Generalized Convolutional Sparse Coding With Unknown Noise

Yaqing Wang, *Member, IEEE*, James T. Kwok, *Fellow, IEEE*, and Lionel M. Ni, *Life Fellow, IEEE*

***Abstract*— Convolutional sparse coding (CSC) can learn representative shift-invariant patterns from data. However, existing CSC methods assume the Gaussian noise, which can be restrictive in some challenging applications. In this paper, we propose a generalized CSC model capable of handling complicated unknown noise. The noise is modeled by the Gaussian mixture model, which can approximate any continuous probability density function. The Expectation-Maximization algorithm is used to solve the resultant learning problem. For efficient optimization, the crux is to speed up the convolution in the frequency domain while keeping the other computations involving the weight matrix in the spatial domain. We design an efficient solver for the weighted CSC problem in the M-step. The dictionary and codes are updated simultaneously by an efficient nonconvex accelerated proximal gradient algorithm. The resultant procedure, called generalized convolutional sparse coding (GCSC), obtains the same space complexity and a smaller running time than existing CSC methods (which are limited to the Gaussian noise). Extensive experiments on synthetic and real-world noisy data sets validate that GCSC can model the noise effectively and obtain high-quality filters and representations.**

***Index Terms*— Convolutional sparse coding, noise modeling, Gaussian mixture model.**

I. Introduction

SPARSE coding learns an over-complete dictionary from a set of samples, and then represents each sample as a sparse combination (code) of the dictionary atoms. Though being popularly used in signal processing [1], [2] and computer vision [3], [4], sparse coding cannot capture shifted local patterns in the data. Hence, pre-processing (such as manual extracting patches from samples) and post-processing (such as aggregating patch representations back to a sample representation) are needed, otherwise redundant representations will be learned.

To alleviate this problem, convolutional sparse coding (CSC) [5] is a recent method which directly learns a shift-invariant dictionary. It replaces the multiplication between codes and dictionary by convolution, which can capture local patterns from different locations of the data. Each sample is represented as the sum of filters convolved with the corresponding codes. CSC has been successfully used in applications such as non-rigid structure motion recovery [6], image denoising and inpainting [7], music transcription [8], video deblurring [7], involving data types such as trajectories [6], images [9], audios [8], videos [7], hyperspectral and light field images [7], and biomedical data [10]–[12].

Learning in CSC is often performed by alternating the updates of the codes and dictionary using block coordinate descent (BCD) [13]. The various solvers mainly differ in how the code/dictionary update subproblems are solved. The pioneering deconvolutional network [5] uses gradient descent for both subproblems. ConvCoD [14] uses stochastic gradient descent for dictionary update and learns an encoder to output the codes. Recently, the alternating direction method of multipliers (ADMM) [15] has been commonly used [9], [16]–[19]. ADMM decomposes the code/dictionary update subproblems into even smaller ADMM subproblems that can be solved with closed-form solutions. It also allows performing faster convolutions in the frequency domain, while enforcing shift invariance and regularization in the spatial domain.

Most CSC methods use the square loss (also known as the ℓ_2-loss), and thus implicitly assume that the noise is Gaussian. This can be inappropriate in some real-world applications. For example, biomedical data often contain artifacts due to large variations in luminance/contrast and disturbance due to small living animals [20]. Though CSC has been popularly used on these data sets [11], [12], it cannot properly handle these noise patterns. Empirically, as the target biomedical structures are often tiny and delicate, the presence of noise can heavily affect the quality of the learned filters and representations [12].

Instead of using the Gaussian noise, Jas *et al.* [12] recently proposed the alpha-stable CSC (αCSC) algorithm, which models the noise by the alpha-stable distribution [21]. This distribution includes the normal distribution and many heavy-tailed distributions, and is more robust to noise and outliers. However, the alpha-stable distribution does not have an analytical form for its probability density function, and its inference needs to be approximated by the computationally expensive Markov chain Monte Carlo (MCMC) [22] procedure. Moreover, the algorithm in [12] can only use the symmetric alpha-stable distribution, which cannot model distributions such as the Laplace distribution, Levy distribution (which is an asymmetric alpha-stable distribution),

Manuscript received February 3, 2019; revised November 11, 2019; accepted March 4, 2020. Date of publication March 27, 2020; date of current version April 6, 2020. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Christophoros Nikou. *(Corresponding author: Yaqing Wang.)*

Yaqing Wang is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, also with the Business Intelligence Lab, Baidu Research, Beijing 100085, China, and also with the National Engineering Laboratory of Deep Learning Technology and Application, Beijing 100085, China (e-mail: ywangcy@connect.ust.hk).

James T. Kwok and Lionel M. Ni are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong.

This article has supplementary downloadable material available at ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TIP.2020.2980980

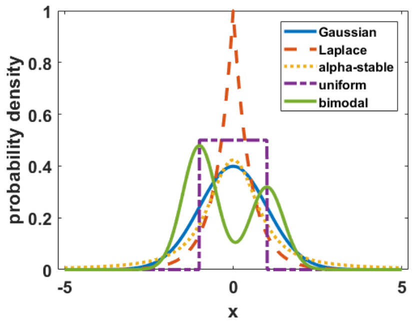


Fig. 1. Probability density functions of some popular used distributions for noise modeling.

bimodal and multimodal distributions. A visual comparison of these distributions is shown in Figure 1.

In this paper, we propose the *generalized CSC* (GCSC) model, which allows CSC to handle an unknown noise distribution. Specifically, the noise is modeled by the Gaussian mixture model (GMM) [23], which can approximate any continuous probability density function. The proposed model is then optimized by the Expectation-Maximization (EM) algorithm [24]. However, as will be seen in the sequel, the maximization step involves a weighted variant of the CSC problem, which cannot be efficiently solved by existing algorithms such as BCD and ADMM. Besides, the resultant weight matrix cannot utilize efficient convolution in the frequency domain.

To address these problems, we develop a new solver that updates the dictionary and codes together by a nonconvex accelerated proximal algorithm [25]. Moreover, we reformulate the problem so that convolutions can still be performed efficiently in the frequency domain, while computations involving the weight matrix are performed in the spatial domain. The resultant algorithm has comparable time and space complexities as the state-of-the-art CSC methods (which use the square loss). Extensive experiments on synthetic and real-world noisy data sets show that the proposed algorithm can model complex underlying noise, and obtains high-quality filters and representations.

In summary, the contributions of this paper are as follows:

1) A generalized CSC model, which allows modeling of an unknown and possibly complicated noise;
2) An efficient solver for the resultant optimization problem, with time and space complexities comparable with the state-of-the-art CSC methods using square loss;
3) Experiments on a number of synthetic and real-world noisy data sets including hyperspectral image data, local field potential data, and retinal image data show that the proposed method obtains high-quality filters and representations, and is empirically more efficient than state-of-the-art CSC methods.

The rest of the paper is organized as follows. Section II provide brief reviews on CSC and the proximal algorithm. Section III describes the proposed method, Section IV presents the experimental results, and the last section gives some concluding remarks and future directions.

Notations: For vector $a \in \mathbb{R}^m$, its ith element is denoted $a(i)$, its ℓ_2-norm is $\|a\|_2 = \sqrt{\sum_{i=1}^m (a(i))^2}$, and its ℓ_1-norm is $\|a\|_1 = \sum_{i=1}^m |a(i)|$. Diag$(a)$ reshapes a to a diagonal matrix with the elements of a on the diagonal. Given another vector $b \in \mathbb{R}^n$, the convolution $a * b$ produces a vector $c \in \mathbb{R}^{m+n-1}$, with $c(k) = \sum_{j=\max(1,k+1-n)}^{\min(k,m)} a(j)b(k-j+1)$. For matrix A, $A^\top$ denotes its transpose, $A^\star$ denotes its complex conjugate, and $A^\dagger$ is its conjugate transpose (i.e., $A^\dagger = (A^\top)^\star$). $\odot$ denotes the pointwise product. The identity matrix is denoted by I. $\mathcal{F}(x)$ is the fast Fourier transform (FFT) that maps x from the spatial domain to the frequency domain, while $\mathcal{F}^{-1}(x)$ is the inverse operator which maps $\mathcal{F}(x)$ back to x.

II. Related Work

A. Convolutional Sparse Coding

Given N samples x_i's, where each $x_i \in \mathbb{R}^P$, CSC learns a dictionary of K filters d_k's, each of length M, such that x_i is represented as

$$\tilde{x}_i = \sum_{k=1}^K d_k * z_{ik}. \tag{1}$$

Here, $*$ is the (spatial) convolution operation, and z_{ik}'s are the codes for x_i, each of length P. The filters and codes are obtained by solving the following optimization problem

$$\min_{\{d_k\}\in\mathcal{D},\{z_{ik}\}} \sum_{i=1}^N \left(\frac{1}{2}\left\| x_i - \sum_{k=1}^K d_k * z_{ik} \right\|_2^2 + \sum_{k=1}^K \beta \|z_{ik}\|_1 \right), \tag{2}$$

where $\mathcal{D} = \{D : \|d_k\|_2 \leq 1, k = 1,\ldots,K\}$ ensures that the filters are normalized, and the ℓ_1-regularizer encourages the codes to be sparse.

To solve (2), block coordinate descent (BCD) [13] is typically used [5], [9], [14], [16]–[18]. The dictionary and codes are updated in an alternating manner as follows.

1) Dictionary Update: Given z_{ik}'s, d_k's are obtained as

$$\min_{\{d_k\}\in\mathcal{D}} \frac{1}{2}\sum_{i=1}^N \left\| x_i - \sum_{k=1}^K d_k * z_{ik} \right\|_2^2. \tag{3}$$

Convolution can be performed much faster in the frequency domain via the convolution theorem[1] [26]. Combining this with the use of Parseval's theorem[2] [26] and linearity of FFT, problem (3) is reformulated in [9], [16], [17] as:

$$\min_{\{d_k\}} \frac{1}{2P}\sum_{i=1}^N \left\| \mathcal{F}(x_i) - \sum_{k=1}^K \mathcal{F}(C^\top d_k) \odot \mathcal{F}(z_{ik}) \right\|_2^2 \tag{4}$$
$$\text{s.t.} \quad \|C\mathcal{F}^{-1}(\mathcal{F}(C^\top d_k))\|_2^2 \leq 1, \forall k,$$

where $C \in \mathbb{R}^{M\times P}$ with $C(i,i) = 1$ and $C(i,j) = 0$ for $i \neq j$, crops the extra dimension to recover the original spatial support, and $C^\top$ pads d_k to be P-dimensional. The constraint normalizes all filters to unit norm.

[1] $\mathcal{F}(d_k * z_{ik}) = \mathcal{F}(C^\top d_k) \odot \mathcal{F}(z_{ik})$, where $C^\top$ zero-pads d_k to be of the same size as z_{ik}.

[2] For $a \in \mathbb{R}^P$, $\|a\|_2^2 = \frac{1}{P}\|\mathcal{F}(a)\|_2^2$.

Algorithm 1 Nonconvex Inexact Accelerated Proximal Gradient (niAPG) Algorithm [25]

```
1:  Initialize: $x^0 = x^1 = 0, \eta$;
2:  for $\tau = 1, \ldots, I$ do
3:      $v^\tau = x^\tau + \frac{\tau-1}{\tau+2}(x^\tau - x^{\tau-1})$;
4:      $\Delta^\tau = \max_{t=\max(1,\tau-5),\ldots,\tau} F(x^t)$;
5:      if $F(v^\tau) \geq \Delta^\tau$ then
6:          $v^\tau = x^\tau$;
7:      end if
8:      $x^{\tau+1} = \text{prox}_{\eta r}(v^\tau - \eta \nabla f(v^\tau))$;
9:  end for
10: return $x^{I+1}$.
```

2) Code Update: Given d_k's, the corresponding z_{ik}'s are obtained as

$$\min_{\{z_{ik}\}} \frac{1}{2}\left\| x_i - \sum_{k=1}^{K} d_k * z_{ik} \right\|_2^2 + \beta \sum_{k=1}^{K} \|z_{ik}\|_1.$$

Similar to dictionary update, it is more efficient to perform convolutions in the frequency domain, as:

$$\min_{\{z_{ik}\}} \frac{1}{2P}\left\| \mathcal{F}(x_i) - \sum_{k=1}^{K} \mathcal{F}(C^\top d_k) \odot \mathcal{F}(z_{ik}) \right\|_2^2 + \beta \sum_{k=1}^{K} \|z_{ik}\|_1.$$

The alternating direction method of multipliers (ADMM) [15] has been commonly used for solving the code update and dictionary update subproblems [9], [16]–[18]. Each ADMM subproblem has a simple closed-form solution. Besides, by introducing auxiliary variables, ADMM separates the computations in the frequency domain (involving convolutions) and the spatial domain (involving the ℓ_1 regularizer and unit norm constraint).

B. Proximal Algorithm

The proximal algorithm [27] is used to solve composite optimization problems of the form

$$\min_x F(x) \equiv f(x) + r(x),$$

where f is smooth, r is nonsmooth, and both are convex. To make the proximal algorithm efficient, its proximal step

$$\text{prox}_{\eta r}(\cdot) = \arg\min_x \frac{1}{2}\|x - \cdot\|_2^2 + \eta r(x),$$

where η is the stepsize, has to be inexpensive.

Recently, the proximal algorithm has been extended to nonconvex problems where both f and r can be nonconvex. A state-of-the-art proximal algorithm is the nonconvex inexact accelerated proximal gradient (niAPG) algorithm [25] shown in Algorithm 1. It can efficiently converge to a critical point of the objective.

III. Proposed Method

The square loss in (2) implicitly assumes normally distributed noise. We relax this in Section III-A, and instead assume that the noise is generated from the flexible Gaussian mixture model (GMM) [23]. As is common in GMMs, we use the Expectation-Maximization (EM) algorithm for inference [24] (Section III-B). However, the resultant M-step involves a difficult weighted CSC problem which cannot be solved by existing algorithms. To address this problem, in Section III-C, we design an efficient solver based on the niAPG algorithm [25] introduced in Section II-B. This allows the convolutions to be computed more efficiently in the frequency domain, while leaving computations involving the weight matrix to be performed in the spatial domain. The complete algorithm is summarized in Section III-D. Complexity analysis is performed in Section III-E, and Section III-F discusses its relationship with existing CSC algorithms. A schematic diagram of the proposed method is shown in Figure 2.

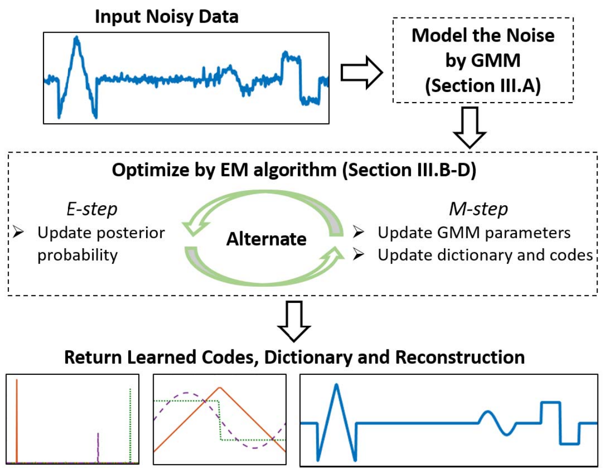

Input Noisy Data
Model the Noise by GMM (Section III.A)
Optimize by EM algorithm (Section III.B-D)
E-step
Update posterior probability
Alternate
M-step
Update GMM parameters
Update dictionary and codes
Return Learned Codes, Dictionary and Reconstruction

Fig. 2. Schematic diagram of the proposed GCSC algorithm.

A. Noise Modeling by GMM

We assume that the noise ϵ_i associated with sample x_i follows the GMM distribution:

$$p(\epsilon_i) = \sum_{g=1}^{G} p(\epsilon_i | \phi_i = g) p(\phi_i = g). \tag{5}$$

Here, G is the number of Gaussian components, ϕ_i is the latent variable denoting which Gaussian component ϵ_i belongs to, and π_g's are the mixing coefficients with $\sum_{g=1}^{G} \pi_g = 1$. Obviously, (5) includes the commonly used normal distribution.

The variable ϕ_i follows the multinomial distribution Multinomial($\{\pi_g\}$), and the conditional distribution of ϵ_i given $\phi_i = g$ follows the normal distribution $\mathcal{N}(\mu_g, \Sigma_g)$ with mean μ_g and diagonal covariance matrix $\Sigma_g = \text{Diag}(\sigma_g^2(1), \ldots, \sigma_g^2(P))$. The ℓ_1-regularizer in (2) corresponds to the use of the Laplace prior (Laplace(0, $\frac{1}{\beta}$)) on each pth element of z_{ik}: $p(z_{ik}(p)) = \frac{\beta}{2}\exp(-\beta|z_{ik}(p)|)$.

B. Optimization by Expectation-Maximization (EM) Algorithm

Let Θ be the collection of the GMM parameters (π_g's, μ_g's, Σ_g's, d_k's and z_{ik}'s). From (5), the log posterior probability for Θ is:

$$\log \mathcal{P} = \sum_{i=1}^{N} \left(\log \sum_{g=1}^{G} p(x_i|\phi_i = g) + \log \sum_{g=1}^{G} p(\phi_i = g) + \sum_{k=1}^{K} \sum_{p=1}^{P} \log p(z_{ik}(p)) \right). \quad (6)$$

In the EM algorithm, the E-step computes $p(\phi_i = g|x_i)$, which is the posterior probability that ϕ_i belongs to the gth Gaussian given x_i. Using Bayes rule, we have

$$p(\phi_i = g|x_i) = \frac{\pi_g p(x_i|\phi_i = g)}{\sum_{g=1}^{G} \pi_g p(x_i|\phi_i = g)}, \quad (7)$$

where $x_i|(\phi_i = g) \sim \mathcal{N}(\tilde{x}_i + \mu_g, \Sigma_g)$ with $\tilde{x}_i$ defined in (1).

The M-step obtains Θ by maximizing the following upper bound of $\log \mathcal{P}$:

$$\begin{aligned} &\arg\max_{\Theta} \sum_{i=1}^{N} \left(\sum_{g=1}^{G} \gamma_{gi} \log \frac{p(x_i, \phi_i)}{\gamma_{gi}} + \sum_{k=1}^{K} \sum_{p=1}^{P} \log p(z_{ik}(p)) \right) \\ &= \arg\max_{\Theta} \sum_{i=1}^{N} \left(\sum_{g=1}^{G} \gamma_{gi} \log p(x_i|\phi_i = g) + \beta \sum_{k=1}^{K} \|z_{ik}\|_1 \right) \\ &= \arg\max_{\Theta} \sum_{i=1}^{N} \left(\beta \sum_{k=1}^{K} \|z_{ik}\|_1 + \sum_{g=1}^{G} \gamma_{gi} \log \pi_g - \frac{\gamma_{gi}}{2} \log(|\Sigma_g|) \right. \\ &\left. - \frac{\gamma_{gi}}{2} \left(x_i - \sum_{k=1}^{K} d_k * z_{ik} - \mu_g \right)^{\top} \Sigma_g^{-1} \left(x_i - \sum_{k=1}^{K} d_k * z_{ik} - \mu_g \right) \right), \end{aligned} \quad (8)$$

where $\gamma_{gi} = p(\phi_i = g|x_i)$.

1) Updating π_g's, μ_g's, Σ_g's in Θ: Given d_k's, z_{ik}'s and γ_{gi}'s, we obtain π_g's, μ_g's and Σ_g's by optimizing (8) as:

$$\max_{\{\pi_g, \mu_g, \Sigma_g\}} \sum_{i=1}^{N} \sum_{g=1}^{G} \left(\gamma_{gi} \log \pi_g - \frac{\gamma_{gi}}{2} \log(|\Sigma_g|) - \frac{\gamma_{gi}}{2} (x_i - \tilde{x}_i - \mu_g)^{\top} \Sigma_g^{-1} (x_i - \tilde{x}_i - \mu_g) \right).$$

Setting the derivative of the objective to zero, the following closed-form solutions can be easily obtained:

$$\pi_g = \frac{1}{N} \sum_{i=1}^{N} \gamma_{gi}, \quad \mu_g = \frac{\sum_{i=1}^{N} \gamma_{gi} x_i}{\sum_{i=1}^{N} \gamma_{gi}},$$
$$\Sigma_g = \frac{\sum_{i=1}^{N} \gamma_{gi} (x_i - \tilde{x}_i - \mu_g)(x_i - \tilde{x}_i - \mu_g)^{\top}}{\sum_{i=1}^{N} \gamma_{gi}}. \quad (9)$$

2) Updating d_k's and z_{ik}'s in Θ: Given π_g's, μ_g's, Σ_g's and γ_{gi}'s, we obtain d_k's and z_{ik}'s from (8) as:

$$\min_{\{d_k\} \in \mathcal{D}, \{z_{ik}\}} \sum_{i=1}^{N} \left(\beta \sum_{k=1}^{K} \|z_{ik}\|_1 - \sum_{g=1}^{G} \frac{\gamma_{gi}}{2} \left(x_i - \sum_{k=1}^{K} d_k * z_{ik} - \mu_g \right)^{\top} \cdot \Sigma_g^{-1} \left(x_i - \sum_{k=1}^{K} d_k * z_{ik} - \mu_g \right) \right).$$

This can be rewritten as

$$\min_{\{d_k\} \in \mathcal{D}, \{z_{ik}\}} F(\{d_k\}, \{z_{ik}\}) \equiv f(\{d_k\}, \{z_{ik}\}) + r(\{d_k\}, \{z_{ik}\}), \quad (10)$$

where

$$f(\{d_k\}, \{z_{ik}\}) \equiv \frac{1}{2} \sum_{i=1}^{N} \sum_{g=1}^{G} \|w_{gi} \odot (x_i - \sum_{k=1}^{K} d_k * z_{ik} - \mu_g)\|_F^2, \quad (11)$$

and $r(\{d_k\}, \{z_{ik}\}) \equiv \beta \sum_{k=1}^{K} \|z_{ik}\|_1 + I_{\mathcal{D}}(\{d_k\})$. Here, $w_{gi}(p) = \sqrt{\frac{\gamma_{gi}}{\sigma_g^2(P)}}$, and $I_{\mathcal{D}}(\cdot)$ is the indicator function on $\mathcal{D}$ (i.e., $I_{\mathcal{D}}(\{d_k\}) = 0$ if $\{d_k\} \in \mathcal{D}$, and ∞ otherwise).

Compared with the standard CSC problem in (2), problem (10) can be viewed as a weighted CSC problem (with weights w_{gi}'s). The CSC variant in [9] assigns weights on $\sum_{k=1}^{K} d_k * z_{ik}$, while in (11) the weights are on $x_i - \sum_{k=1}^{K} d_k * z_{ik} - \mu_g$. Another CSC variant in [12] also leads to a weighted CSC problem. However, the authors there mentioned that it is not clear how to solve a weighted CSC problem in the frequency domain. Instead, they resorted to solving it in the spatial domain, which is less efficient as discussed in Section II-A.

C. Solving Subproblem (10) in M-Step

The weights w_{gi}'s in (11) prevent us from transforming the objective in (10) to the frequency domain. Recall that (3) is transformed to (4) by first transforming terms in the ℓ_2-norm to the frequency domain by Parseval's theorem, separately computing $\mathcal{F}(x_i)$ and $\mathcal{F}(\sum_{k=1}^{K} d_k * z_{ik})$ by linearity of FFT, and then replacing the convolution in $\mathcal{F}(\sum_{k=1}^{K} d_k * z_{ik})$ by pointwise product using the convolution theorem. However, with the w_{gi}'s, $\mathcal{F}(w_{gi} \odot (\sum_{k=1}^{K} d_k * z_{ik}))$ can no longer use the convolution theorem to speed up. Thus, the key in designing an efficient solver is to transform only terms involving convolutions to the frequency domain, while leaving the weights w_{gi}'s in the spatial domain. Hence, we will replace the $x_i - \sum_{k=1}^{K} d_k * z_{ik} - \mu_g$ term in (11) by $\mathcal{F}^{-1}(\mathcal{F}(x_i - \mu_g) - \sum_{k=1}^{K} \mathcal{F}(C^{\top} d_k) \odot \mathcal{F}(z_{ik}))$.

To solve the weighted CSC problem (10), we design an efficient solver based on the niAPG algorithm [25] in Section II-B. The core steps in niAPG are computing (i) the gradient $\nabla f(\cdot)$ w.r.t. d_k's and z_{ik}'s, and (ii) the proximal step $\text{prox}_{\eta r}(\cdot)$. We first show that the gradient $\nabla f(\cdot)$ w.r.t. d_k's and z_{ik}'s can be obtained by the following Proposition. Proof is in Appendix A.

Proposition 1: For f in (11),

$$\frac{\partial f(\{d_k\}, \{z_{ik}\})}{\partial d_k} = -C \mathcal{F}^{-1} \left(\sum_{i=1}^{N} (\mathcal{F}(z_{ik}))^{\star} \odot \mathcal{F}(u_i) \right), \quad (12)$$

$$\frac{\partial f(\{d_k\}, \{z_{ik}\})}{\partial z_{ik}} = -\mathcal{F}^{-1}((\mathcal{F}(d_k))^{\star} \odot \mathcal{F}(u_i)), \quad (13)$$

where $u_i = \sum_{g=1}^{G} w_{gi} \odot w_{gi} \odot \mathcal{F}^{-1}(\mathcal{F}(x_i - \mu_g) - \sum_{k=1}^{K} \mathcal{F}(C^{\top} d_k) \odot \mathcal{F}(z_{ik}))$.

As for the second issue (computing the proximal step $\text{prox}_{\eta r}(\cdot)$), the following Lemma shows that it can be simplified as $r(\{d_k\}, \{z_{ik}\})$ is separable.

Lemma 2 [27]: For separable $r(x, y)$ (i.e., $r(x, y) = r_1(x) + r_2(y)$), $\text{prox}_r(v, w) = \text{prox}_{r_1}(v) + \text{prox}_{r_2}(w)$.

Algorithm 2 WeightedCSC($\{d_k^0\}$, $\{z_{ik}^0\}$)

```
1: Initialize: $d_k^1 = d_k^0, \forall k$, $z_{ik}^1 = z_{ik}^0, \forall i, k$, $\eta$;
2: for $\tau = 1, \dots, I$ do
3:     $u_k^\tau = d_k^\tau + \frac{\tau-1}{\tau+2}(d_k^\tau - d_k^{\tau-1}), \forall k$;
4:     $v_{ik}^\tau = z_{ik}^\tau + \frac{\tau-1}{\tau+2}(z_{ik}^\tau - z_{ik}^{\tau-1}), \forall i, k$;
5:     $\Delta_\tau = \max_{t=\max(1,\tau-5),\dots,\tau} F(\{u_k^t\},\{v_{ik}^t\})$;
6:     if $F(\{u_k^\tau\},\{v_{ik}^\tau\}) \geq \Delta_\tau$ then
7:         $u_k^\tau = d_k^\tau, \forall k$;
8:         $v_{ik}^\tau = z_{ik}^\tau, \forall i, k$;
9:     end if
10:    compute $d_k^{\tau+1} = \text{prox}_{\eta I_{\mathcal{D}}}(u_k^\tau - \eta \frac{\partial f(\{d_k\},\{z_{ik}\})}{\partial d_k})$ using (12) and (14);
11:    compute $z_{ik}^{\tau+1} = \text{prox}_{\beta\eta\|\cdot\|_1}(v_{ik}^\tau - \eta \frac{\partial f(\{d_k\},\{z_{ik}\})}{\partial z_{ik}})$ using (13) and (15);
12: end for
13: return $\{d_k^{I+1}\}$ and $\{z_{ik}^{I+1}\}$.
```

Using Lemma 2, the component proximal steps can be easily computed in closed form as [27]:

$$\text{prox}_{\eta I_{\mathcal{D}}}(d_k) = d_k / \max(\|d_k\|_2, 1), \tag{14}$$

$$\text{prox}_{\beta\eta\|\cdot\|_1}(z_{ik}(p)) = \text{sign}(z_{ik}(p)) \odot \max(|z_{ik}(p)| - \beta\eta, 0). \tag{15}$$

The resultant procedure for solving the weighted CSC subproblem (10), which is based on Algorithm 1, is shown in Algorithm 2.

D. Complete Algorithm

The complete EM algorithm, which will be called generalized CSC (GCSC), is shown in Algorithm 3. After initialization (step 1), we alternate the E-step (step 3), which estimates the posterior probability using the current model parameters, and the M-step (step 4), which updates the GMM parameters, dictionary and codes until convergence. The learned dictionary and codes can then be used to reconstruct the samples (step 6) or be directly used for subsequent tasks.

E. Complexity Analysis

In each EM iteration, the E-step in (7) takes $O(GNP)$ time. The M-step is dominated by gradient computations in (15) and (16). These take $O(NKP \log P)$ time for the underlying FFT and inverse FFT operations, and $O(GNP)$ time for the pointwise product. Thus, each EM iteration takes a total of $O(I(GNP + NKP \log P))$ time, where I is the number of inner iterations for solving the weighted CSC problem. Empirically, I is around 50. As for space, this is dominated by the K P-dimensional codes for each of the N samples, leading to a space complexity of $O(NKP)$.

F. Discussion With Existing CSC Algorithms

Table I compares GCSC with the existing batch CSC algorithms. GCSC is the most flexible in modeling the noise. On the other hand, αCSC uses the symmetric alpha-stable distribution, and all others use the simple normal distribution.

As for optimization, all algorithms except GCSC use BCD, which alternates the updates of codes and dictionary. Moreover, most of them then update the codes and dictionary separately by ADMM. In contrast, GCSC uses niAPG to directly update the codes and dictionary together.

All methods in Table I take $O(NKP)$ space. As for time, the state-of-the-art batch CSC method is SBCSC [19] which takes $O(T^{\text{alt}} I \cdot (NK^3P + NKPM + K^2M))$, where T^{alt} is the number of alternating iterations between codes and dictionary update subproblems in BCD. In contrast, the proposed GCSC takes $O(T^{\text{EM}} I \cdot (GNP + NKP \log P))$ time, where T^{EM} is the number of EM iterations. Usually $G < K^3$ and $\log(P) < M$. Moreover, empirically, $T^{\text{EM}} < 10$ is enough for GCSC to obtain good results, while T^{alt} needs to be at least 100. Thus, GCSC can be empirically faster than SBCSC. This will be further validated in Section IV-B.

Algorithm 3 Generalized CSC With GMM Loss (GCSC)

```
Input: A set of $N$ noisy samples $\{x_i\}$, number of filters $K$, number of Gaussian components $G$;
1: Initialize: dictionary $d_k^0, k = 1, \dots, K$ by random Gaussian; codes $z_{ik}^0 = 0, i = 1, \dots, N, k = 1, \dots, K$; random GMM parameters $\{\pi_g, \mu_g, \Sigma_g\}, g = 1, \dots, G$;
2: for $t = 1, \dots, T$ do
3:     // E-step
       compute posterior probability $p(\phi_i^t = g | x_i)$ by (7);
4:     // M-step
       update GMM parameters $\{\pi_g^t, \mu_g^t, \Sigma_g^t\}$ by (9);
       update dictionary and codes using Algorithm 2 as $\{d_k^t\},\{z_{ik}^t\}$ =WeightedCSC($\{d_k^{t-1}\},\{z_{ik}^{t-1}\}$);
5: end for
6: reconstruct denoised sample $\tilde{x}_i = \sum_{k=1}^K d_k^T * z_{ik}^T$ for $x_i$;
7: return $\{d_k^T\},\{z_{ik}^T\}$ and $\{\tilde{x}_i\}$.
```

IV. Experiments

In this section, we perform experiments on both synthetic and real-world data sets. Section IV-A introduces the experimental setup. Section IV-B compares the proposed GCSC on synthetic data with various kinds of noise. This is then followed by experiments on real-world data sets, including hyperspectral image data (Section IV-C), local field potential data (Section IV-D) and retinal image data (Section IV-E). Results show that GCSC can model complicated noise, provide cleaner reconstructions, and learn useful filters and representations.

A. Experimental Setup

Baselines: For the proposed GCSC, the number of mixture components G is selected automatically by the pruning strategy in [30]. We start with a relatively large G ($G = 10$). At each EM iteration, the relative differences in diagonal covariance matrices $\{\Sigma_g = \text{Diag}(\sigma_g^2(1), \dots, \sigma_g^2(P))\}_{g=1,\dots,G}$ between any two Gaussian components are computed:

$$\sum_{p=1}^P \frac{|\sigma_a^2(p) - \sigma_b^2(p)|}{\sigma_a^2(p) + \sigma_b^2(p)}, \ \forall a, b \in \{1, \dots, G\}.$$

TABLE I

COMPARISON BETWEEN THE PROPOSED GCSC AND EXISTING BATCH CSC ALGORITHMS. HERE, N IS THE NUMBER OF TRAINING SAMPLES, K IS THE NUMBER OF FILTERS, P IS THE SAMPLE DIMENSIONALITY, M IS THE FILTER LENGTH IN THE SPATIAL DOMAIN, G IS THE NUMBER OF GAUSSIAN COMPONENTS USED IN OUR GMM MODEL. T^{EM} IS THE NUMBER OF EM ITERATIONS, T^{alt} IS THE NUMBER OF ALTERNATING ITERATIONS BETWEEN CODES AND DICTIONARY UPDATE SUBPROBLEMS IN BCD, AND I IS THE NUMBER OF INNER ITERATIONS FOR EACH SUBPROBLEM. αCSC NEEDS MARKOV CHAIN MONTE CARLO (MCMC) IN ITS E-STEP, WHOSE TIME COMPLEXITY IS WRITTEN AS O(MCMC) DUE TO A LACK OF ALGORITHMIC DETAILS IN [12]

	convolution	noise	optimization	space	time
DeconvNet [5]	spatial	Gaussian	BCD, updating codes and dictionary by gradient descent	$O(NKP)$	$O(T^{\text{alt}}I \cdot (NK^2P^2M))$
ConvCod [14]	spatial	Gaussian	BCD, updating codes by encoder and dictionary by gradient descent	$O(NKP)$	$O(T^{\text{alt}}I \cdot (NK^2P^2M))$
FCSC [16]	frequency	Gaussian	BCD, updating codes and dictionary by ADMM	$O(NKP)$	$O(T^{\text{alt}}I \cdot (NK^3P + NKP\log P))$
FFCSC [9]	frequency	Gaussian	BCD, updating codes and dictionary by ADMM	$O(NKP)$	$O(T^{\text{alt}}I \cdot (NK^2P + NKP\log P))$
CBPDN [17]	frequency	Gaussian	BCD, updating codes and dictionary by ADMM	$O(NKP)$	$O(T^{\text{alt}}I \cdot (N^2KP + NKP\log P))$
CONSENSUS [18]	frequency	Gaussian	BCD, updating codes and dictionary by ADMM	$O(NKP)$	$O(T^{\text{alt}}I \cdot (NKP^2 + NKP\log P))$
SBCSC [19]	spatial	Gaussian	BCD, updating dictionary by ADMM and codes by LARS[28]	$O(NKP)$	$O(T^{\text{alt}}I \cdot (NK^3P + NKPM + K^2M))$
αCSC [12]	spatial	alpha-stable	BCD, updating codes and dictionary by L-BFGS [29]	$O(NKP)$	$O(T^{\text{EM}}T^{\text{alt}}I \cdot (NK^2PM^2 + K^3M^3)) + O(\text{MCMC})$
GCSC	frequency	Gaussian mixture	niAPG	$O(NKP)$	$O(T^{\text{EM}}I \cdot (GNP + NKP\log P))$

For the Gaussian pair with the smallest relative difference, if this value is small (less than 0.1), they are merged as

$$\begin{aligned}\pi_a &\leftarrow \pi_a + \pi_b,\\ \mu_a &\leftarrow \frac{\pi_a\mu_a + \pi_b\mu_b}{\pi_a + \pi_b},\\ \Sigma_a &\leftarrow \text{Diag}(\sigma_a^2(1), \ldots, \sigma_a^2(P)),\end{aligned}$$

where $\sigma_a^2(p) = \frac{\pi_a\sigma_a^2(p) + \pi_b\sigma_b^2(p)}{\pi_a(p) + \pi_b(p)}$.

We compare GCSC with the following state-of-the-art CSC methods that are designed for various noise distributions:

1) αCSC, which uses the symmetric alpha-stable distribution $\mathcal{S}(\alpha, \delta, \sigma, \mu)$ with stability α, skewness δ, scale σ, and position μ. In the implementation[3] of [12], $\delta = 0, \sigma = 1/\sqrt{2}$ and $\mu = \sum_{i=1}^N \sum_{p=1}^P \tilde{x}_i(p)$ where $\tilde{x}_i$ is as defined in (1).
2) CSC-ℓ_2, which uses the normal distribution. As shown in Table I, existing CSC methods (except αCSC and GCSC) all assume Gaussian noise. Though they use different optimization solvers and thus have different speeds, they have the same denoising performance as the underlying noise models are the same (this is also empirically validated in Appendix B). Thus, we only compare with the state-of-the-art SBCSC [19].[4] As shown in [7], [19], SBCSC outperforms the other batch CSC algorithm (with the ℓ_2-loss) in terms of both speed and recovery error.
3) CSC-ℓ_1, which uses the Laplace distribution and corresponds to the ℓ_1-loss. We take this as an additional baseline, even though we are not aware of any existing CSC method using the Laplace noise. The Laplace distribution is more robust to outliers [31]. It is formulated as:

$$\min_{\{d_k\}\in\mathcal{D},\{z_{ik}\}} \sum_{i=1}^N \left(\frac{1}{2}\left\|x_i - \sum_{k=1}^K d_k * z_{ik}\right\|_1 - \sum_{k=1}^K \beta\|z_{ik}\|_1\right). \quad (16)$$

The detailed derivations are in Appendix C.

Experiments are performed on a PC with Intel i7 4GHz CPU with 32GB memory.

Hyperparameter Tuning: We create a validation set by randomly sampling 20% of the samples. This is then used to tune the hyperparameters α (used in αCSC) and β (which controls sparsity of the CSC codes).

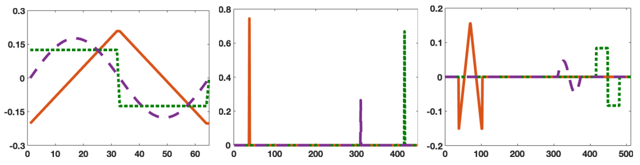
Stopping Criteria: αCSC and GCSC are solved by EM. We stop the EM iterations when the relative change of log posterior in consecutive iterations is smaller than 10^{-4}. In the M-step, we stop the updating of weighted CSC (Algorithm 2 for GCSC, and the algorithm for M-step in Appendix B of [12]) if the relative change of the respective objective value is smaller than 10^{-4}. The optimization problems in CSC-ℓ_1 and CSC-ℓ_2 are solved by BCD. Alternating minimization is stopped when the relative change of objective value ((2) for CSC-ℓ_2 and (16) for CSC-ℓ_1) in consecutive iterations is smaller than 10^{-4}. As for the optimization subproblems of d_k's (given z_{ik}'s) and z_{ik}'s (given d_k's), we stop when the relative change of objective value is smaller than 10^{-4}.

B. Synthetic Data

In this experiment, we first demonstrate the performance on synthetic data. Following [12], we use $K = 3$ filters d_k's (triangle, square, and sine), each of length $M = 65$ (Figure 3(a)). Each d_k is normalized to have zero mean and

[3]https://alphacsc.github.io/

[4]http://vardanp.cswp.cs.technion.ac.il/wp-content/uploads/sites/62/2015/12/SliceBasedCSC.rar



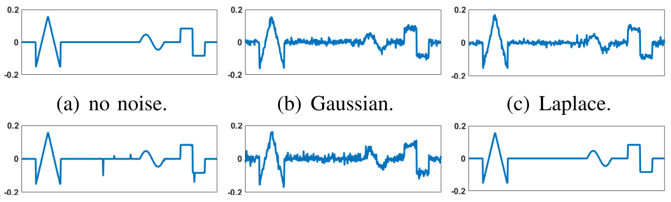
(a) The 3 d_k's. (b) The 3 z_{ik}'s. (c) x_i^{clean}.

Fig. 3. Example of synthesizing a sample.

TABLE II

TYPES OF NOISE ADDED TO SYNTHETIC DATA

noise	distribution	SNR (dB)
Gaussian	$\mathcal{N}(0, 0.01^2)$	13.04
Laplace	$\mathcal{L}(0, 0.01)$	13.03
alpha-stable	$\mathcal{S}(1, 0, 0.01^2, 0)$	10.43
zero-mean mixture	20% from $\mathcal{U}(-0.01, 0.01)$, 20% from $\mathcal{N}(0, 0.01^2)$, 60% from $\mathcal{N}(0, 0.015^2)$	10.98
nonzero-mean mixture	20% from $\mathcal{U}(-0.01, 0.01)$, 20% from $\mathcal{N}(0.01, 0.01^2)$, 60% from $\mathcal{N}(-0.005, 0.005^2)$	14.16



(a) no noise. (b) Gaussian. (c) Laplace.

(d) alpha-stable. (e) zero-mean mixture. (f) nonzero-mean mixture.

Fig. 4. Example noisy x_i's constructed by adding noises listed in Table II on x_i^{clean}'s.

unit variance. Each z_{ik} has only one nonzero entry, whose magnitude is uniformly drawn from [0, 1] (Figure 3(b)). $N = 100$ clean samples, each of length $P = 512$, are generated as: $x_i^{\text{clean}} = \sum_{k=1}^{K} d_k * z_{ik}$ (Figure 3(c)).

Following [30], different types of noise (Table II) are added to x_i^{clean} to generate noisy observations x_i's. To compare with αCSC, we additionally consider the symmetric alpha-stable noise from the Cauchy distribution, which is a symmetric alpha-stable distribution. Results are averaged over five runs with different random seeds.

Following [30], performance is evaluated on the denoised reconstructions $\tilde{x}_i = \sum_{k=1}^{K} d_k * z_{ik}$, using d_k's and z_{ik}'s learned from the noisy observation x_i's. The performance evaluation criteria are (i) mean absolute error (MAE): $\frac{1}{NP} \sum_{i=1}^{N} \|x_i^{\text{clean}} - \tilde{x}_i\|_1$); (ii) root mean squared error (RMSE): $\sqrt{\frac{1}{NP} \sum_{i=1}^{N} \|x_i^{\text{clean}} - \tilde{x}_i\|_2^2}$); and (iii) total training time.

Results are shown in Table III. As can be seen, when there is no noise, all methods obtain comparable MAE and RMSE. When there is noise, as expected, the model whose underlying noise assumption matches the actual noise distribution performs the best. Hence, CSC-ℓ_2 performs well on Gaussian noise, CSC-ℓ_1 performs well on Laplacian noise, and αCSC performs well on alpha-stable noise. However, these CSC methods do not perform well on mixture noise. On the other hand, GCSC is the best or comparable with the best method on all types of noise.

TABLE III

PERFORMANCE ON THE SYNTHETIC DATA. THE BEST AND COMPARABLE RESULTS (ACCORDING TO THE PAIRWISE t-TEST WITH 95% CONFIDENCE) ARE HIGHLIGHTED IN BOLD

	MAE	RMSE	time (sec)
	no noise		
CSC-ℓ_2	**0.000359±0.000027**	**0.000847±0.000109**	**417.9±41.8**
CSC-ℓ_1	**0.000364±0.000171**	**0.000871±0.000183**	651.7±98.9
αCSC	**0.000362±0.000109**	**0.000868±0.000122**	2217.4±346.0
GCSC	**0.000330±0.000125**	**0.000849±0.000141**	**414.3±34.5**
	Gaussian noise		
CSC-ℓ_2	**0.00368±0.00036**	**0.00775±0.00072**	**240.7±46.9**
CSC-ℓ_1	0.0104±0.0012	0.0249±0.0008	715.4±93.9
αCSC	**0.00353±0.00017**	**0.00766±0.00052**	1986.1±262.5
GCSC	**0.00343±0.00021**	**0.00762±0.00026**	**238.5±19.8**
	Laplace noise		
CSC-ℓ_2	0.00835±0.00042	0.0114±0.0010	391.4± 57.3
CSC-ℓ_1	**0.00347±0.00026**	**0.00735±0.00038**	**358.7±175.4**
αCSC	0.00692±0.00042	0.00973±0.00013	2309.9±724.5
GCSC	**0.00335±0.00017**	**0.00732±0.00020**	**350.8±68.3**
	alpha-stable noise		
CSC-ℓ_2	0.0702±0.0060	0.0840±0.0091	498.4±69.2
CSC-ℓ_1	0.0160±0.0025	0.0337±0.0047	476.2±37.9
αCSC	**0.00416±0.00039**	**0.00821±0.00024**	2198.3±470.6
GCSC	**0.00402±0.00030**	**0.00815±0.00041**	**412.4±71.2**
	zero-mean mixture noise		
CSC-ℓ_2	0.0321±0.0007	0.0545±0.0010	**337.1±70.3**
CSC-ℓ_1	0.0604±0.0055	0.0849±0.0059	588.6±88.9
αCSC	0.0114±0.0002	0.0158±0.0003	1120.7± 463.7
GCSC	**0.00531±0.00021**	**0.00971±0.00082**	**336.0±77.9**
	nonzero-mean mixture noise		
CSC-ℓ_2	0.0732±0.0015	0.0151±0.0011	558.3±81.3
CSC-ℓ_1	0.0670±0.0037	0.0130±0.0013	788.6±88.9
αCSC	0.0667±0.0002	0.0127±0.0014	2882.3±907.3
GCSC	**0.00556±0.00024**	**0.00818±0.00037**	**471.4±87.9**

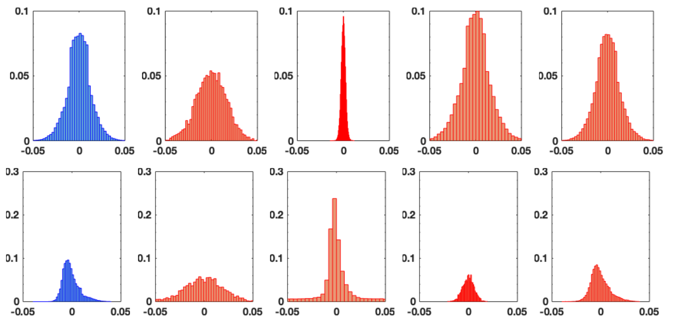
As for speed, GCSC is always the fastest. αCSC is the slowest, as it performs convolution in the spatial domain, requires expensive Markov chain Monte Carlo (MCMC) in its E-step, and also needs an extra alternating loop for the dictionary and code update subproblems. CSC-ℓ_2 and CSC-ℓ_1 can be as fast as GCSC, but only when the actual noise distribution matches their underlying noise models.

Figure 5 compares the ground truth noise with those fitted by the models. Results for other noise models are in Appendix D-A. As can be seen, GCSC fits the noise well in all cases, while the other methods cannot fit the noise well due to a mismatch between their underlying noise distributions and the actual noise. Figure 6 shows the learned filters. It can be seen that GCSC recovers the underlying filters more reliably. Figure 7 shows the reconstructions on synthetic data with nonzero-mean noise. GCSC is the only one that denoises well and recovers the underlying clean data.

C. Hyperspectral Image Data

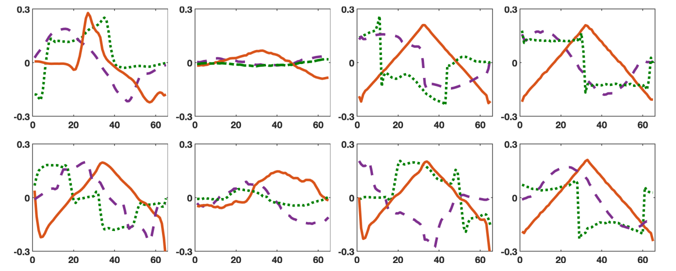
In this section, we perform experiments on the hyperspectral image data set[5] *Urban*. It contains a 210-band hyperspectral image of size 307 × 307. Pixel values are scaled to [0,1]. 48 of the bands are known to be noisy, with obvious noisy patterns such as stripes. To denoise a certain noisy band (e.g., band 103), we randomly sample 4 clean bands, and then denoise these five bands together. The clean bands can help extract local spatial patterns that are useful for denoising the

[5]http://acwc.sdp.sirsi.net/client/en_US/search/asset/1045185



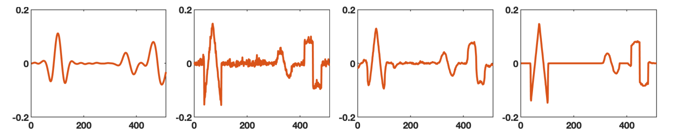
(a) Ground truth. (b) CSC-ℓ_2. (c) CSC-ℓ_1. (d) αCSC. (e) GCSC.

Fig. 5. Noise histograms fitted by different CSC algorithms on synthetic data, corrupted with *zero-mean mixture* noise (top), and *nonzero-mean mixture* noise (bottom).



(a) CSC-ℓ_2. (b) CSC-ℓ_1. (c) αCSC. (d) GCSC.

Fig. 6. Filters obtained by different CSC algorithms on synthetic data corrupted with *zero-mean mixture* noise (top) and *nonzero-mean mixture* noise (bottom).



(a) CSC-ℓ_2. (b) CSC-ℓ_1. (c) αCSC. (d) GCSC.

Fig. 7. Reconstruction results on synthetic data with nonzero-mean mixture noise.

TABLE IV

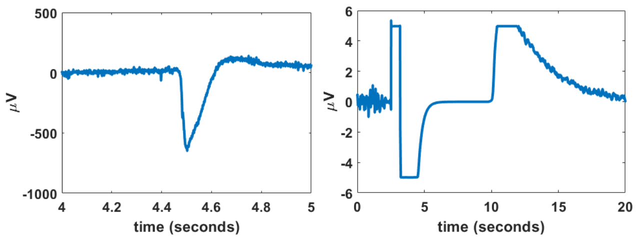
PERFORMANCE ON THE NOISY BANDS OF *Urban*

	CSC-ℓ_2	CSC-ℓ_1	αCSC	GCSC
PSNR (dB)	22.87±0.13	24.71±0.12	24.32±0.17	**26.01±0.11**
time (sec)	723.3±69.7	801.1±91.2	2627.8±258.7	**705.4±57.6**

noisy band. The proposed GCSC is compared with CSC-ℓ_2, CSC-ℓ_1 and αCSC (all with $K = 50$ and $M = 11 \times 11$).

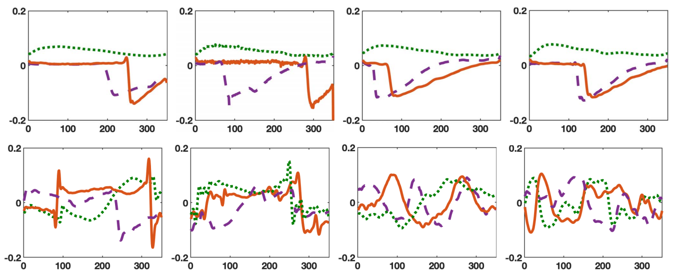
As the ground truth is not available, we compare the reconstructions with those obtained by NGmeet [32], the state-of-the-art in hyperspectral image denoising. For band b, let x_b^{NGmeet} be NGmeet's reconstruction, and $\tilde{x}_b = \sum_{k=1}^{K} d_k * z_{bk}$ be the CSC reconstruction based on the obtained d_k's and z_{bk}'s. We define the reconstructed peak signal-to-noise ratio (PSNR) on the noisy bands as: PSNR $= \frac{1}{|\Omega|}\sum_{b\in\Omega} 20\log_{10}\left(\frac{\sqrt{P}}{\|x_b^{\text{NGmeet}} - \tilde{x}_b\|_2}\right)$, where Ω contains the 48 noisy bands. The performance is obtained by averaging over five repetitions on the choice of the clean bands.

Table IV shows the PSNR and clock time. As can be seen, GCSC is the best, which is then followed by CSC-ℓ_1, αCSC and CSC-ℓ_2. Some example reconstruction results are shown in Appendix D-B.



(a) *Cortical*. (b) *Striatal*.

Fig. 8. Example segments from LFP data sets.



(a) CSC-ℓ_2. (b) CSC-ℓ_1. (c) αCSC. (d) GCSC.

Fig. 9. Filters obtained by different CSC algorithms on *Cortical* (top) and *Striatal* (bottom).

TABLE V

TIMING RESULTS (SECONDS) ON THE *LFP* DATA

	CSC-ℓ_2	CSC-ℓ_1	αCSC	GCSC
Cortical	721.7±47.2	737.6±68.9	2919.3±290.7	**607.2±57.6**
Striatal	729.4±62.4	783.2±76.1	3004.9±320.8	**611.2±69.9**

D. Local Field Potential Data

In this section, experiments are performed on two local field potential (LFP) data sets obtained from [12]. LFP is an electrophysiological signal recording the collective activities nearby neurons, which is closely related to cognitive mechanisms such as attention and motor control. The first signal (*Cortical*) is recorded in the rat cortex [20], while the second one (*Striatal*) is recorded in the rat striatum [33]. Following [12], we extract from each data set $N = 100$ non-overlapping segments, each of length $P = 2500$. Figure 8 shows the sample segments. Note that *Striatal* contains heavier artifact. The other preprocessing steps and parameter setting ($K = 3$ and $M = 350$) are the same as in [12]. The experiment is repeated five times with different initializations.

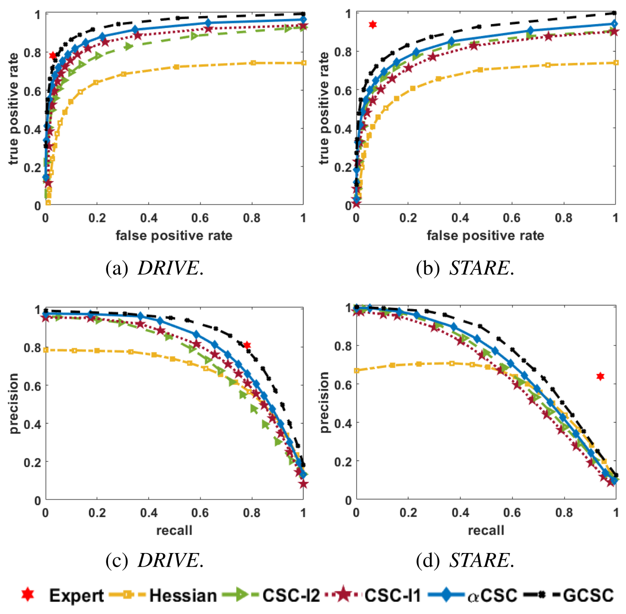
Figure 9 shows the learned filters. As there is no ground truth, we only evaluate the results qualitatively. For *Cortical*, the learned filters are similar to the local regions in segments. As for *Striatal*, severe artifacts contaminate the filters learned by CSC-ℓ_1 and CSC-ℓ_2, but the filters learned by GCSC and αCSC are similar in shape to the clean part of the segments. Table V compares the time. As can be seen, GCSC is the fastest on both *Cortical* and *Striatal*.

E. Retinal Image Data

Finally, we perform vessel segmentation via pixelwise classification of retinal image data sets. A pixel from the retinal vessel is labeled 1, while a pixel from the background is labeled 0. We use the two popular data sets from the authors of [34]: *DRIVE*[6] [35] and *STARE*[7] [36]. *DRIVE* contains

[6]http://www.isi.uu.nl/Research/Databases/DRIVE/

[7]http://cecas.clemson.edu/~ahoover/stare/



(a) *DRIVE*. (b) *STARE*.

(c) *DRIVE*. (d) *STARE*.

Fig. 10. ROC (top) and PR (bottom) curves on the retinal image data sets.

TABLE VI

PERFORMANCE ON THE RETINAL IMAGE DATA SETS

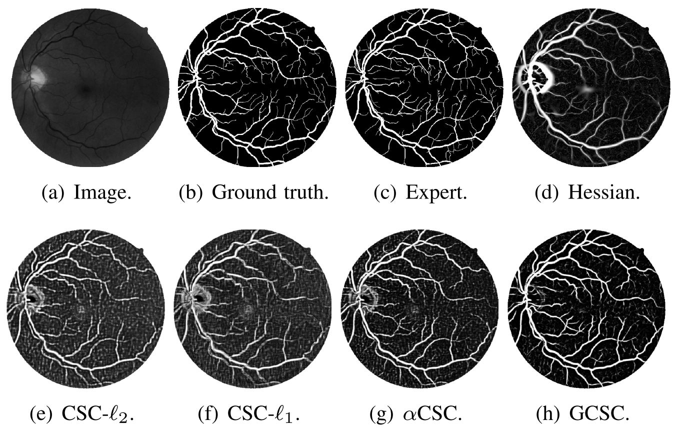
		AUC	best F-score
DRIVE	Expert	-	0.8935 ± 0.0000
	Hessian	0.7314 ± 0.0000	0.8755 ± 0.0000
	CSC-ℓ_2	0.9044 ± 0.0067	0.9806 ± 0.0065
	CSC-ℓ_1	0.9383 ± 0.0071	0.9821 ± 0.0063
	αCSC	0.9401 ± 0.0051	0.9850 ± 0.0064
	GCSC	**0.9504± 0.0048**	**0.9969 ± 0.0066**
STARE	Expert	-	0.7790 ± 0.0000
	Hessian	0.6623 ± 0.0000	0.8495 ± 0.0000
	CSC-ℓ_2	0.9033 ± 0.0089	0.9838 ± 0.0080
	CSC-ℓ_1	0.8964 ± 0.0087	0.9757± 0.0073
	αCSC	0.9101 ± 0.0056	0.9907 ± 0.0062
	GCSC	**0.9203± 0.0066**	**0.9999± 0.0065**

20 training images and 20 test images, each of size 584×565. *STARE* contains 20 images of size 605×700. As in [34]. we use the first 10 images for training, and the rest for testing. Both data sets are provided with manual segmentation results from two experts. Following [10], [34], we use the first expert's segmentation as ground truth.

The proposed GCSC is compared with CSC-ℓ_2, CSC-ℓ_1 and αCSC (all with $K = 50$ and $M = 11 \times 11$). The experiment is repeated five times, with different random initializations of the dictionary and GMM parameters. Following [34], each pixel, represented by the K learned codes, is classified using gradient boosting [37] with 500 weak learners. From each image, 15,000 vessel pixels are sampled as positive, and 15,000 background pixels are sampled as negative. We also compare with the provided second expert's manual segmentation (denoted "Expert") and the state-of-the-art hand-crafted multi-scale Hessian filter[8] (denoted "Hessian") [38] as baselines.

Figure 10 shows the Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curves. Table VI shows the corresponding Area Under ROC Curve (AUC) and the best F-score. As shown, GCSC outperforms all the other methods. αCSC performs slightly better than CSC-ℓ_1 and CSC-ℓ_2. The multi-scale Hessian filter is much worse. The classification performance of Expert is low, which is also noted in [10].

[8]https://www.mathworks.com/matlabcentral/fileexchange/63171-jerman-enhancement-filter



(a) Image. (b) Ground truth. (c) Expert. (d) Hessian.

(e) CSC-ℓ_2. (f) CSC-ℓ_1. (g) αCSC. (h) GCSC.

Fig. 11. Vessel segmentation results on a test retinal image from *STARE*.

Figures 11 shows the segmentation results for a test image from *STARE*. Results on *DRIVE* are in Appendix D-C. As can be seen, the segmentation results produced by αCSC, CSC-ℓ_2 and CSC-ℓ_1 are still noisy. The Hessian filter shows clearer vessels, but enlarges the pupil and shrinks some tiny vessels. In contrast, GCSC obtains cleaner segmented vessels.

V. CONCLUSION

In this paper, we proposed the generalized CSC (GCSC) algorithm to handle various kinds of noise. The noise is modeled with the Gaussian mixture model, and learned by using the EM algorithm. To solve the resultant difficult optimization problem in the M-step, we designed a new solver based on a recent accelerated proximal algorithm. Extensive experiments on both synthetic and real-world data sets validate the efficacy and efficiency of the proposed method on obtaining high-quality filters and representations from data with complicated unknown noises.

The proposed GCSC algorithm is learned in the batch mode. Though it is more efficient than state-of-the-art batch CSC algorithms, it may not be efficient on large data sets. To handle large-scale data, one future direction is to optimize GCSC by online or stochastic learning. Recently, some online CSC methods have been proposed along this direction [7], [39]. However, they are all designed for the ℓ_2-loss. In the future, we will extend these to the GMM noise proposed in this paper. Moreover, it will also be interesting to use multiscale filters to capture local patterns of different sizes.

REFERENCES

[1] 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.

[2] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 801–808.

[3] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 2272–2279.

[4] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1794–1801.

[5] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2528–2535.

[6] Y. Zhu and S. Lucey, "Convolutional sparse coding for trajectory reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 529–540, Mar. 2015.

[7] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Online convolutional sparse coding with sample-dependent dictionary," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5209–5218.

[8] A. Cogliati, Z. Duan, and B. Wohlberg, "Context-dependent piano music transcription with convolutional sparse coding," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 12, pp. 2218–2230, Dec. 2016.

[9] F. Heide, W. Heidrich, and G. Wetzstein, "Fast and flexible convolutional sparse coding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5135–5143.

[10] A. Sironi, B. Tekin, R. Rigamonti, V. Lepetit, and P. Fua, "Learning separable filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 1, pp. 94–106, Jan. 2015.

[11] H. Chang, J. Han, C. Zhong, A. M. Snijders, and J.-H. Mao, "Unsupervised transfer learning via multi-scale convolutional sparse coding for biomedical applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1182–1194, May 2018.

[12] M. Jas, T. La Tour, U. Simsekli, and A. Gramfort, "Learning the morphology of brain signals using alpha-stable convolutional sparse coding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1099–1108.

[13] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *J. Optim. Theory Appl.*, vol. 109, no. 3, pp. 475–494, Jun. 2001.

[14] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun, "Learning convolutional feature hierarchies for visual recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1090–1098.

[15] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.

[16] H. Bristow, A. Eriksson, and S. Lucey, "Fast convolutional sparse coding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 391–398.

[17] B. Wohlberg, "Efficient algorithms for convolutional sparse representations," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 301–315, Jan. 2016.

[18] M. Šorel and F. Šroubek, "Fast convolutional sparse coding using matrix inversion lemma," *Digit. Signal Process.*, vol. 55, pp. 44–51, Aug. 2016.

[19] V. Papyan, Y. Romano, M. Elad, and J. Sulam, "Convolutional dictionary learning via local processing," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5296–5304.

[20] S. Hitziger, M. Clerc, S. Saillet, C. Benar, and T. Papadopoulo, "Adaptive waveform learning: A framework for modeling variability in neurophysiological signals," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4324–4338, Aug. 2017.

[21] B. Mandelbrot, "The Pareto–Lévy law and the distribution of income," *Int. Econ. Rev.*, vol. 1, no. 2, pp. 79–106, May 1960.

[22] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice*. Boca Raton, FL, USA: CRC Press, 1995.

[23] G. J. McLachlan and K. E. Basford, *Mixture Models: Inference and Applications to Clustering*. New York, NY, USA: Marcel Dekker, 1988.

[24] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc. B, Methodol.*, vol. 39, pp. 1–22, Sep. 1977.

[25] Q. Yao, J. T. Kwok, F. Gao, W. Chen, and T.-Y. Liu, "Efficient inexact proximal gradient algorithm for nonconvex problems," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3308–3314.

[26] S. Mallat, *A Wavelet Tour of Signal Processing*. New York, NY, USA: Academic, 1999.

[27] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.

[28] B. Efron *et al.*, "Least angle regression," *Ann. Stat.*, vol. 32, no. 2, pp. 407–499, 2004.

[29] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM J. Sci. Comput.*, vol. 16, no. 5, pp. 1190–1208, Sep. 1995.

[30] D. Meng and F. De la Torre, "Robust matrix factorization with unknown noise," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1337–1344.

[31] Q. Ke and T. Kanade, "Robust ℓ_1 norm factorization in the presence of outliers and missing databy alternative convex programming," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 739–746.

[32] W. He, Q. Yao, C. Li, N. Yokoya, and Q. Zhao, "Non-local meets global: An integrated paradigm for hyperspectral denoising," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 6868–6877.

[33] G. Dallérac *et al.*, "Updating temporal expectancy of an aversive event engages striatal plasticity under amygdala control," *Nature Commun.*, vol. 8, no. 1, Apr. 2017, Art. no. 13920.

[34] C. Becker, R. Rigamonti, P. Lepetit, and V. Fua, "Supervised feature learning for curvilinear structure segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent*, 2013, pp. 526–533.

[35] J. Staal, M. D. Abramoff, M. Niemeijer, M. A. Viergever, and B. van Ginneken, "Ridge-based vessel segmentation in color images of the retina," *IEEE Trans. Med. Imag.*, vol. 23, no. 4, pp. 501–509, Apr. 2004.

[36] A. D. Hoover, V. Kouznetsova, and M. Goldbaum, "Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response," *IEEE Trans. Med. Imag.*, vol. 19, no. 3, pp. 203–210, Mar. 2000.

[37] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[38] T. Jerman, F. Pernus, B. Likar, and Z. Spiclin, "Enhancement of vascular structures in 3D and 2D angiographic images," *IEEE Trans. Med. Imag.*, vol. 35, no. 9, pp. 2107–2118, Sep. 2016.

[39] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Scalable online convolutional sparse coding," *IEEE Trans. Image Process.*, vol. 27, no. 10, pp. 4850–4859, Oct. 2018.

[40] J. W. Cooley, P. A. W. Lewis, and P. D. Welch, "The fast Fourier transform and its applications," *IEEE Trans. Educ.*, vol. E-12, no. 1, pp. 27–34, Mar. 1969.

Yaqing Wang (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, in 2019. She is currently a Researcher with Baidu Research, China. Her research interest is on machine learning algorithms and its applications. She received the Outstanding Undergraduate Award of the China Computer Federation and the Google Excellence Scholarship in 2013. She was also a recipient of the Hong Kong Ph.D. Fellowship.

James T. Kwok (Fellow, IEEE) received the Ph.D. degree in computer science from The Hong Kong University of Science and Technology in 1996. He was an Assistant Professor with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. He is currently a Professor with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology. His research interests include machine learning, artificial neural networks, and artificial intelligence. He received the IEEE Outstanding 2004 Paper Award and the Second Class Award in Natural Sciences by the Ministry of Education, China, in 2008. He has been the program chair and the area chair of a number of international conferences. He has served as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2006 to 2012. He is currently an Associate Editor of the *Neurocomputing* journal and the *International Journal of Data Science and Analytics*.

Lionel M. Ni (Life Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from Purdue University in 1980. He is currently a Vice Rector (Academic Affairs) with The Hong Kong University of Science and Technology. He is a fellow of the Hong Kong Academy of Engineering Science. He has chaired over 30 professional conferences and has received eight awards for authoring outstanding articles. He has been serving on the Editorial Board of the *Communications of the ACM* and the IEEE TRANSACTIONS ON BIG DATA.