of the data than patch-based ones and also solve the issue of linearly dependent filters (see Figure 1).
The new encountered hindrance for the convolution is that the solution suffers from boundary effect. The point that are closed to the boundary is covered by less filters than those in the middle of an image or a signal. This affects the difference between the original and the reconstructed data, especially in the boundary when the filter size is larger. In this project, we present the work of the authors in [6] who present a new method to find convolutional sparse codings. In this approach, they improve the accuracy

of the solution by using a new formula for the CSC optimization problem and show that this can handle better the obstacle of boundary effect. Details about this approach is presented in Section 1.2.

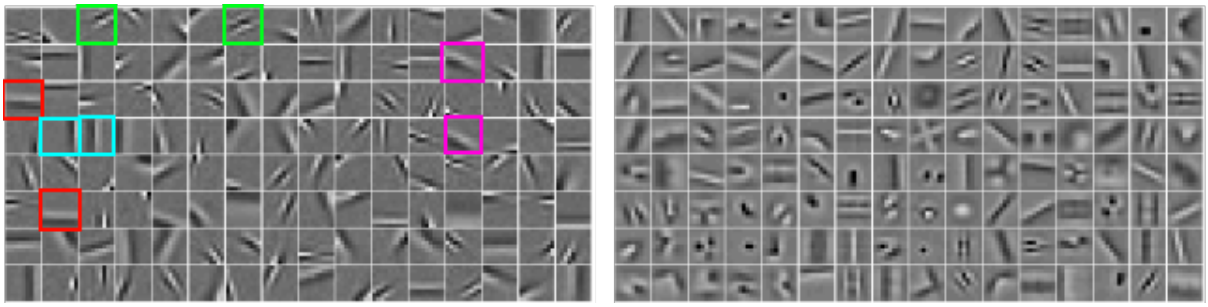

Figure 1: Filters learned by patch-based approach (left) and convolution approach (right). On the left, there are many filters that appear as a translation of some others. And we can see larger variety of features in the right image. (Source:[7].)

1.2 The authors' proposed method

1.2.1 The new optimization problem

The new proposed formula in [6] is as following

$$\begin{aligned} \underset{d,z}{\text{argmin}} \quad & \frac{1}{2}\left\|x - M\sum_{k=1}^{K} d_k * z_k\right\|_2^2 + \beta\sum_{k=1}^{K}\|z_k\|_1 \\ \text{subject to} \quad & \|d_k\|_2^2 \leq 1 \ \forall k \in \{1,\dots,K\} \end{aligned} \tag{3}$$

In this new problem, M is a diagonal or block-diagonal matrix that masks out the padded boundary when solving d_k and z_k. So the algorithm can apply the same convolution operator with the points in the boundary without having to handle the boundary problem. Each point is covered by the same number of filters regardless of their position on the image. This can make the reconstruction become more accurate especially on the boundary regions.

1.2.2 Splitting of the objective

Due to the appearance of M in the convolution between d_k and z_k, this problem cannot be solved by the Fourier method as in (2). To use this, we need the following transformations of (3):

$$\begin{aligned} & \underset{d,z}{\text{argmin}} \ \frac{1}{2}\left\|x - M\sum_{k=1}^{K} d_k * z_k\right\|_2^2 + \beta\sum_{k=1}^{K}\|z_k\|_1 + \sum_{k=1}^{K} ind_C(d_k) \\ = \quad & \underset{d,z}{\text{argmin}} \ f_1(Dz) + \sum_{k=1}^{K}\left(f_2(z_k) + f_3(d_k)\right) \end{aligned} \tag{4}$$

where $D = [D_1, ..., D_K]$ is a concatenation of Toeplitz matrices corresponding to a convolution of each filter d_k to the sparse feature map $z = [z_1^T, ..., z_K^T]$. The splitting functions are

$$f_1(v) = \frac{1}{2}\|x - Mv\|_2^2, f_2(v) = \beta\|v\|_1, f_3(v) = ind_C(v) \text{ with } C = \{v | \ \|Sv\|_2^2 \leq 1\}$$

From this transformation, the application of ADMM to find the codes and filters can be expressed as follow

$$\begin{aligned} \underset{u}{\text{argmin}} \quad & f(u) \\ \text{subject to} \quad & Kw = u \end{aligned} \tag{5}$$

where

- f is the objective function corresponding to an ADMM problem. The authors reuse the idea of coordinate descent with two steps to update separately the codes (the z-step) and the filters (d-step). In each step, the corresponding ADMM objective is different.
- u is the variable that we would like to get result. In z-step, u contains the codes z_k. And in d-step, u contains the filters d_k
- w is the new introduced variable to conform the standard ADMM problem, the use of this variable is presented in details in Section 1.2.3.

1.2.3 ADMM for each step in the coordinate descent

The main schema of this method composes of two main structures: the outer structure is the coordinate descent algorithm that updates the filters (the d-step) and the codes (the z-step) respectively in each iteration. The inner structure of each step is the ADMM algorithm that solves the d-step and the z-step.

The d-step to find the filters
In this step, we find the filters via the following problem

$$\underset{d}{\text{argmin}}\ f_1(Zd) + \sum_{k=1}^{K} f_3(d_k) \tag{6}$$

where Z has the same meaning as D in In this step, the Toeplitz matrices Using ADMM, we join the two functions f_1 and f_3 into $f = f_1 + \sum f_3$ and introduce a new variable w by setting

$$w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} Zd \\ d_1 \\ d_2 \\ \dots \\ d_K \end{pmatrix}$$

where w_1 is the components that corresponds to the Zd part and w_2 is for the rest. The optimization problem (6) becomes

$$\begin{aligned} &\underset{w,d}{\text{argmin}} && f(w) + g(d) \\ &\text{subject to} && w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} Zd \\ d_1 \\ d_2 \\ \dots \\ d_K \end{pmatrix} \end{aligned} \tag{7}$$

where g=0, the function g is added to be compatible with the standard form of ADMM in [1]. The following augmented Lagrangian is formed with two pairs of Lagrange and augmented Lagrange multipliers.

$$L_{\rho_d}(w,d,y) = f(w)+g(d)+[y_1,y_2]^T([Zd,d_1,...,d_K]^T-w)+\frac{\rho_{d,1}}{2}\left\|(Zd)^T-w_1\right\|_2^2+\frac{\rho_{d,2}}{2}\left\|[d_1,...,d_K]^T-w_2\right\|_2^2$$

where $y = [y_1, y_2]^T$ is the Lagrange multiplier whose the first part y_1 is compatible with $(Zd)^T$ and w_1 in the computation, the second part y_2 is for the rest $[d_1, d_2, ..., d_K]^T$ and w_2. The authors rewrite the

problem in scale form by setting $\lambda_i = \frac{y_i}{\rho_{d,i}}$ (i=1,2) and the iterations under scaled form is as following

$$
\begin{aligned}
d^{k+1} &= \underset{d}{\operatorname{argmin}}\ g(d) + \left(\frac{\rho_{d,1}}{2}\left\|Zd - w_1 + \lambda_1^k\right\|_2^2 + \frac{\rho_{d,2}}{2}\left\|[d_1,...,d_K]^T - w_2 + \lambda_2^k\right\|_2^2\right) \\
&= \underset{d}{\operatorname{argmin}}\ \left(\frac{\rho_{d,1}}{2}\left\|Zd - w_1 + \lambda_1^k\right\|_2^2 + \frac{\rho_{d,2}}{2}\left\|[d_1,...,d_K]^T - w_2 + \lambda_2^k\right\|_2^2\right) \\
w^{k+1} &= \underset{w}{\operatorname{argmin}}\ f(w) + \left(\frac{\rho_{d,1}}{2}\left\|Zd^{k+1} - w_1 + \lambda_1^k\right\|_2^2 + \frac{\rho_{d,2}}{2}\left\|[d_1^{k+1},...,d_K^{k+1}]^T - w_2 + \lambda_2^k\right\|_2^2\right) \\
&= \underset{w}{\operatorname{argmin}}\ f_1(w_1) + \sum_i f_3(w_{2,i}) + \left(\frac{\rho_{d,1}}{2}\left\|Zd^{k+1} - w_1 + \lambda_1^k\right\|_2^2 + \frac{\rho_{d,2}}{2}\left\|[d_1^{k+1},...,d_K^{k+1}]^T - w_2 + \lambda_2^k\right\|_2^2\right) \\
&= \underset{w}{\operatorname{argmin}}\ f_1(w_1) + \frac{\rho_{d,1}}{2}\left\|w_1 - \left(Zd^{k+1} + \lambda_1^k\right)\right\|_2^2 + \sum_{i=1}^{K}\left(f_3(w_{2,i}) + \frac{\rho_{d,2}}{2}\left\|w_{2,i} - \left(d_i^{k+1} + \lambda_{2,i}^k\right)\right\|_2^2\right) \\
&= \begin{pmatrix} prox_{\frac{f_1}{\rho_{d,1}}}\left(Zd^{k+1} + \lambda_1^k\right) \\ prox_{\frac{f_3}{\rho_{d,2}}}\left(d_1^{k+1} + \lambda_{2,1}^k\right) \\ \cdots \\ prox_{\frac{f_3}{\rho_{d,2}}}\left(d_K^{k+1} + \lambda_{2,K}^k\right) \end{pmatrix}
\end{aligned}
\tag{8}
$$

To solve the first iteration of d^{k+1}, we first rewrite the function to minimize as

$$
\begin{aligned}
&\underset{d}{\operatorname{argmin}}\ \frac{\rho_{d,1}}{2}\left(\left\|Zd - w_1 + \lambda_1^k\right\|_2^2 + \frac{\rho_{d,2}}{\rho_{d,1}}\sum_{i=1}^{K}\left\|d_i - w_{2,i} + \lambda_{2,i}^k\right\|_2^2\right) \\
= \quad &\underset{d}{\operatorname{argmin}}\ \left(\|Zd - \tau_1\|_2^2 + \frac{\rho_{d,2}}{\rho_{d,1}}\sum_{i=1}^{K}\|d_i - \tau_{2,i}\|_2^2\right)
\end{aligned}
$$

where $\tau_i = w_i - \lambda_i^k$. Without the presence of M, the minimization of this function can be solved efficiently in the Fourier domain. Specifically, the authors reuse the idea in [8] that solves the following optimization problem

$$
\underset{\hat{d}}{\operatorname{argmin}}\ \left(\left\|\hat{Z}\odot\hat{d} - \hat{\tau}_1\right\|_2^2 + \frac{\rho_{d,2}}{\rho_{d,1}}\sum_{i=1}^{K}\left\|\hat{d}_i - \hat{\tau}_{2,i}\right\|_2^2\right)
$$

This expression can be optimized separately and parallelly. Specifically, at each point u, the operation $\hat{Z}\odot\hat{d}$ can be expressed as the multiplication of two matrices: $\hat{Z}_u$ with size $N\times K$ (N is the number of instances) and $\hat{d}$ with size $K\times 1$ (see Figure 2). Therefore, this quadratic function attains the minimum at

$$
\hat{d}^* = \left(\hat{Z}_u^H\hat{Z}_u + \frac{\rho_{d,2}}{\rho_{d,1}}\mathbb{I}\right)^{-1}\left(\hat{Z}_u^H\hat{\tau}_1 + \hat{\tau}_2\right)
$$

The first term with the inverse operator can be calculated and cached before the iterations of ADMM. But when $N << K$, the multiplication of $\hat{Z}_u^H\hat{Z}_u$ leads to a $K\times K$ matrix and the cost to inverse it is large. Using Woodbury formula, the size of the matrix that needs to be inversed can be reduced dramatically to $N\times N$, we have:

$$
\begin{aligned}
\left(\hat{Z}_u^H\hat{Z}_u + \frac{\rho_{d,2}}{\rho_{d,1}}\mathbb{I}\right)^{-1} &= \left(\frac{\rho_{d,2}}{\rho_{d,1}}\mathbb{I}\right)^{-1} - \left(\frac{\rho_{d,2}}{\rho_{d,1}}\mathbb{I}\right)^{-1}\hat{Z}_u^H\left(\mathbb{I} + \hat{Z}_u\left(\frac{\rho_{d,2}}{\rho_{d,1}}\mathbb{I}\right)^{-1}.\hat{Z}_u^H\right)^{-1}\hat{Z}_u\left(\frac{\rho_{d,2}}{\rho_{d,1}}\mathbb{I}\right)^{-1} \\
&= \frac{\rho_{d,1}}{\rho_{d,2}}\mathbb{I} - \frac{\rho_{d,1}^2}{\rho_{d,2}^2}\hat{Z}_u^H\left(\mathbb{I} + \hat{Z}_u\hat{Z}_u^H\right)^{-1}\hat{Z}_u
\end{aligned}
$$

To solve the second iteration of w^{k+1}, the authors use the proximal operators that are presented in [10]. Specifically, the proximal operators for f_1 and f_3 are

$$
\begin{aligned}
\operatorname{prox}_{\theta f_1}(v) &= \left(\mathbb{I} + \theta M^T M\right)^{-1}\left(v + \theta M^T x\right) \\
\operatorname{prox}_{\theta f_3}(v) &= \begin{cases} \frac{Sv}{\|Sv\|_2} & \|Sv\|_2^2 \geq 1 \\ Sv & \text{otherwise} \end{cases}
\end{aligned}
$$

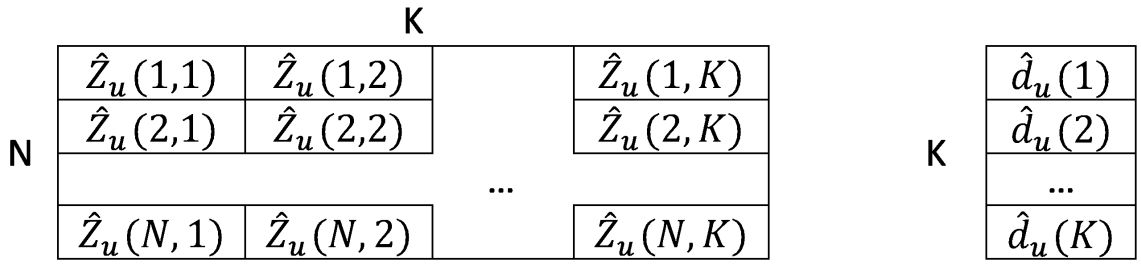

Figure 2: Demonstration for the reordering of matrices $\hat{Z}_u$ (left) and $\hat{d}_u$ (right) for a point u

The z-step to find the codes

In this step, we find the filters via the following problem

$$\underset{d}{\text{argmin}}\ f_1(Dz) + \sum_{k=1}^{K} f_2(z_k) \tag{9}$$

By applying the same idea as d-step, we join the two functions f_1 and f_2 into $f = f_1 + \sum f_2$ and introduce a new variable w by setting

$$w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} Dz \\ z_{1,1} \\ z_{1,2} \\ \dots \\ z_{1,K} \\ \dots \\ z_{N,K} \end{pmatrix}$$

The optimization problem (9) becomes

$$\begin{aligned} &\underset{w,d}{\text{argmin}} && f(w) + g(z) \\ &\text{subject to} && w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} Dz \\ z_{1,1} \\ z_{1,2} \\ \dots \\ z_{N,K} \end{pmatrix} \end{aligned} \tag{10}$$

where g=0. The following augmented Lagrangian is formed with two pairs of Lagrange and augmented Lagrange multipliers.

$$\begin{aligned} L_{\rho_z}(w,z,y) = f(w) + g(z) + [y_1,y_2]^T([Dz, z_{1,1}, ..., z_{N,K}]^T - w) \\ + \frac{\rho_{z,1}}{2}\left\|(Dz)^T - w_1\right\|_2^2 + \frac{\rho_{z,2}}{2}\left\|[z_{1,1}, ..., z_{N,K}]^T - w_2\right\|_2^2 \end{aligned}$$

where $y = [y_1, y_2]^T$ is the Lagrange multiplier whose the first part y_1 is compatible with $(Dz)^T$ and w_1 in the computation, the second part y_2 is for the rest $[z_{1,1}, ..., z_{N,K}]^T$ and w_2. The authors rewrite the

problem in scale form by setting $\lambda_i = \frac{y_i}{\rho_{z,i}}$ (i=1,2) and the iterations under scaled form is as following

$$
\begin{aligned}
z^{k+1} &= \underset{z}{\operatorname{argmin}}\ g(z) + \left(\frac{\rho_{z,1}}{2}\left\|Dz - w_1 + \lambda_1^k\right\|_2^2 + \frac{\rho_{z,2}}{2}\left\|[z_{1,1},...,z_{N,K}]^T - w_2 + \lambda_2^k\right\|_2^2\right) \\
&= \underset{z}{\operatorname{argmin}}\ \left(\frac{\rho_{z,1}}{2}\left\|Dz - w_1 + \lambda_1^k\right\|_2^2 + \frac{\rho_{z,2}}{2}\left\|[z_{1,1},...,z_{N,K}]^T - w_2 + \lambda_2^k\right\|_2^2\right) \\
w^{k+1} &= \underset{w}{\operatorname{argmin}}\ f(w) + \left(\frac{\rho_{z,1}}{2}\left\|Dz - w_1 + \lambda_1^k\right\|_2^2 + \frac{\rho_{z,2}}{2}\left\|[z_{1,1},...,z_{N,K}]^T - w_2 + \lambda_2^k\right\|_2^2\right) \\
&= \underset{w}{\operatorname{argmin}}\ f_1(w_1) + \sum_{i,j} f_2(w_{2,i,j}) \\
&+ \left(\frac{\rho_{z,1}}{2}\left\|Dz^{k+1} - w_1 + \lambda_1^k\right\|_2^2 + \frac{\rho_{z,2}}{2}\left\|[z_{1,1}^{k+1},...,z_{N,K}^{k+1}]^T - w_2 + \lambda_2^k\right\|_2^2\right) \\
&= \underset{w}{\operatorname{argmin}}\ f_1(w_1) + \frac{\rho_{z,1}}{2}\left\|w_1 - \left(Dz^{k+1} + \lambda_1^k\right)\right\|_2^2 \\
&+ \sum_{i=1}^{N}\sum_{j=1}^{K}\left(f_2(w_{2,i,j}) + \frac{\rho_{z,2}}{2}\left\|w_{2,i,j} - \left(z_{i,j}^{k+1} + \lambda_{2,i,j}^k\right)\right\|_2^2\right) \\
&= \begin{pmatrix} prox_{\frac{f_1}{\rho_{z,1}}}\left(Dz^{k+1} + \lambda_1^k\right) \\ prox_{\frac{f_2}{\rho_{z,2}}}\left(z_{1,1}^{k+1} + \lambda_{2,1,1}^k\right) \\ \cdots \\ prox_{\frac{f_2}{\rho_{z,2}}}\left(z_{N,K}^{k+1} + \lambda_{2,N,K}^k\right) \end{pmatrix}
\end{aligned}
\tag{11}
$$

To solve the first iteration of d^{k+1}, we first rewrite the function to minimize as

$$
\begin{aligned}
&\underset{z}{\operatorname{argmin}}\ \frac{\rho_{z,1}}{2}\left(\left\|Dz - w_1 + \lambda_1^k\right\|_2^2 + \frac{\rho_{z,2}}{\rho_{z,1}}\sum_{i=1}^{N}\sum_{j=1}^{K}\left\|z_{i,j} - w_{2,i,j} + \lambda_{2,i,j}^k\right\|_2^2\right) \\
= &\underset{z}{\operatorname{argmin}}\ \left(\|Dz - \tau_1\|_2^2 + \frac{\rho_{z,2}}{\rho_{z,1}}\sum_{i=1}^{N}\sum_{j=1}^{K}\|z_{i,j} - \tau_{2,i,j}\|_2^2\right)
\end{aligned}
$$

where $\tau_i = w_i - \lambda_i^k$. The minimal value is found by Fourier trick as following:

$$
\underset{\hat{z}}{\operatorname{argmin}}\ \left(\left\|\hat{D}\odot\hat{z} - \hat{\tau}_1\right\|_2^2 + \frac{\rho_{z,2}}{\rho_{z,1}}\sum_{i=1}^{N}\sum_{j=1}^{K}\|\hat{z}_{i,j} - \hat{\tau}_{2,i,j}\|_2^2\right)
$$

Similarly to the d-step, at each point u, the operation $\hat{D}\odot\hat{z}$ is expressed as the multiplication of two matrices: $\hat{D}_u$ with size $1\times K$ and $\hat{z}$ with size $K\times N$ (see Figure 3 for demonstration):

$$
\hat{z}^* = \left(\hat{D}_u^H\hat{D}_u + \frac{\rho_{z,2}}{\rho_{z,1}}\mathbb{I}\right)^{-1}\left(\hat{D}_u^H\hat{\tau}_1 + \hat{\tau}_2\right)
$$

Since $\hat{D}_u$ just contains a row vector, by applying Woodbury formula, we can write the inversion as:

$$
\left(\hat{D}_u^H\hat{D}_u + \frac{\rho_{z,2}}{\rho_{z,1}}\mathbb{I}\right)^{-1} = \frac{\rho_{z,1}}{\rho_{z,2}}\mathbb{I} - \frac{\rho_{z,1}^2}{\rho_{z,2}^2}\cdot\frac{\hat{D}_u^H.\hat{D}_u}{\left(1 + \frac{\rho_{z,1}}{\rho_{z,2}}\hat{D}_u\hat{D}_u^H\right)}
$$

The proximal operators to solve the second iteration of w^{k+1} are

$$
\begin{aligned}
\text{prox}_{\theta f_1}(v) &= \left(\mathbb{I} + \theta M^T M\right)^{-1}\left(v + \theta M^T x\right) \\
\text{prox}_{\theta f_2}(v) &= \max\left(1 - \frac{\theta\beta}{|v|}, 0\right)\odot v
\end{aligned}
$$

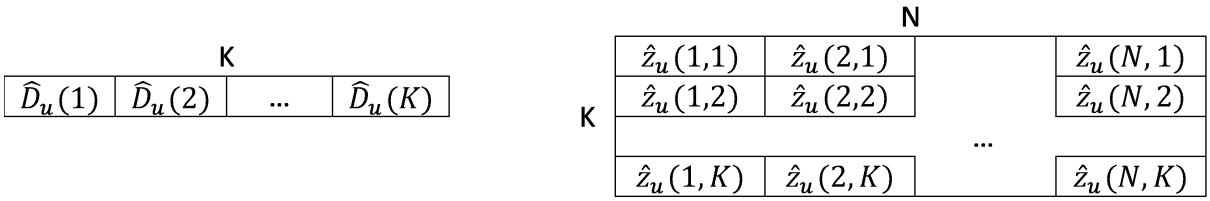

Figure 3: Demonstration for the reordering of matrices $\hat{D}_u$ (left) and $\hat{z}_u$ (right) for a point u

1.2.4 Putting things together

Wrapping up all calculation that is done in the previous sections, we have the new final method to solve the CSC problem in Algorithm 1

Algorithm 1 CSC solving by coordinate descent and ADMM

Input: Initialized variables: $d^0, z^0, \lambda_d^0, \lambda_z^0$
Penalty parameters: ρ_d, ρ_z

1: **repeat**
2: **d-step - filter update:** Using the ADMM method to solve
3: $d^i, \lambda_d^i \leftarrow \text{argmin}_d \left(f_1(Zd) + \sum_i f_3(d_i)\right)$
4: **z-step - code update:** Using the ADMM method to solve
5: $z^i, \lambda_z^i \leftarrow \text{argmin}_z \left(f_1(Dz) + \sum_i f_2(z_i)\right)$
6: **until** Enough iterations or no more change in two directions.

2 Our experiments

In this section, we conduct the experiments of the CSC algorithms on digital images (section 2.1) and EEG signals (section 2.2). In all experiments, the parameters ρ_d and ρ_z are fixed heuristically at $\left(\frac{3}{250.max(x)}, \frac{60}{max(x)}\right)$ and $\left(\frac{3}{25.max(x)}, \frac{60}{max(x)}\right)$ respectively (like the implementation of the authors.) The data is first normalize to have zero means and unit variance to have the same.

2.1 Experiments on images

For image data, we conduct two experiments. The CSC algorithm runs with 20 iterations. In the first experiment, we measure the trade-off between the quality of the reconstruction and the sparsity of the feature map via the change of parameter β. The value of β varies from 0.5 to 5 with an increment of 0.5. We construct 10 5-by-5 filters for a small dataset of 10 50-by-40 images. Figure 4 shows an image example in the dataset and the convergence of the objective function for $\beta = 1$. Some feature maps of an image in the dataset and the reconstruction error function with respect to β are shown in Figures 5 and 6 respectively.
We can see that when β is small, the algorithm tends to find a more accurate reconstruction of the original data and therefore the reconstruction error is small. However, we lose the sparsity of the feature map. When β is larger, we obtain a sparser feature map but the reconstruction becomes worse (the reconstructed image in Figure 5 loses details and tend to be blurrier when β increases). This verifies the trade-off between these two quantities: when we would like to obtain a good reconstruction, we have to sacrifice the sparsity. The data is therefore more difficult to compress and vice versa.

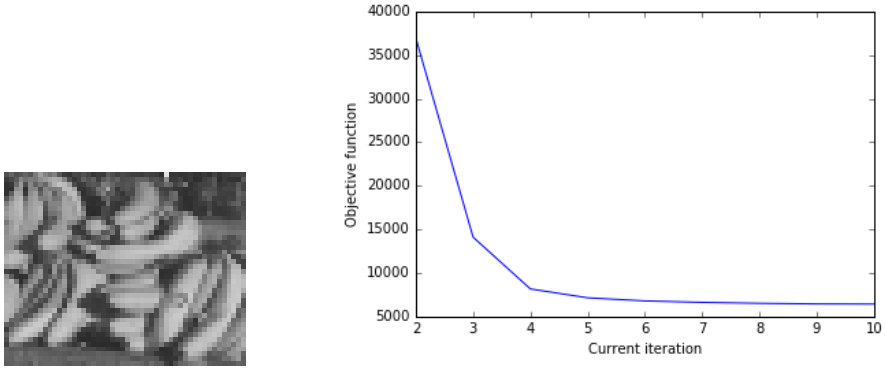

Figure 4: An example of the original image in the dataset (left) and the convergence of the objective function between the 2^{nd} and 10^{th} iterations for $\beta = 1$ (right.)

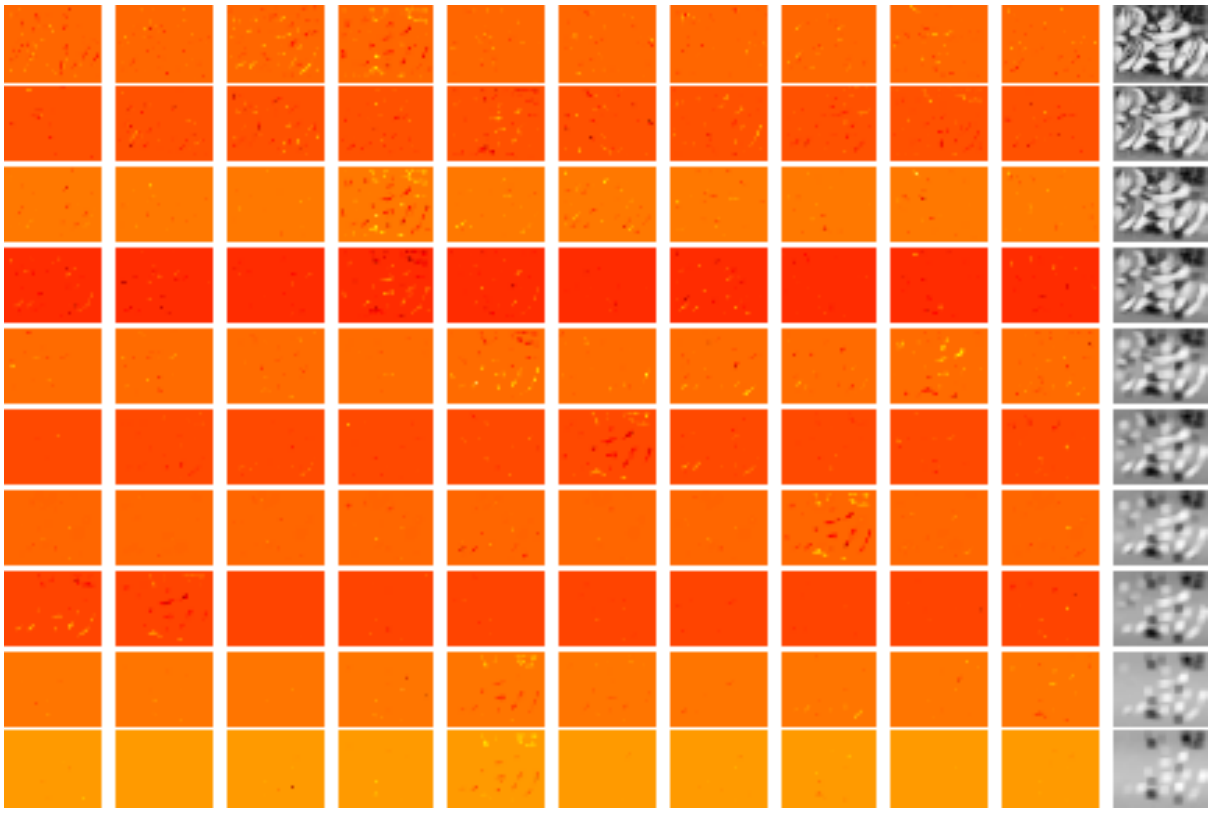

Figure 5: Learned feature maps with respect to the parameter β. From top to bottom: each row contains the 10 codes corresponding to the values of β from 0.5 to 5 (with increment 0.5). The rightmost cell is the reconstruction of the image shown in Figure 4.

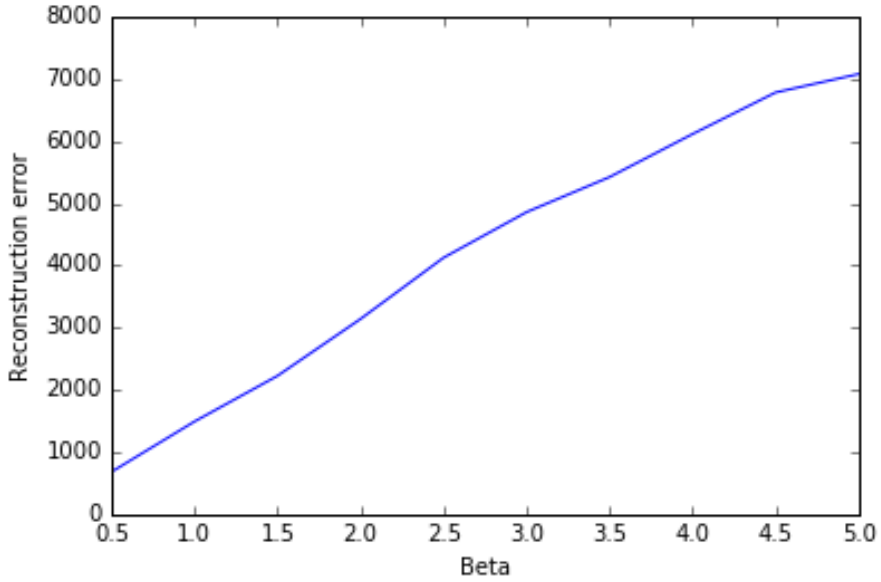

Figure 6: The reconstruction error with respect to the parameter β.

The goal of the second experiment is to verify the performance of new CSC solver on a relatively larger image dataset, we use a dataset that contains 10 100-by-80 images. The output data contains 100 11-by-11 filters. We choose the parameter $\beta = 1$ like the choice of the authors. The output filters, an image example, its feature map and its reconstruction are shown in Figures 7, 8 and 9 respectively.

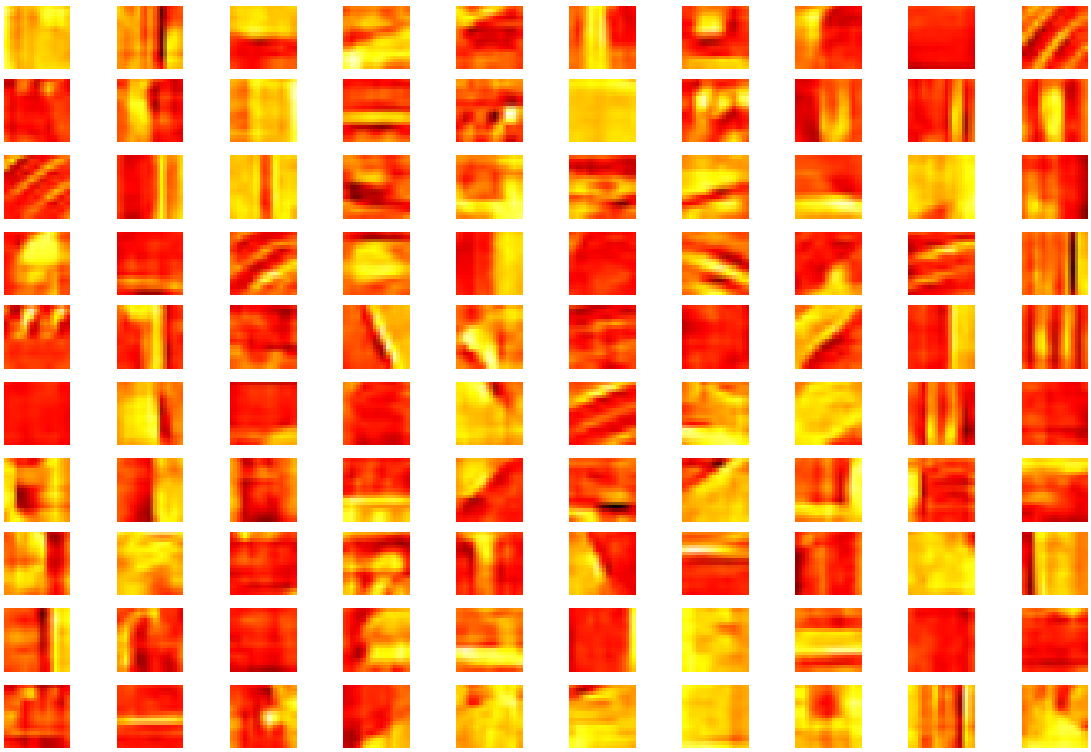

Figure 7: Learned filters of the medium image dataset.

Figure 8: An example of the original image in the dataset (left) and its reconstruction (right)

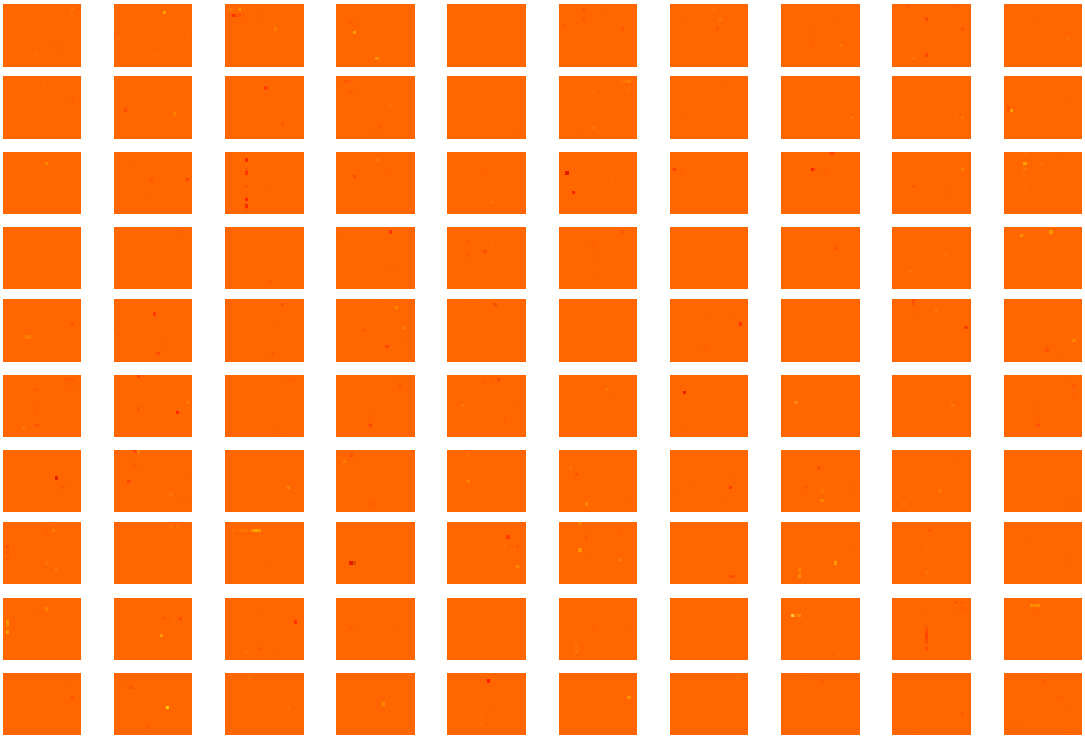

Figure 9: Learned sparse feature map of the medium image dataset. The orange regions that cover the most area in each code corresponds to zeros values. Due to the sparsity of the codes, there are only a few values that are much different than 0 and they are marked with red or yellow colors.

2.2 Experiments on EEG signals

In this section, we apply the method on real EEG dataset. The original dataset contains 50 signals of 5 types (10 signals of each type) corresponding to 5 sleep states. Each signal is sampled at rate $200Hz$ and lasts 30 seconds, which yields 6000 values. We set the number of kernels to 128 with the size of 201, and run the method with 30 iterations on the normalized data. The objective function actually decreases over the iterations and is shown in Figure 10.

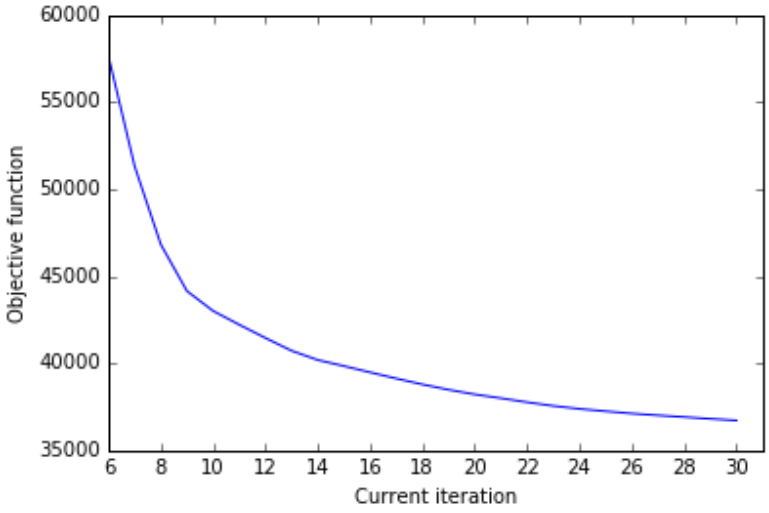

Figure 10: The objective function after each iteration. We eleminate the first 5 iterations because their objective functions are too large, making hard to view the other iterations.

We choose one signal of each EEG type and observe the reconstruction quality. The result is in Figure 11. We can observe that the reconstructed signal has some small difference in the high frequency components.

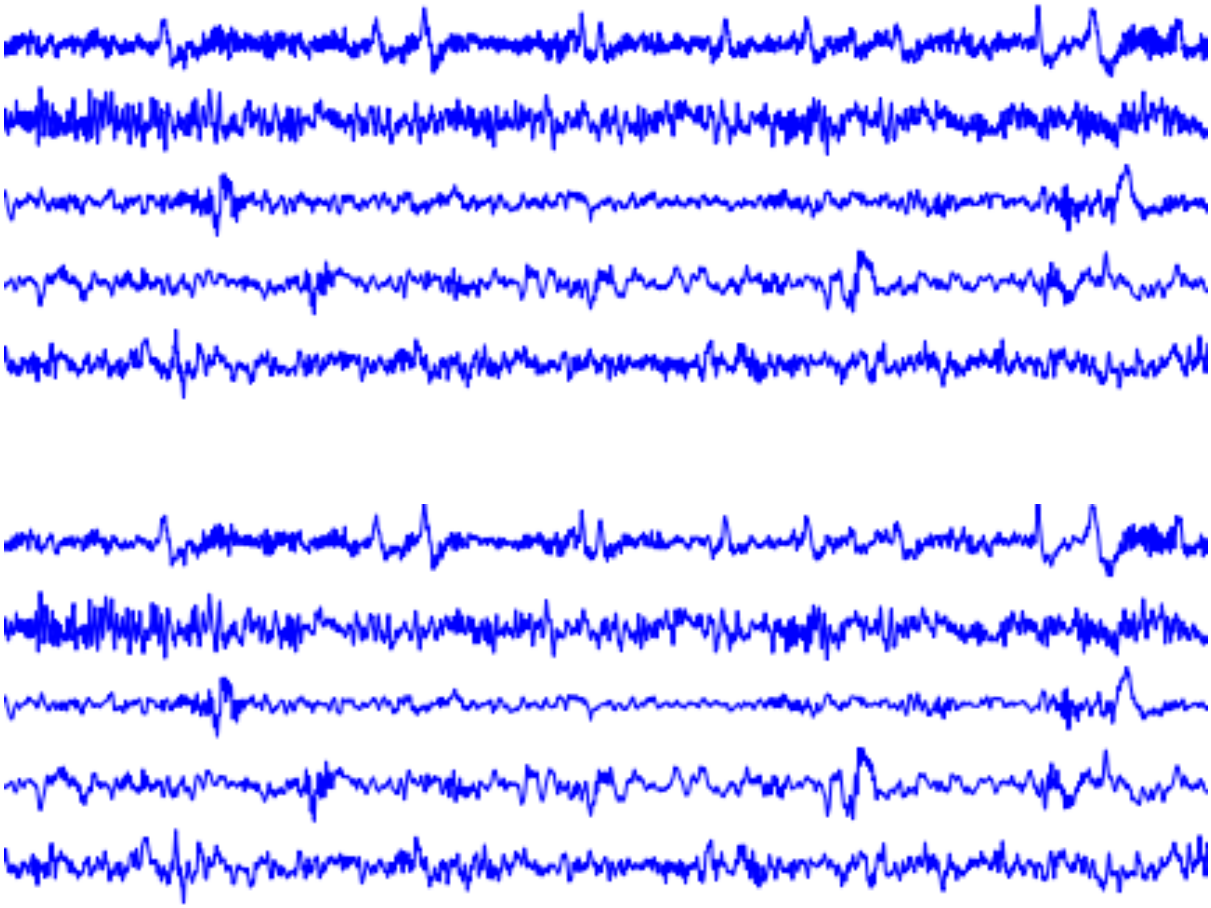

Figure 11: 5 example original signals for each EEG type (first 5 lines) and their corresponding reconstructions respectively (last 5 lines.)

Figures 12 and 13 sbow 128 kernels (each of which has size of 201) and 10 corresponding first codes for the first EEG series. We can observe that the filters are very diverse and this makes sense because of the different structures in 5 types of EEG signals. The codes for each EEG series is really sparse, this result is the same as when we do experiment with images.

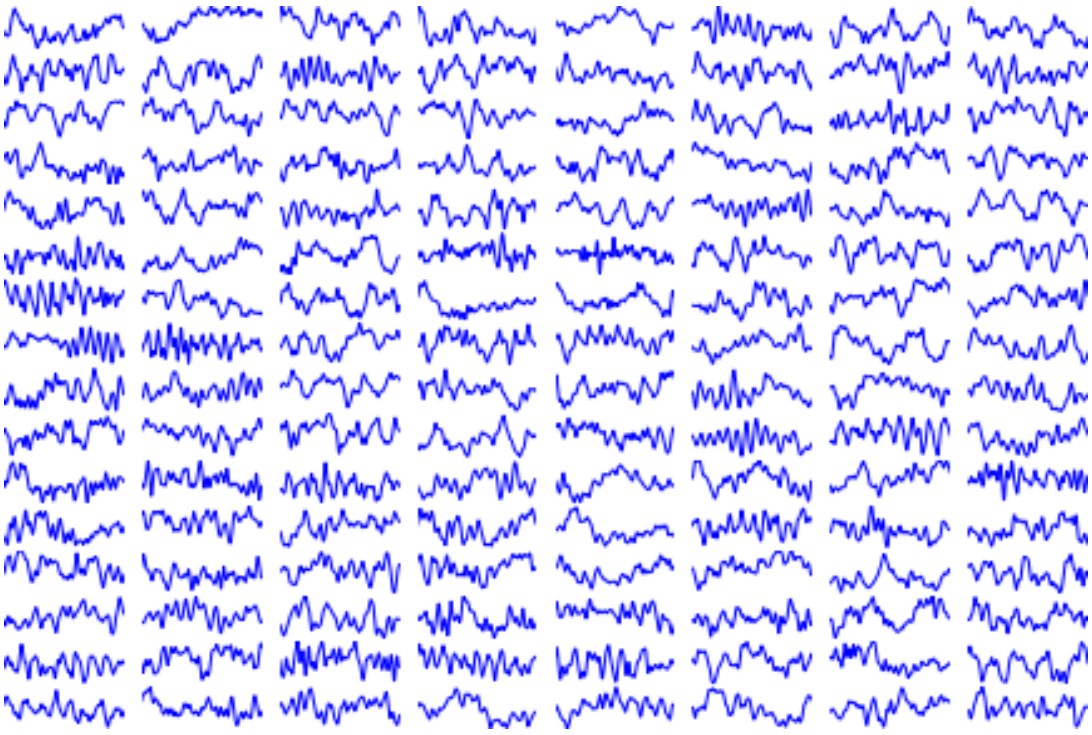

Figure 12: 128 common kernels of 50 given EEG signals.

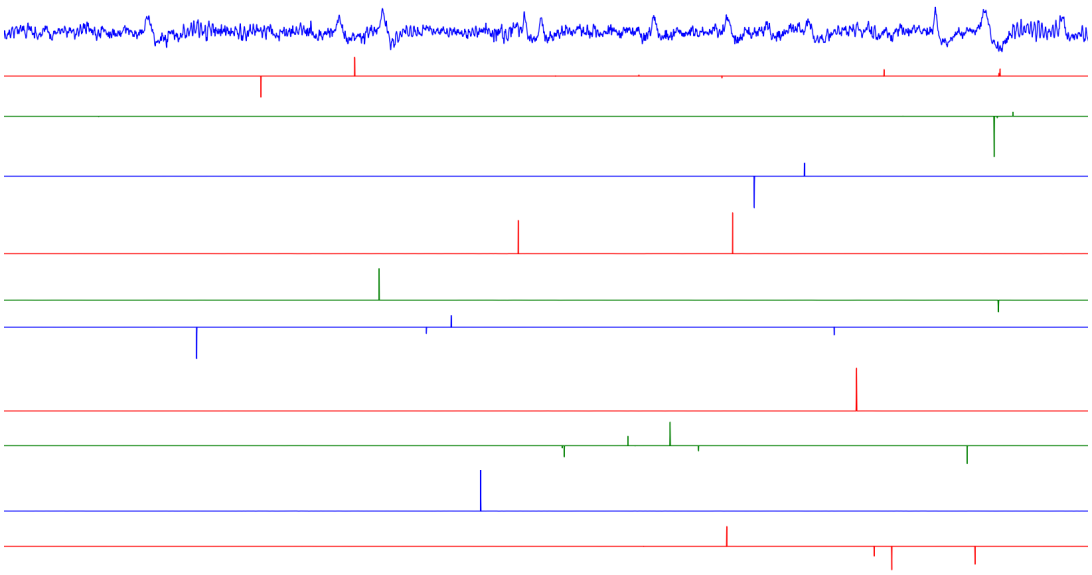

Figure 13: The first EEG series (the first line) and its first 10 codes (ten lines after). For each code, the flat line is the zero value that spans almost all elements. There are only a few elements that are different from zeros.

3 Conclusion and future works

In this project, we understood one of the newest method to solve the problem of convolutional sparse coding and implemented it to perform experiments on two kinds of data which are images and EEG signals. Results show that the method works well on both two data types.
A point that we intended to go deeper is to use the result of the CSC problem to learn and classify data. However, the main problem that we encounter is the high computational complexity of the algorithm (it took 12 hours to run on 50 EEG series.) Therefore, the filters and the codes are not sufficient yet to perform learning tasks on new data. Further research about how to parallelize this CSC solver still need to be considered to deal with large scale data.

References

[1] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[2] Hilton Bristow, Anders Eriksson, and Simon Lucey. Fast convolutional sparse coding. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 391–398, Washington, DC, USA, 2013. IEEE Computer Society.

[3] Hilton Bristow, Anders Eriksson, and Simon Lucey. Fast convolutional sparse coding. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 391–398, Washington, DC, USA, 2013. IEEE Computer Society.

[4] Hilton Bristow and Simon Lucey. Optimization methods for convolutional sparse coding. *CoRR*, abs/1406.2407, 2014.

[5] E.J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, Feb 2006.

[6] F. Heide, W. Heidrich, and G. Wetzstein. Fast and flexible convolutional sparse coding. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 5135–5143, June 2015.

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

[8] Bailey Kong and Charless C. Fowlkes. Fast convolutional sparse coding. Technical report, Department of Computer Science, University of California, Irvine, 2014.

[9] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *J. Mach. Learn. Res.*, 11:19–60, March 2010.

[10] Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014.

[11] Jianchao Yang, Kai Yu, Yihong Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801, June 2009.