Compressive learning for patch-based image denoising

Hui Shi, Yann Traonmilin, Jean-François Aujol

Compressive learning for patch-based image denoising*

Hui Shi†, Yann Traonmilin†, and Jean-François Aujol†

Abstract. The Expected Patch Log-Likelihood algorithm (EPLL) and its extensions have shown good performances for image denoising. The prior model used by EPLL is usually a Gaussian Mixture Model (GMM) estimated from a database of image patches. Classical mixture model estimation methods face computational issues as the high dimensionality of the problem requires training on large datasets. In this work, we adapt a compressive statistical learning framework to carry out the GMM estimation. With this method, called *sketching*, we estimate models from a compressive representation (the *sketch*) of the training patches. The cost of estimating the prior from the sketch no longer depends on the number of items in the original large database. To accelerate further the estimation, we add another dimension reduction technique (low-rank modeling of the covariance matrices) to the compressing learning framework. To demonstrate the advantages of our method, we test it on real large-scale data. We show that we can produce denoising performances similar to performances obtained with models estimated from the original training database using GMM priors learned from the sketch with improved execution times.

Key words. Image denoising, Compressive learning, Sketching, Optimization,

AMS subject classifications. 68U10, 94A08, 49N30

1. Introduction.

We consider the classical noisy observation model of a clean natural image $u \in \mathbb{R}^N$ (composed of N pixels):

$$v = u + w \tag{1.1}$$

where v is the observed degraded version of u. The acquisition noise w is usually assumed to be an additive white Gaussian noise of variance σ, i.e. $w \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2 I_N)$. In the last two decades, non local patch-based methods have been proven successfull for denoising. Methods such as Piecewise Linear Estimators [61, 1], BM3D [10, 29] or NL-Bayes [31, 30, 59, 28] are examples of non-local methods [3]. In patch-based image denoising, the noisy image v is divided into small patches $\{v_i\}_{i=1}^M$. Each patch $v_i \in \mathbb{R}^P$ (P is the patch size) can be seen as a vector in a high dimensional space. The denoising problem is considered on each patch:

$$v_i = u_i + w_i, \tag{1.2}$$

and a corresponding denoised version u_i^* of the true values u_i are estimated. To overcome the ill-posedness of this inverse problem, various denoising methods [32, 31, 30, 22] consider patch models within a Bayesian framework. According to the Bayes' theorem, the objective is to find u_i^* which maximizes the posterior probability distribution $f(u_i|v_i)$ under the prior

*Submitted to the editors DATE.

Funding: This work was funded by the French National Research Agency (ANR) under reference ANR-20-CE40-0001 (EFFIREG project).

†Univ. Bordeaux, Bordeaux INP, CNRS, IMB, UMR 5251, F-33400 TALENCE, FRANCE. ({hui.shi, yann.traonmilin, jean-francois.aujol}@math.u-bordeaux.fr).

$p(u_i)$. The Maximum A Posteriori (MAP) problem is formulated as

$$u_i^* = \arg\max_{u_i \in \mathbb{R}^P} f(u_i|v_i) = \arg\max_{u_i \in \mathbb{R}^P} f(v_i|u_i)p(u_i) \propto \arg\max_{u_i \in \mathbb{R}^P} e^{-\frac{\|u_i - v_i\|^2}{2\sigma^2}} p(u_i) \tag{1.3}$$

where $\|\cdot\|$ denotes the ℓ^2-norm. This yields

$$u_i^* = \arg\min_{u_i \in \mathbb{R}^P} \frac{\|u_i - v_i\|^2}{2\sigma^2} - log(p(u_i)). \tag{1.4}$$

Ideally, the choice of the prior distribution should be determined by the nature of the image to be estimated. In practice, Gaussian Mixture Models (GMM) [61, 57, 22] have shown their effectiveness. With the GMM prior, the solution of problem (1.4) can be approximated by a Wiener filter solution (see subsection 2.2).

Among these various non-local denoising methods, the Expected Patch Log-Likelihood algorithm (EPLL) [64] occupies a central position due to its efficient denoising performance. A large number of works build on the original EPLL formulation to deal with more general prior or go beyond the denoising problem [12, 37, 6, 33, 41, 52, 11, 44]. EPLL uses a GMM prior learned from a very large set of patches extracted from clean images. The key to the success of EPLL is to find a good prior distribution. Since in practice patch sizes are typically greater than 5×5, estimating prior distributions in such a high-dimensional space is a difficult task. Moreover, to estimate the best possible model, we need to maximize the redundancy of structural information and use training databases as large as possible. As the traditional empirical minimization approaches require access to the whole training dataset, when the collection size is large, the learning process can be extremely costly. For instance, in the case of the classical learning method Expectation Maximization (EM), the memory consumption and computation time depend on the size of the database (see section 3).

Leveraging ideas from compressive sensing [15] and streaming algorithms [9], R. Gribonval et al. propose a *sketching* method [25, 19, 20, 18, 17] to compress the training database. This scalable technique compress the whole training collection into a fixed-size representation (a vector): a *sketch* of the training dataset before learning. The sketch captures the necessary information for the considered learning task. For certain mixture model estimation, it is then possible to learn their parameters directly from the sketch, without access to the original dataset. Hence the space and time complexity of the learning algorithm no longer depends on the original database size, but only on the size of the sketch which is linked to the dimensionality of the model. Sketching has been already used successfully in machine learning [45, 17, 27, 7, 5, 40], generative networks [46], source localization [13, 14], independent component analysis [48] and depth imaging [49]. In [25], the sketching is implemented and evaluated on synthetic data to estimate a GMM with diagonal covariances. It is shown that on large synthetic data, for the estimation of GMM, the sketching produces precise results while requiring fewer memory space and computations. In this work, we explore the sketching method in the image patches context where GMM with full covariance must be estimated from the compressed database.

Due to the curse of dimensionality, it is computationally expensive to manipulate the GMMs' covariance matrices. In [42], the authors show that most natural images and videos

can be represented by a GMM with low-rank covariance matrices. The experiments have also shown the efficiency of low-rank covariance matrices applied to image denoising [38], image inpainting, high-speed video and hyperspectral imaging [60]. This motivates us to use such low-rank covariances in the GMM modeling of patches and extend the sketching framework accordingly to gain computational speedup and to manage the modeling of the image patches in the most possible flexible way.

1.1. Contributions. A preliminary and short version of this work has appeared in [51]. In this paper, we provide a more detailed version of this work with a final consolidated version of the proposed learning algorithm, validated by extended numerical experiments.

Figure 1 summarizes the principle of our approach. We first construct a sketch by averaging random Fourier features computed over the whole image patch database. Then the model parameters are learned directly from the sketch by our Low-rank Continuous Orthogonal Matching Pursuit (LR-COMP) algorithm without access to the original database. Finally, the learned model is used with a Bayesian method (EPLL) for the denoising task. Our

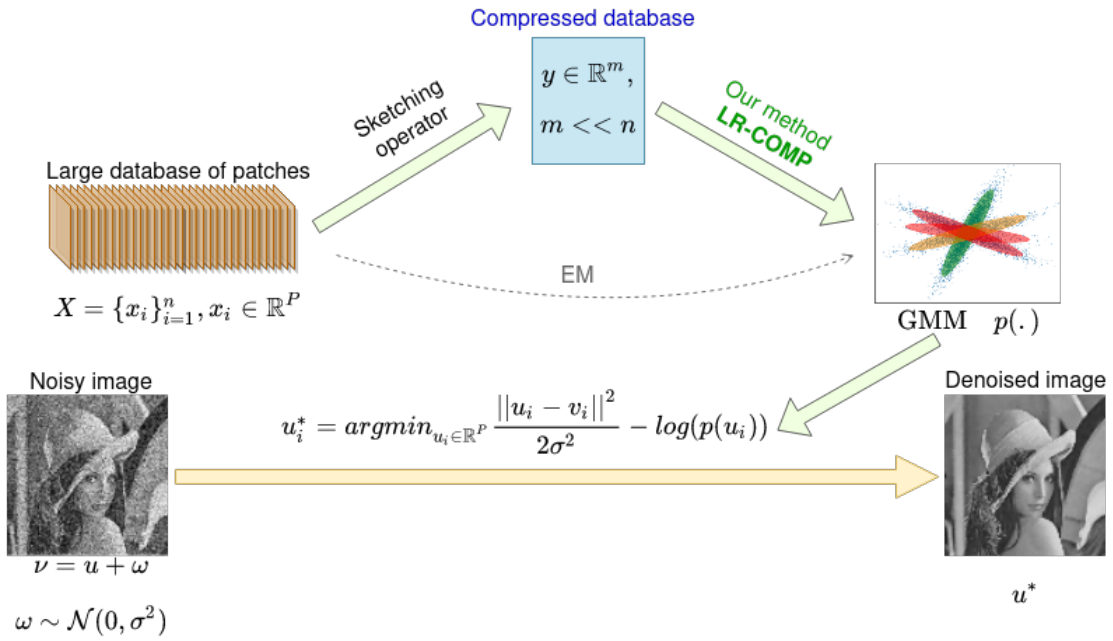

Figure 1. *A summary of our method*

contributions of this piece of work are the following:

- In this work, we propose an algorithm LR-COMP to estimate a GMM with non-diagonal and low-rank covariance matrices. Compared to previous work in [25], our extension to non-diagonal covariance matrices allows us to learn a GMM prior from a compressed database of patches in the context of image denoising. Moreover, with the low-rank approximation of the covariance matrices, we lighten the computation burden in the denoising process while keeping good denoising performances.

- We demonstrate the performance of our approach on real large-scale data (over 4 millions training samples of patch size of 7×7) for the task of patch-based image denoising. We show that using models trained with the compressed database, we can obtain similar denoising performances compared to the models obtained with the classical EM algorithm. To the best of our knowledge, this is also the first time that the sketching framework has been applied with such high dimensional GMMs.
- Computationally, we estimate the model from a compressed database which is about 1000 times smaller than the original patch database. It leads to running time approximately two times faster compared to the EM method.

The paper is organized as follows. Section 2 is a reminder of the EPLL framework. Then we review the EM algorithm in section 3. In section 4, we explain the compressive learning method. In section 5, we focus on explaining how to adapt the sketching framework to learn a GMM in the image patch context. We also interpret the extension to low-rank covariances and the implementation details of the adapted learning algorithm LR-COMP. In section 6, we provide numerical experiments that demonstrate the performance of our approach. Some conclusions and tracks for further works follow in section 7.

2. Image denoising with EPLL.

We review in this section the Expected Patch Log-Likelihood (EPLL) framework for image denoising. EPLL is a patch-based image restoration algorithm introduced by Zoran and Weiss [64]. The EPLL framework restores an image u by performing the following maximum a posteriori (MAP) estimation over all N patches:

$$u^* = \arg\min_{u \in \mathbb{R}^N} \frac{P}{2\sigma^2}\|u - v\|^2 - \sum_{i=1}^{N} log(p(\mathcal{P}_i u)) \tag{2.1}$$

where $\mathcal{P}_i : \mathbb{R}^N \to \mathbb{R}^P$ is a linear operator that extracts a patch of P pixels centered at the position i, typically $P = 7 \times 7$. The function $p(\cdot)$ is the density of the prior probability distribution of the patches. Note that in practice, we assume that patches are distributed independently.

2.1. Optimization.

Due to the non-convexity of $p(\cdot)$, direct optimization of the problem may be difficult. The authors of EPLL propose to perform the optimization with "half-quadratic splitting" [16]. By introducing N auxiliary unknown vectors $z_i \in \mathbb{R}^P$ and a denoising parameter $\beta > 0$, the problem is then considered as:

$$u^* = \arg\min_{\substack{u \in R^N \\ z_1,\dots,z_N \in R^P}} \frac{P}{2\sigma^2}\|u - v\|^2 + \frac{\beta}{2}\sum_{i=1}^{N}\|\mathcal{P}_i u - z_i\|^2 - \sum_{i=1}^{N} log(p(z_i)). \tag{2.2}$$

The optimization (2.2) is accomplished by alternating the minimization of u and z_i.

- **Solving u for fixed** z_i — Problem (2.2) turns into a linear inverse problem with the

Tikhonov regularization. It has a closed form solution:

$$\begin{aligned}\hat{u} &= \arg\min_{u\in\mathbb{R}^N} \frac{P}{2\sigma^2}\|u-v\|^2 + \frac{\beta}{2}\sum_{i=1}^{N}\|\mathcal{P}_i u - z_i\|^2 \\ &= (I + \frac{\beta\sigma^2}{P}\sum_{i=1}^{N}\mathcal{P}_i^T\mathcal{P}_i)^{-1}(v + \frac{\beta\sigma^2}{P}\sum_{i=1}^{N}\mathcal{P}_i^T z_i)\end{aligned} \tag{2.3}$$

with $\sum_{i=1}^{N}\mathcal{P}_i^\top\mathcal{P}_i = PI$, where P is the number of patches overlapping each pixel. Hence we have

$$\hat{u} = (I + \sigma^2\beta I)^{-1}(v + \sigma^2\beta\bar{z}_i) \tag{2.4}$$

where $\bar{z}_i := (\sum_{i=1}^{N}\mathcal{P}_i^\top\mathcal{P}_i)^{-1}\sum_{i=1}^{N}\mathcal{P}_i^\top z_i = \frac{1}{P}\sum_{i=1}^{N}\mathcal{P}_i^\top z_i$ is the image after averaging all overlapping patches z_i.

- **Solving z_i for fixed $\mathbf{u}$** — The minimization problem (2.2) is separable with respect to the latent variable z_i. It means that for each z_i we solve a patch MAP estimation under the patch prior $p(z_i)$, i.e. for all i,

$$\hat{z}_i = \arg\min_{z_i\in\mathbb{R}^P} \frac{\beta}{2}\|\mathcal{P}_i\hat{u} - z_i\|^2 - log(p(z_i)). \tag{2.5}$$

The solution of this problem depends on the choice of patch prior $p(\cdot)$.

2.2. Denoising with a GMM prior. EPLL assumes that the prior is a finite Gaussian mixture model (GMM) with zero-mean on centered patches: the empirical mean estimated from noisy patches are removed before the denoising process (2.5) and added back in the end. We consider that a zero-mean patch $x \in \mathbb{R}^P$ is a random vector generated from a distribution with density $p(x)$ defined as

$$p(x) = \sum_{k=1}^{K}\alpha_k\mathcal{N}_P(x;0,\Sigma_k) \tag{2.6}$$

where K is the number of Gaussian components and $\alpha_k \geq 0$ are weights of each component such that $\sum_{k=1}^{K}\alpha_k = 1$. The function $\mathcal{N}_P(x;0,\Sigma_k)$ denotes the density of a Gaussian distribution with zero-mean with covariance $\Sigma_k \in \mathbb{R}^{P\times P}$. Recall that the zero-mean Gaussian distribution density is:

$$\mathcal{N}_P(x;0,\Sigma_k) = \frac{1}{(2\pi)^{P/2}|\Sigma_k|^{1/2}}e^{-\frac{1}{2}x^T\Sigma_k^{-1}x}. \tag{2.7}$$

Hence, under the GMM prior, Problem (2.5) turns to:

$$\hat{z}_i = \arg\min_{z_i\in\mathbb{R}^P} \frac{\beta}{2}\|\mathcal{P}_i\hat{u} - z_i\|^2 - log(\sum_{k=1}^{K}\alpha_k\mathcal{N}_P(z_i;m_i,\Sigma_k)) \tag{2.8}$$

where we supposed that the mean m_i are correctly estimated by the empirical mean of noisy patches.

This problem cannot be solved in closed form as the second term is the logarithm of a sum of exponential. In [64], the authors proposed to solve this problem by keeping only one Gaussian component. For a given centered patch $\tilde{z}_i = \mathcal{P}_i\hat{u} - m_i$, we chose the component k_i^* that maximizes the posterior probability $p(k_i|\tilde{z}_i)$. This leads to computationally efficient implementations. [54] also justified that only one component is required for good reconstructions. The index k_i^* is chosen by

$$\begin{aligned} k_i^* &= \arg\max_{1\leq k_i\leq K} p(k_i|\tilde{z}_i) = \arg\max_{1\leq k_i\leq K} p(k_i)p(\tilde{z}_i|k_i) \\ &= \arg\min_{1\leq k_i\leq K} -2\log\alpha_{k_i} + \log\left|\Sigma_{k_i} + \frac{1}{\beta}I_P\right| + \tilde{z}_i^\top(\Sigma_{k_i} + \frac{1}{\beta}I_P)^{-1}\tilde{z}_i \end{aligned} \tag{2.9}$$

where α_{k_i} and Σ_{k_i} are the weights and the covariance matrices of the kth Gaussian component for the given patch $\tilde{z}_i$. With k_i^* (instead of a sum of K components), the solution of (2.8) is then a Wiener filtering solution:

$$\hat{z}_i = (\Sigma_{k_i^*} + \frac{1}{\beta}I_P)^{-1}\Sigma_{k_i^*}\tilde{z}_i + m_i. \tag{2.10}$$

2.3. Eigenspace implementation of EPLL. The matrix inversions in (2.9) and (2.10) can be done efficiently by using the singular value decomposition over the covariance matrices. We denote $\Sigma_k = U_k\Lambda_k U_k^T$, with $U_k \in \mathbb{R}^{P\times P}$ an unitary matrix and $\Lambda_k = \mathrm{diag}(\lambda_1^{(k)}, ..., \lambda_P^{(k)})$ a diagonal matrix. The diagonal entries $\lambda_i^{(k)}$ of Λ_k are the singular values of Σ_k. Then we can compute (2.9) by:

$$k_i^* = \arg\min_{1\leq k\leq K} -2\log\alpha_k + \sum_{j=1}^{P}\left(\log(\lambda_j^{(k)} + \frac{1}{\beta}) + \frac{[\tilde{v}_i^{(k)}]_j^2}{\lambda_j^k + \frac{1}{\beta}}\right) \tag{2.11}$$

where

$$\tilde{v}_i^{(k)} = U_k^T\tilde{z}_i. \tag{2.12}$$

Then (2.10) leads to

$$\hat{z}_i = U_{k_i^*}S_{k_i^*}U_{k_i^*}^T\tilde{z}_i + m_i = U_{k_i^*}S_{k_i^*}\tilde{v}_i^{(k_i^*)} + m_i \tag{2.13}$$

with

$$S_{k_i^*} = \mathrm{diag}\left(\frac{\lambda_j^{(k_i^*)}}{\lambda_j^{(k_i^*)} + \frac{1}{\beta}}\right)_{j=1,...,P}. \tag{2.14}$$

3. Learning a GMM with EM. The Expectation-Maximization (EM) algorithm is a classical mixture estimation approach. This algorithm starts with some initial estimates of model parameters and then iteratively updates the estimate until the the estimates are not changing much. See Appendix B for the details of the EM algorithm. In each iteration, it carries out two steps: the E-Step (expectation step) and the M-Step (maximization step). In E-Step, using the current estimate of the parameters, we evaluate the posterior probabilities. In the M-Step we compute parameters that maximize the probabilities found on the E-Step. These estimated parameters are then used to determine the distribution of the latent variables in the next E-Step.

As for the time complexity of one iteration of this algorithm, it is linear in the number of model components K and the number of elements in the database n. However it is cubic with respect to the dimensions P due to the fact that we need to inverse the covariance matrix when calculating the density in E-Step. Thus, when estimating a K-components GMM on a database of n elements of dimension P, the computational complexity of one iteration of the EM algorithm is $\mathcal{O}(nKP^2 + KP^3)$. The major criticism of the EM algorithm is that when dealing with a large dataset, it often converges slowly. To address this problem, researchers have developed various variations of the traditional EM algorithm [36, 56]. Learning parameters using EM technique face computational issues linked to the size of the dataset and the number of parameters to estimate, which would make the use of (very) large image patches databases impractical. In the next section we will see an alternative manner to learn parameters using compressive learning.

4. Sketching. Sketching is a dimensionality reduction method. The principle is to compress the whole dataset massively before learning. First, the dataset $\chi = \{x_i\}_{i=1}^n$ is summarized into a vector $y \in \mathbb{C}^m$ ($m \ll n$) called the *sketch*:

$$y := \text{Sketch}(\chi). \tag{4.1}$$

Note that the computation of the sketch can be performed in a distributed manner. Then we apply a learning procedure Υ that allows us to learn an estimate Ψ^* of some statistical parameters Ψ of the dataset directly from the sketch y, namely

$$\Psi^* = \Upsilon(y) = \Upsilon(\text{Sketch}(\chi)) \tag{4.2}$$

More specifically, learning from the sketch corresponds to a minimization problem

$$\Psi^* \in \arg\min_{\Psi} E(y, \Psi) \tag{4.3}$$

where the energy of the model $E(\cdot, \cdot)$ quantifies the fit between the sketch y and the parameter Ψ. In the context of statistical learning, the energy E can be seen as a proxy of the empirical risk. The principle of sketching is summarized in Fig. 2.

4.1. Compressive mixture estimation. In machine learning, the data $x_i \in \mathbb{R}^d$ (e.g. in our case, the patches with $d = P$) are often modeled as i.i.d. random samples generated from a probability distribution parameterized by Θ with a density $f_\Theta \in \mathcal{D}$ ($\mathcal{D}$ is the set of probability measures over $\mathbb{R}^d$). The idea of sketching is to project the measure f_Θ on a low-dimensional

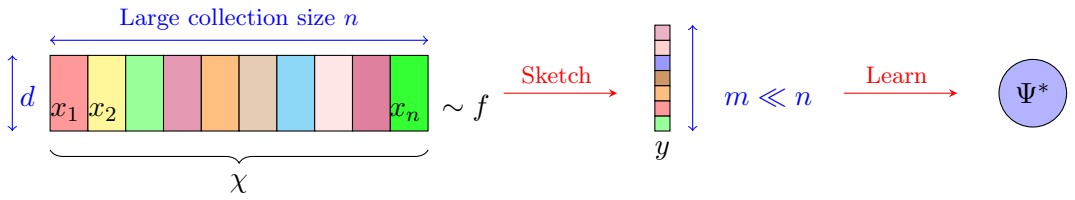

Figure 2. *Schema of sketching*

vector space while keeping all the necessary information of the dataset. Mathematically, given a linear sketching operator $\mathcal{S}$:

$$\begin{aligned} \mathcal{S} : &\mathcal{D} \longrightarrow \mathbb{C}^m \\ &y = \mathcal{S}f \end{aligned} \tag{4.4}$$

and for some finite $K \in \mathbb{N}^*$, we define a K-sparse model $f_{\Theta,\alpha} \in \mathcal{D}$:

$$f_{\Theta,\alpha} = \sum_{k=1}^{K} \alpha_k f_{\theta_k} \tag{4.5}$$

where $f_{\theta_k} \in \mathcal{D}$ are elementary measures parametrized by θ_k, $\alpha_k \geq 0$ for all components and $\sum_{k=1}^{K} \alpha_k = 1$. We can express the vector y as

$$y = \mathcal{S}f_{\Theta,\alpha} = \sum_{k=1}^{K} \alpha_k \mathcal{S} f_{\theta_k}. \tag{4.6}$$

In practice we only have access to the empirical probability distribution $\hat{f} = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}$ where δ_{x_i} is a unit mass at x_i. So we can define the empirical sketch as $\hat{y} = \frac{1}{n}\mathcal{S}\sum_{i=1}^{n} \delta_{x_i}$. The goal of the sketching framework is to recover $f_{\Theta,\alpha}$ from y, hence we do the following minimization to estimate the parameters

$$(\Theta^*, \alpha^*) \in \underset{\substack{\Theta=(\theta_k)_{k=1}^K \\ \alpha\in\mathbb{R}^K, \alpha_k\geq 0, \sum_{k=1}^K \alpha_k = 1}}{\arg\min} \|\mathcal{S}f_{\Theta,\alpha} - \hat{y}\|_2^2. \tag{4.7}$$

The objective of sketched learning algorithms is to minimize a datafit functional between the compressed database and the sketch of the estimation. In other words, our aim is to find parameters α, Θ such that the sketch of the probability distribution parameterized by α, Θ is the closest to the empirical sketch $\hat{y}$.

4.2. Recovery guarantees. It was shown in [18] that we can guarantee theoretically the success of this estimation with a condition on the sketch size. These guarantees necessitate a "Lower Restricted Isometry Property" (LRIP) of the sketching operator. This property, is verified with high probability, for GMM with sufficiently separated mean and random Fourier sketching as long as the sketch size $m \geq O(K^2 d \text{polylog}(K, d))$, i.e. when the size of the sketch

essentially depends on the parameters K (the number of components) and d (the model dimension). Empirical results seem to indicate that for d_{tot} the total number of parameters, a database size of the order of d_{tot} is sufficient: in the case of estimating a GMM with diagonal covariance matrices [25], the authors observe that the quality of the reconstruction exhibits a sharp phase-transition with respect to the sketch size m. This phase transition happens for m proportional to d_{tot}. In our model, $d_{\text{tot}} = K(Pr+1)$. The excess risk of the GMM learning task is then controlled by the sum of an empirical error term and a modeling error term. This guarantees that the estimated GMM approximates well the distribution of the data [19].

Note that EPLL uses a zero-mean GMM as the patch prior, therefore, during the learning process, the patches are centered before sketching and we do not estimate the mean of Gaussians. In our case, the sketched GMM learning problem reduces to the estimation of the sum of k zero-mean Gaussians with covariances $\Theta = (\Sigma_k)_{k=1}^K$, i.e $f_{\Theta,\alpha} = \sum_{k=1}^K \alpha_k g_{\Sigma_k}$ where g_Σ is the zero-mean Gaussian measure with covariance Σ. In this context, the notion of separation used to prove guarantees in [18] does not hold. We still show empirically that the sketching process is successful without this separation assumption.

4.3. Design of sketching operator: randomly sampling the characteristic function. In [25], the sketch is a sampling of the characteristic function (*i.e* the Fourier transform of the probability distribution f). Recall that the characteristic function ψ_f of a measure f is defined as:

$$\psi_f(\omega) = \int_{\mathbb{R}^d} e^{-i\omega^T x} df(x) \quad \forall \omega \in \mathbb{R}^d. \tag{4.8}$$

The sketching operator is therefore expressed as:

$$\mathcal{S}f = [\psi(\omega_1), ..., \psi(\omega_m)]^T \tag{4.9}$$

where $\{\omega_1, ..., \omega_m\}$ is a set of well chosen frequencies. In the spirit of Random Fourier Sampling, the authors of [25] propose to draw the frequencies from a probability distribution, i.e. $(\omega_1, ..., \omega_m) \overset{i.i.d.}{\sim} \Delta$. The choice of frequencies is essential to the success of sketching, and we will discuss it in details in subsection 5.1.

5. Sketching image patches. In this section, we adapt the sketching framework to the context of image patches. As when using the classical EM algorithm, the GMM learning from sketch is performed under the assumption that the training patches are i.i.d. Given a training set of n centered patches $\chi = \{x_1, ..., x_n\} \subset \mathbb{R}^P$, we define the empirical characteristic function with

$$\hat{\psi}(w) = \frac{1}{n} \sum_{j=1}^n e^{-i\omega^T x_j} \quad \text{with} \quad \omega \in \mathbb{R}^P. \tag{5.1}$$

Thus the empirical sketch $\hat{y}$ is expressed as

$$\hat{y} = [\hat{\psi}(\omega_1), ..., \hat{\psi}(\omega_m)]^T = \frac{1}{n} \left[\sum_{j=1}^n e^{-i\omega_1^T x_j}, ..., \sum_{j=1}^n e^{-i\omega_m^T x_j} \right]^T. \tag{5.2}$$

In other words, a sample of the sketched database is a P-dimensional frequency component calculated by averaging over patches (not to be mixed with usual 2D Fourier components of images). Thanks to the properties of the Fourier transform of Gaussians, the sketch of a single zero-mean Gaussian component g_{Σ_k} at frequency ω_l is

$$(\mathcal{S}(g_{\Sigma_k}))_l = \psi_{g_{\Sigma_k}}(\omega_l) = e^{-\frac{1}{2}\omega_l^T \Sigma_k \omega_l}. \tag{5.3}$$

Thus, given the weights $\alpha = (\alpha_k)_{k=1}^K$ and the covariance matrices $\Sigma = (\Sigma_k)_{k=1}^K$, the sketch of a zero-mean GMM $f_{\Sigma,\alpha} = \sum_{k=1}^K \alpha_k g_{\Sigma_k}$ is

$$y = [\mathcal{S}(f_{\Sigma,\alpha})_l]_{l=1,\dots,m} = \left[\sum_{k=1}^K \alpha_k e^{-\frac{1}{2}\omega_l^T \Sigma_k \omega_l}\right]_{l=1,\dots,m}. \tag{5.4}$$

As a consequence, denoting PSD_P the set of $P \times P$ positive symmetric definite matrices, the problem (4.7) of estimating GMM parameters becomes

$$(\Sigma^*, \alpha^*) \in \underset{\substack{\Sigma=(\Sigma_k)_{k=1}^K, \Sigma_k \in PSD_P \\ \alpha \in \mathbb{R}^K, \alpha_k \geq 0, \sum_{k=1}^K \alpha_k = 1}}{\arg\min} \|\hat{y} - \mathcal{S} f_{\Sigma,\alpha}\|_2^2, \tag{5.5}$$

i.e.

$$\left((\Sigma_k^*)_{k=1}^K, (\alpha_k^*)_{k=1}^K\right) \in \underset{\substack{\Sigma_k \in PSD_P, \forall k \\ \alpha_k \geq 0, \sum_{k=1}^K \alpha_k = 1}}{\arg\min} \sum_{l=1}^m \left| \frac{1}{n} \sum_{j=1}^n e^{-i\omega_l^T x_j} - \sum_{k=1}^K \alpha_k e^{-\frac{1}{2}\omega_l^T \Sigma_k \omega_l} \right|^2. \tag{5.6}$$

In practice, the positive definite constraint in the optimization problem is hard to enforce directly on the space of $P \times P$ matrices (as previous work only considered diagonal covariances, it was not an issue). Our method, based on the Burer-Monteiro, permits us to respect the PSD constraint by recasting the covariance estimation problem as an optimization over $\mathbb{R}^{P \times P}$ without constraint (see subsection 5.2 for more details).

5.1. Frequency sampling. The design of the probability distribution Δ for sampling the frequencies $\{\omega_1, ..., \omega_m\}$ is essential to the success of sketching. In our work, we draw frequencies from the *adapted radius* frequency distribution proposed in [25]. The adapted radius heuristic proposes to sample ω as

$$\omega = R\varphi \tag{5.7}$$

where $R \in \mathbb{R}_+$ is the norm of ω and $\varphi \in \mathbb{R}^P$ is the random direction. The radius R is chosen with a radius distribution $R \sim p_R(R;\eta) = ((\eta R)^2 + \frac{1}{4}(\eta R)^4)^{\frac{1}{2}} e^{-\frac{1}{2}(\eta R)^2}$ where η is a scale parameter that should be adjusted to the current dataset to ensure that most of the spectral content of the GMM is sampled. By combining this radius distribution with the decomposition (5.7), we have a frequency distribution referred as *adapted radius* frequency distribution. See Appendix C for details. With this distribution, we avoid sampling very low frequencies. Figure 3 illustrates the curve of $p(R)$ with different values of η.

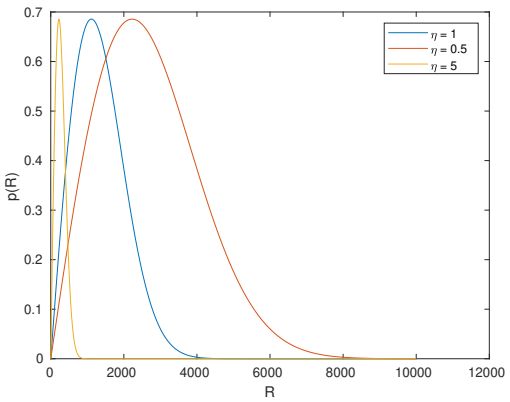

Figure 3. *Curve of the radius distribution density*

5.2. Extension to low-rank covariances. Bayesian MAP theory permits to use a GMM with degenerate covariance matrices as a denoising prior. In this case, the prior is given only in the union of subspaces spanned by the r leading eigenvectors of the K covariance matrices of the GMM. The experiments [38, 42] have shown that we can use low-rank covariance matrices for denoising while keeping good performance. This motivates us to approximate the covariance matrices in the GMM prior by low-rank matrices.

Following classical Burer-Monteiro method [4, 8] in low-rank matrix estimation, we parameterize Σ_k by its factors X_k: $\Sigma_k = X_k X_k^T$. We define $f_{X,\alpha}$ the density function of a zero-mean GMM with $X = (X_k)_{k=1}^K$, where X_k is a factor of a covariance matrix.

Supposing that $\|\hat{y} - \mathcal{S} f_{X,\alpha}\|_2^2$ has a minimizer, we approximate the minimization (5.5) by

$$(\hat{X}, \hat{\alpha}) \in \underset{\substack{X=(X_k)_{k=1}^K, X_k \in \mathbb{R}^{P\times r} \\ \alpha \in \mathbb{R}^K, \alpha_k \geq 0, \sum_{k=1}^K \alpha_k = 1}}{\arg\min} \|\hat{y} - \mathcal{S} f_{X,\alpha}\|_2^2, \tag{5.8}$$

i.e.

$$\left((\hat{X}_k)_{k=1}^K, (\hat{\alpha}_k)_{k=1}^K\right) \in \underset{\substack{X_k \in \mathbb{R}^{P\times r}, \forall k \\ \alpha_k \geq 0, \sum_{k=1}^K \alpha_k = 1}}{\arg\min} \sum_{l=1}^m \left| \hat{y} - \sum_{k=1}^K \alpha_k e^{-\frac{1}{2}\omega_l^T X_k X_k^T \omega_l} \right|^2 \tag{5.9}$$

where $\hat{X} = \{\hat{X}_1, ..., \hat{X}_K\}$ is the collection of factorized rank reduced covariances. With the following proposition, we show that the difference between the costs minimized in (5.5) and (5.9) (the full rank and the low-rank cases, respectively) is associated with the smallest eigenvalues of the covariance matrices. We qualitatively validate this approximation since these eigenvalues are typically small.

Proposition 5.1. *Let* $\Phi^* = \{\Sigma_1^*, ..., \Sigma_K^*, \alpha_1^*, ..., \alpha_K^*\}$ *be a minimizer of* (5.5). *Suppose that there exists a minimizer* $\hat{\Phi} = \{\hat{X}_1, ..., \hat{X}_K, \hat{\alpha}_1, ..., \hat{\alpha}_K\}$ *for the problem* (5.8). *Let* $C =$

$\frac{1}{2}\sqrt{\sum_{l=1}^{m}\|\omega_l\|_2^4}$. *Then we have:*

$$\left\|\mathcal{S}f_{\hat{\Phi}} - \hat{y}\right\|_2 - \|\mathcal{S}f_{\Phi^*} - \hat{y}\|_2 \leq C \max_k \left(\sigma_{r+1}(\Sigma_k^*)\right)$$

where the $\sigma_{r+1}(\Sigma_k^*)$ *is the* $(r+1)$*-th singular value of* Σ_k^* *sorted by decreasing order.*

The proof is detailed in Appendix D.

Ideally, we would like to obtain a similar bound for $\|\Sigma_k^* - \hat{X}_k\hat{X}_k^T\|_F$. We conjecture that a RIP (Restricted Isometry Property) would be needed for such a result. As the verification of RIP remains an open theoretical question in the zero-mean GMM case, we leave this theoretical question for further work.

5.2.1. Related work. There exist other methods to learn GMMs by incorporating a dimensionality reduction technique. In the mixture models of probabilistic PCAs (MPPCA) [53], the covariance matrix Σ_k is parameterized as

$$\Sigma_k = X_k X_k^\top + \gamma_k^2 I. \tag{5.10}$$

The authors use the EM algorithm to optimize the parameters γ_k and X_k. In the high dimensional data clustering (HDDC) model [2], the authors generalized the MPPCA by setting the covariance matrix as

$$\Sigma_k = X_k diag(\lambda_k) X_k^T + \gamma_k^2 I, \quad \lambda_k > 0. \tag{5.11}$$

As for the MPPCA, the parameters are learned by the EM algorithm. In the HDMI model [22], a model selection algorithm for the intrinsic dimension of each mixture component is proposed. In this model, the noise variance γ_k is a priori fixed and the other parameters are optimized by the EM algorithm. Finally, in the PCA-GMM model [21], the covariance matrix Σ_k is expressed as:

$$\Sigma_k = \left(\frac{1}{\gamma_k^2}(I - X_k X_k^T) + X_k \tilde{S}^{-1} X_k^T\right)^{-1}, \quad \tilde{S} \in SPD(P). \tag{5.12}$$

This model is a more general one than HDDC. The parameter γ_k can either be fixed a priori or optimized simultaneously with the other parameters by the EM algorithm. Unlike the above models, our model doesn't estimate γ_k, we rather set a similar user-defined parameter (called μ in our case) at the denoising step. In our model, we also assume that the intrinsic dimension of each Gaussian component is the same and a priori fixed. Using automatically estimated ranks for covariances is a possible future work.

5.3. An algorithm for learning patch prior from a sketch : LR-COMP (Low-Rank Continuous Orthogonal Matching Pursuit). Problem (4.7) can be solved approximately using the greedy Compressive Learning OMP called CL-OMP and a variation of CL-OMP called CL-OMP with Replacement (CL-OMPR) [25, 26]. These algorithms are based on the Matching Pursuit [34], Orthogonal Matching Pursuit [39] and Orthogonal Matching Pursuit with Replacement [23] for classical compressive sensing, which handle sparse approximation problems. It starts from an empty support and it expands the support by greedily adding new atoms to

the current support. Each new atom θ' is found by maximizing the correlation $\langle \mathcal{S}f_{\theta\prime}, r\rangle$ where r is the current residual. Then it updates the weights and reduces the cost function with a descent algorithm initialized with the current parameters. For better feasible recovery, the algorithm using the replacement method was proposed. The approach increases the number of iterations of CL-OMP and extends the size of support more than the desired sparsity. Then it deletes the extra atoms by using the hard thresholding operator. We adapt these algorithms in the GMMs context with our low-rank approximation. Several modifications are detailed below:

- **No Replacement.** Although the algorithms using the replacement method show better results on synthetic data, our results tested on image patches show that the replacement step has a negligible effect. Therefore, we run our algorithm without this hard thresholding operator to decrease the computation time.
- **Estimation of the factors of covariance instead of the covariance matrices.** As we approximate the covariance matrices with their factors, in each step of the algorithm, we do operations directly on the factorized rank reduced covariance X instead of the covariance matrix Σ to lighten the computations.
- **Non-negativity and the normalization of the weights.** In Step 1 of the algorithm, we compute the real part of the correlation between the normalized atom and the residual as done in CL-OMP(R). This avoids a negative correlation and negative weights in practice. No matter how Step 3 was computed, using the projected gradient descent or the gradient descent, or with a direct calculation, there's negligible difference in the result and the running time. The weights are not forced to be sum-to-one at each iteration. However, after transforming the negative weights to zero, an l_1-normalization of the weights is performed at the end of the algorithm.

Algorithm 5.1 LR-COMP: Compressive GMM estimation with low-rank covariances [50].

Input Empirical sketch $\hat{y}$, sketching operator $\mathcal{S}$, sparsity K, rank r
$\hat{r} \leftarrow \hat{y}$; $X \leftarrow \emptyset$
for t $= 1$ **to** K **do**
  Step 1: Perform a descent initialized with a $P \times r$ matrix of normally distributed random numbers:
    $X_k^* \leftarrow \arg\max_{X_k} \mathrm{Re}\left\langle \frac{\mathcal{S}f_{X_k}}{\|\mathcal{S}f_{X_k}\|_2}, \hat{r} \right\rangle_2$, init = rand
  Step 2: Extend the support: $X \leftarrow X \cup \{X_k^*\}$
  Step 3: Find weights: $\alpha \leftarrow \arg\min_\alpha \left\| \hat{y} - \sum_{k=1}^{|X|} \alpha_k \mathcal{S}f_{X_k} \right\|_2^2$
  Step 4: Perform a descent initialized with current parameters:
    $X, \alpha \leftarrow \arg\min_{X,\alpha} \left\| \hat{y} - \sum_{k=1}^{|X|} \alpha_k \mathcal{S}f_{X_k} \right\|_2^2$, init $= (X, \alpha)$
  Step 5: Update residual: $\hat{r} \leftarrow \hat{y} - \sum_{k=1}^{|X|} \alpha_k \mathcal{S}f_{X_k}$;
end for
Normalize the weights α_k such that $\sum_k \alpha_k = 1$
return Support X, weights α

The proposed algorithm is summarized in Algorithm 5.1. In practice, we perform Step 4 with

a descent algorithm (L-BFGS). We use more iteration in the ultimate Step 4 (for $t = K$) as a "final adjustment". With this "final adjustment" step, we could reduce the running time by using fewer iterations in Step 4 for $t < K$. Our algorithm was implemented by extending the MATLAB toolbox [24]. The Matlab implementation of our approach is available at [50]. The main tool for the implementation of Algorithm 5.1 is to compute the necessary gradients for the optimization problems in Steps 1, 3, and 4.

For the following section, denote the vector $v(X) = \mathcal{S}f_X \in \mathbb{R}^m$.

5.3.1. Expression of the gradient for Step 1. In step 1, we have the optimization problem

$$X_k^* \in \underset{X_k \in \mathbb{R}^{P\times r}}{\arg\max}\, \mathrm{Re}\left\langle \frac{\mathcal{S}f_{X_k}}{\|\mathcal{S}f_{X_k}\|_2}, \hat{r} \right\rangle_2, \quad \hat{r} \in \mathbb{C}^m. \tag{5.13}$$

Let $F(X_k) = -\mathrm{Re}\left\langle \frac{\mathcal{S}f_{X_k}}{\|\mathcal{S}f_{X_k}\|_2}, \hat{r} \right\rangle_2 = -\frac{v(X_k)^T \mathrm{Re}(\hat{r})}{\|v(X_k)\|_2}$, then problem (5.13) turns to

$$X_k{}^* \in \underset{X_k \in \mathbb{R}^{P\times r}}{\arg\min}\, F(X_k). \tag{5.14}$$

In practice, with $W = [\omega_1, ..., \omega_m] \in \mathbb{R}^{P\times m}$ the frequency matrix, we compute the gradient of $F(X_k)$ with the following operation:

$$G = -\frac{1}{\|v(X_k)\|_2} W\left(W^T X_k \dot{*}\left(v\left(X_k\right)\dot{*}\left(\frac{F\left(X_k\right)v\left(X_k\right)}{\|v\left(X_k\right)\|_2} - \mathrm{Re}\left(\hat{r}\right)\right)\right)\right). \tag{5.15}$$

Here the symbol $\dot{*}$ represents the multiplication element by element in MATLAB. A matrix of size $m \times r$ multiplied using $dot*$ with a $m \times 1$ vector leads to a matrix of size $m \times r$ (multiplying all columns of the left side by the same column vector of the right side). The result G is a matrix of size $P \times r$. We need to reshape all the elements of the matrix G into a single column vector, whose result is the gradient $\nabla_{X_k} F(X_k)$. The detailed computation is in Appendix E.

5.3.2. The solution of Step 3. The problem is

$$\alpha^* = \underset{\alpha\in\mathbb{R}^{|X|}}{\arg\min} \left\| y - \sum_{k=1}^{|X|} \alpha_k \mathcal{S}f_{X_k} \right\|_2^2 \quad y \in \mathbb{C}^m. \tag{5.16}$$

Denote $V(X) = [v(X_1), ..., v(X_{|X|})] \in \mathbb{R}^{m\times|X|}$, $\alpha = [\alpha, ..., \alpha_{|X|}]^T \in \mathbb{R}^{|X|}$, then the problem can be expressed as a least-squares minimization

$$\alpha^* = \underset{\alpha\in\mathbb{R}^{|X|}}{\arg\min}\, g(\alpha) = \underset{\alpha\in\mathbb{R}^{|X|}}{\arg\min} \|y - V\alpha\|_2^2. \tag{5.17}$$

We thus have

$$\alpha^* = (V^T V)^{-1} V^T \hat{y}. \tag{5.18}$$

5.3.3. Expression of the gradient for Step 4. The problem is

$$(X^*, \alpha) \in \underset{\substack{X \in \mathbb{R}^{|X|}, X_k \in \mathbb{R}^{P+P\times r} \\ \alpha \in \mathbb{R}^{|X|}}}{\arg\min} \left\| \hat{y} - \sum_{k=1}^{|X|} \alpha_k \mathcal{S} f_{X_k} \right\|_2^2 . \tag{5.19}$$

Denote $V = [v(X_1), ..., v(X_{|}X|)]$, $\alpha = [\alpha_1, ...\alpha_K]^T$, we express

$$h(X, \alpha) = \|\hat{y} - V\alpha\|_2^2 , \tag{5.20}$$

so we have the gradients

$$\nabla_\alpha h(X, \alpha) = 2V^T(V\alpha - \hat{y}) \tag{5.21}$$

and

$$\nabla_{X_k} h(X, \alpha) = 2\alpha_k \nabla_{X_k} v(X_k)^T (V\alpha - \hat{y}). \tag{5.22}$$

In practice, as in Step 1, we compute the second gradient by calculating the matrix

$$G_2 = -2\alpha_k W(W^T X_k \dot{*} v(X_k) \dot{*} (V\alpha - y)). \tag{5.23}$$

The gradient $\nabla_{X_k} h(X, \alpha)$ corresponds the vector after reshaping G_2.

As the function minimized here is smooth; the descent will be guaranteed to converge to a local minimum. Recent works suggest that if all the OMP steps fall close enough to the Gaussian of the global optimum [55], this step will converge to the global optimum under a restricted isometry condition.

5.4. Complexity of LR-OMP. When estimating a K-components GMM, the proposed algorithm LR-OMP has a computational cost of the order of $O(mP^2rK^2)$. In each iteration, the computational cost is dominated by the matrix-vector product $W(W^T X)$ where W is a matrix of size $P \times m$ and $W^T X$ is a matrix of size $m \times r$. As $m \ll n$, the computational cost of our algorithm is lower than that of the EM. Moreover, it is possible to exploit the advantages of GPU computing, the matrix multiplication can be performed by using multiple GPUs in parallel [63]. This could result in a speed-up, especially for the "final adjustment" step.

5.5. Denoising with low-rank covariance matrices. In this section, we describe some modifications required in EPLL to use our estimated model. The estimated parameters are $\hat{\Phi} = \{\hat{X}_1, ..., \hat{X}_K, \hat{\alpha}_1, ..., \hat{\alpha}_K\}$ with $\hat{X}_k \in \mathbb{R}^{P\times r}$ and $\alpha_k \in \mathbb{R}_+$. A singular value decomposition of $\hat{X}_k$ is given by $\hat{X}_k = \hat{U}_k \hat{S}_k \hat{V}_k^T$. $\hat{U}_k, \hat{V}_k \in \mathbb{R}^{P\times P}$ are orthogonal matrices and $\hat{S}_k = \mathrm{diag}(\hat{s}_{k_1}, \cdots, \hat{s}_{k_r}, 0, \cdots, 0) \in \mathbb{R}^{P\times P}$ is a diagonal matrix. The r-rank covariance matrix can be expressed with $\hat{\Sigma}_{kr} = \hat{X}_k \hat{X}_k^T = \hat{U}_k \hat{S}_k^2 \hat{U}_k^T$. We approximate the covariance matrix Σ_k with $\Sigma_k \simeq \hat{\Sigma}_k = \hat{U}_k \hat{\Lambda}_k \hat{U}_k^T$ where $\hat{\Lambda}_k$ is formed as:

$$\hat{\Lambda}_k = \begin{pmatrix} \hat{s}_{k_1}^2 & & & & & \\ & \ddots & & & 0 & \\ & & \hat{s}_{k_r}^2 & & & \\ & & & \mu & & \\ & 0 & & & \ddots & \\ & & & & & \mu \end{pmatrix} \tag{5.24}$$

where μ is a user parameter. Denoting $\hat{U}_k^r \in \mathbb{R}^{P\times r}$ the matrix formed by the first r columns of $\hat{U}_k$ and $\hat{\Lambda}_k^r$ the matrix formed with the first r rows and r columns of $\hat{\Lambda}_k$, we have:

$$\left(\Sigma_k + \frac{1}{\beta} I_P\right)^{-1} = \hat{U}_k^r (\hat{\Lambda}_k^r + \frac{1}{\beta} I_r)^{-1} \hat{U}_k^{rT} + \frac{\beta}{\beta\mu + 1}(I_p - \hat{U}_k^r \hat{U}_k^{rT}) \tag{5.25}$$

and

$$\left(\Sigma_k + \frac{1}{\beta} I_P\right)^{-1} \Sigma_k = \hat{U}_k^r (\hat{\Lambda}_k^r + \frac{1}{\beta} I_r)^{-1} \hat{\Lambda}_k^r \hat{U}_k^{rT} + \frac{\beta\mu}{\beta\mu + 1}(I_p - \hat{U}_k^r \hat{U}_k^{rT}). \tag{5.26}$$

The detailed computation of (5.25) is in Appendix F. Then the Gaussian selection step of EPLL (2.11) becomes

$$k_i^* = \underset{1\leq k\leq K}{\arg\min} -2\log\alpha_k + \sum_{j=1}^{r}\left(\log(\hat{s}_{k_j}^2 + \frac{1}{\beta}) + \frac{[\hat{v}_i^{(k)}]_j^2}{\hat{s}_{k_j}^2 + \frac{1}{\beta}} - \frac{\beta}{\beta\mu + 1}[\hat{v}_i^{(k)}]_j^2\right) \tag{5.27}$$

where

$$\hat{v}_i^{(k)} = \hat{U}_k^{rT} \tilde{z}_i. \tag{5.28}$$

With the optimal component k_i^*, the estimated patch (2.10) becomes (recall that $\tilde{z}_i$ are centered patches and that m_i are the estimated mean of patches

$$\begin{aligned}
\hat{z}_i &= (\Sigma_{k_i^*} + \frac{1}{\beta} I_P)^{-1} \Sigma_{k_i^*} \tilde{z}_i \\
&= \hat{U}_{k_{i*}^r} (\hat{\Lambda}_{k_i^*}^r + \frac{1}{\beta} I_r)^{-1} \hat{\Lambda}_{k_i^*}^r \hat{U}_{k_i^*}^{rT} \tilde{z}_i + \frac{\beta\mu}{\beta\mu + 1}(I_p - \hat{U}_{k_i^*}^r \hat{U}_{k_i^*}^{rT})\tilde{z}_i \\
&= \hat{U}_{k_i^*}^r \hat{\Lambda}'_{k_i^*} \hat{v}_i^{(k_i^*)} + \frac{\beta\mu}{\beta\mu + 1}(\tilde{z}_i - \hat{U}_{k_r^*} \hat{v}_i^{(k_i^*)}) + m_i
\end{aligned} \tag{5.29}$$

with

$$\hat{\Lambda}'_{k_i^*} = (\hat{\Lambda}_{k^*}^r + \frac{1}{\beta} I_r)^{-1} \hat{\Lambda}_{k^*}^r = \operatorname{diag}\left(\frac{\hat{s}_{k_{ij}^*}^2}{\hat{s}_{k_{ij}^*}^2 + \frac{1}{\beta}}\right)_{j=1,\dots,r}. \tag{5.30}$$

6. Experimental Results.

In this section we present several numerical experiments to illustrate the benefits of our approach.

We randomly extract $n = 4\times 10^6$ patches of size $P = 7\times 7$ from the training images of the Berkeley Segmentation Database (BSDS) [35]. Then the patches are compressed into a sketch. Based on observations from numerical simulations, the scale parameter η must be adjusted for each task and dataset [47]. In [25], the authors propose to estimate this parameter with a small sketch on a small subset from the dataset. In our work, we choose the optimal parameter η by hand. We then learn a mixture model of $K = 20$ Gaussian components with low-rank covariance matrices. We compare the denoised results with the results obtained with a GMM (full-rank) prior model learned by the EM algorithm. For the comparison, we train the prior

from the same image patches dataset. The denoising is performed with EPLL[1] . To evaluate the quality of denoised images, we use two measures: PSNR (Peak Signal to Noise Ratio) and SSIM (Structural Similarity) [58]. For the test images, we use two datasets: Set12 [62] and BSD68 [43] for a thorough evaluation. The code is available at [50] to reproduce the results below.

Figure 4 shows the denoising performance on 6 images of the Set12 dataset. The noisy images are obtained by adding zero-mean Gaussian noise with standard deviations $\sigma = 20$ to the test images. The covariance matrices of the model learned by the sketching have the rank $r = 20$. We observe that for most of images, we obtain similar or better values of PSNR and SSIM.

Another evaluation was carried out on the images from the BSD68 dataset. The test images have been corrupted by adding white Gaussian noise with standard deviations $\sigma = 15, 60$. Table 1 shows the average PSNR and SSIM values on the dataset. On average, our approach results are 0.2dB below the results with EM in terms of PSNR. However, our approach is about 2 times faster than the EM. Moreover, a loss of 0.2dB does not affect the visual quality in most natural images.

Table 1

The average PSNR and SSIM on the BSD68 dataset with 2 different levels of noise.

σ	Sketching	EM
15	31.8 / .876	32.0 / .879
50	24.4 / .637	24.6 / .646

We also evaluate the similarity of the models learned via EM and LR-OMP. In Figure 5 we visualize the leading eigenvectors of the learned covariance whose weight is the largest. The represented eigenvectors are ordered decreasingly with respect to the eigenvalues. The figure shows that the learned components have rich and similar structures except for the smallest eigenvalues where we observe differences. As we also observe that the eigenvalues decay much faster with our method than with EM, it is hard to interpret further the difference with the result of EM. We can still say that these different "dictionaries" lead to similar denoising results.

6.1. Influence of realization of sketching operator. Our approach performs stable performances with different initialization. Table 2 shows the variability of the PSNR/SSIM over different random sketch realizations. The evaluation is carried out on the classical images: cameraman, house, etc. The noisy images are obtained by adding white Gaussian noise with standard deviations $\sigma = 20$ to the test images.

6.2. Influence of sketch size and the compression rate. Theoretically, we can successfully estimate a GMM with sufficiently separated mean and random Fourier sketching with high probability as long as the sketch size $m \geq O(K^2 P \text{polylog}(K, P))$. In our case, we learn zero-mean Gaussians. From [25], empirical results indicate that a sketch size of the order of the number of parameters is sufficient (i.e. it is conjectured that $K^2 P$ could be reduced to

[1]Matlab implementation based on the code of [38].

Figure 4. *From left to right: Original images, noisy images with noise $\sigma = 20$, results with EM model, results with LR-COMP model. The denoising results are evaluated with PSNR/SSIM. Similar denoising performances are obtained with LR-COMP with a 1000 times smaller compressed database. To estimate the prior model, our method is 2 times faster than the EM algorithm.*

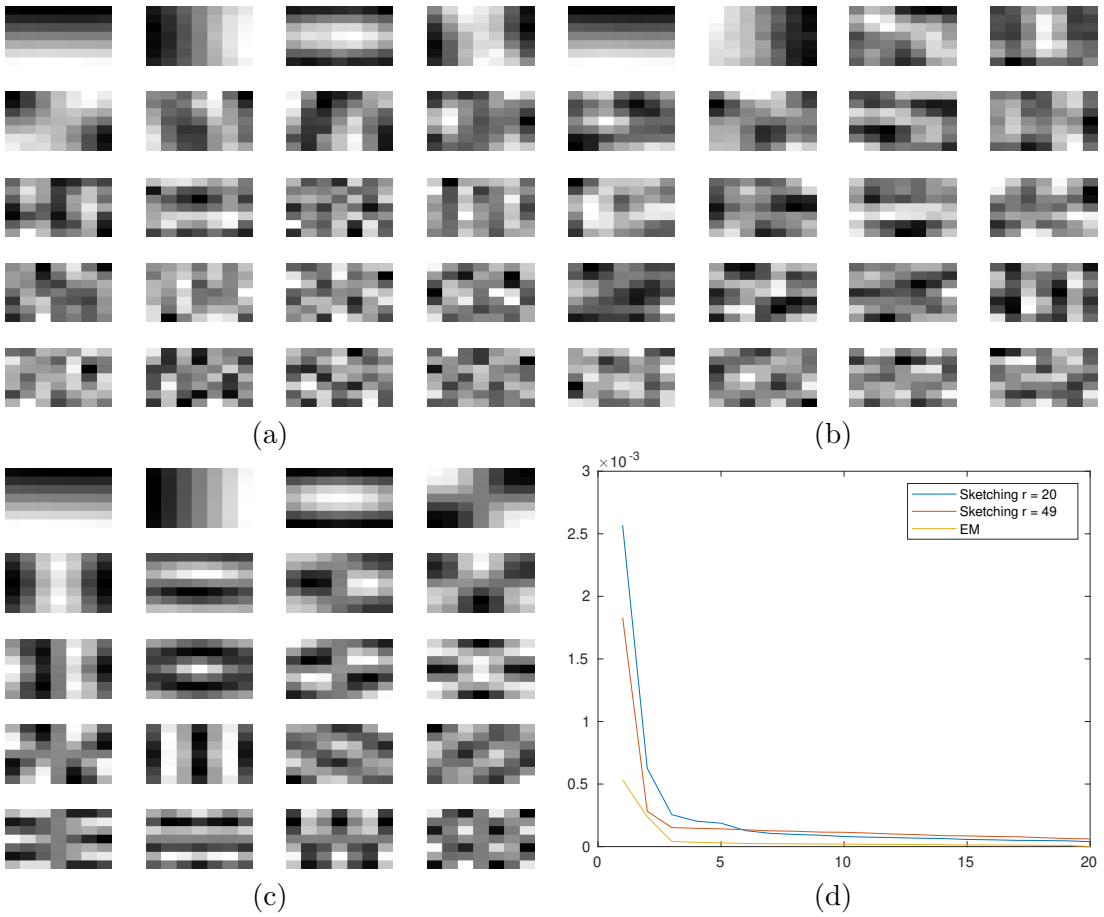

Figure 5. *The first 20 eigenvectors of the covariance matrices (for the heaviest weight) learned by LR-OMP with rank $r = 20$ (a), $r = 49$(b) and EM (c). The decay of the corresponding eigenvalues (d).*

Table 2

Image denoising performance comparison of models estimated over different random sketch realization.

	Realization 1	Realization 2	Realization 3	Realization 4
cameraman	33.1 / .930	33.3 / .930	33.4 / .930	33.4 / .930
house	35.4 / .926	35.4 / .925	35.4 / .926	35.4 / .925
jetplane	32.1 / .936	32.3 / .937	32.4 / .937	32.4 / .937
lena	32.0/ .931	32.2 / .931	32.2 / .931	32.2 / .931
pirate	29.8 / .907	30.0 / .908	30.0 / .908	30.0 / .908

KP). In our experiments, we set $m = cK(P \times r + 1) = 10K(P \times r + 1) \approx 2 \times 10^5$, i.e the compressed database is approximately 1000 times smaller than the original patch database. The gains in terms of memory is approximately $\frac{n}{m}$ times compared to the EM approach. Figure 6 shows the denoising performance and estimation time with models learned by using different sketch sizes ($c = 1, 5, 10, 20$). Our experiments show that a larger sketch size doesn't improve the denoising performance necessarily, and indeed it causes more learning time.

Figure 6. *Denoising performance (PSNR/SSIM) and estimation time (hours) of models learned with different sketch size. c =1 (a), 5 (b), 10 (c), 20 (d).*

6.3. Influence of the rank r. Figure 7 shows the denoising performance of models estimated with different ranks. Our experiments show that the model with reduced rank results in a minor PSNR/SSIM drop compared to the full-rank model. However, the learning time is much faster. According to the experiments, we cannot reduce the rank further (less than 20) to keep good denoising performance.

Figure 7. *Denoising performance (PSNR/SSIM) of models learned with different intrinsic dimensions. r = 10 (left), 20 (middle), 49(right).*

6.4. Learning time. In terms of time complexity, the running time depends on the number of components K and the complexity of the descent algorithm. In our approach, we use the Limited-memory BFGS algorithm to handle the optimization problems in Step 1 and 4. The latter is the most time-consuming part of the algorithm. To get the model ($c = 10, r = 20$) that achieves the denoising performance of our experiments (Figure 4), it takes less than 2 hours on a computer with 2 * 32 cores AMD EPYC 7452 @ 2,35 GHz. With the same environment, our learning algorithm is about 2 times faster than the EM algorithm (3.68h)[2].

[2]Mo Chen (2021). EM Algorithm for Gaussian Mixture Model (EM GMM) (https://www.mathworks.com/matlabcentral/fileexchange/26184-em-algorithm-for-gaussian-mixture-model-em-gmm), MATLAB Central File Exchange. Retrieved October 11, 2021.

7. Conclusions. In this work, we adapt the sketching framework in the context of image patches. We propose an algorithm LR-COMP to estimate a GMM with low-rank approximation and provide an implementation of the algorithm. Experiments illustrate that a high-dimensional GMM can be learned from a compressed database and then used for patch-based denoising. We achieve denoising performances close to state-of-the art model based methods while the learning procedure uses less memory and time than the classical EM algorithm.

In future works, we can generalize our approach to other models such as GGMM (Generalized Gaussian Mixture Model) for a better denoising performance [11]. We also aim to adapt the sketching to more inverse problems such as image super-resolution, image deblurring, etc. Another perspective is to extend our model to the study of video denoising method as the potential of the technique for video restoration remains unexplored. As mentioned earlier, the scale parameter η must be adjusted for each task and dataset. In our work, we choose this parameter by hand. Moreover, in our model, the intrinsic dimension of each Gaussian component is assumed the same and a prior fixed. It could be useful to design a procedure to estimate the hyper-parameters automatically in future work. In our work, we estimate a GMM with zero-mean. In this context, the notion of separation used to prove the restricted isometry property which in turn proves identifiability of the GMM in [18] and convergence of gradient descent in [55]. Proving a RIP on zero-mean would require a new notion of separation. We conjecture that an angular separation between Gaussian might enable us to prove such RIP. Such separation could e.g. compare the angle between eigenvectors of covariances by decreasing eigenvalue amplitude and weight the separation accordingly. The low-rank model should add even more separation as the Gaussian are supported on different sub-spaces. We still show empirically that the sketching process is successful without this separation assumption. This opens interesting new theoretical questions for the study of the success of compressive learning in patch-based image processing.

Appendix A. Definitions and theorems.

Definition A.1. ***Singular values*** *For $A \in \mathbb{C}^{m\times n}$ and $i = 1, ..., \min(m, n)$, the singular values $\sigma_i(A)$ (that we suppose sorted by decreasing order) of the matrix A are the absolute values of the eigenvalues of the matrix AA^T:*

$$\sigma_i^2(A) = \lambda_i(AA^T). \tag{A.1}$$

Definition A.2. ***Frobenius norm.*** *For a matrix $A \in \mathbb{C}^{m\times n}$, the Frobenius norm of A is defined as*

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{i,j}|^2} = \sqrt{\operatorname{trace}\left(A^T A\right)} = \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2(A)} \tag{A.2}$$

where $\sigma_i(A)$ are the singular values of A.

Definition A.3. ***Operator norm.*** *For a continuous linear operator $A : V \to W$, the operator norm of A is defined as*

$$\begin{aligned} \|A\|_{\text{op}} &= \inf\{c \geq 0 : \|Av\| \leq c\|v\| \quad \forall v \in V\} \\ &= \sup\left\{\frac{\|Av\|}{\|v\|} : v \neq 0 \quad \textit{and} \quad v \in V\right\}. \end{aligned} \tag{A.3}$$

Theorem A.4. ***Eckart-Young-Mirsky theorem.*** *Let $D = U\Sigma V^\top \in \mathbb{R}^{m\times n}$, $m \geq n$ be the singular value decomposition of D with $\Sigma = \mathrm{diag}(\sigma_1, ..., \sigma_m)$. Let U_r (resp. V_r) be the matrix formed by the first r columns of U (resp. V) and $\Sigma_r = \mathrm{diag}(\sigma_1, ..., \sigma_r)$. Then the r-rank matrix, obtained from the truncated singular value decomposition: $D^* = U_r\Sigma_r V_r^T$ is the minimizer of the low-rank approximation:*

$$\|D - D^*\|_F = \min_{\mathrm{rank}(D')\leq r} \|D - D'\|_F = \sqrt{\sum_{j\geq r+1} \sigma_j^2(D)}. \tag{A.4}$$

The minimizer D^* is unique if and only if $\sigma_{r+1} < \sigma_r$.

Appendix B. EM algorithm. Given a data set of n clean training patches $\chi = \{x_1, ..., x_n\} \subset \mathbb{R}^{P\times n}$, the EM algorithm for estimating a GMM can be summarized as follows:

1. Define the number of components K. For each component k, we initialize the parameters $\Theta_k = (\mu_k, \Sigma_k, \alpha_k)$ randomly, and we compute the log likelihood

$$log\mathcal{L}(\Theta_k; x_1, ..., x_n) = \sum_{i=1}^{n} log(\sum_{k=1}^{K} \alpha_k \mathcal{N}_P(x_i; \mu_k, \Sigma_k)) \tag{B.1}$$

2. E-Step
   Compute the posterior function $\Gamma_{i,k}$ with the current parameters Θ_k:

$$\Gamma_{i,k} = \frac{\alpha_k \mathcal{N}_P(x_i; \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \alpha_j \mathcal{N}_P(x_i; \mu_j, \Sigma_j)} \tag{B.2}$$

3. M-Step
   Re-estimate the parameters Θ_k^{new} with the $\Gamma_{i,k}$ obtained in the E-Step:

$$\mu_k^{new} = \frac{1}{N_k} \sum_{i=1}^{n} \Gamma_{i,k} x_i \tag{B.3}$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{i=1}^{n} \Gamma_{i,k} (x_i - \mu_k)^T (x_i - \mu_k) \tag{B.4}$$

$$\alpha_k^{new} = \frac{N_k}{\sum_{k=1}^{K} N_k} \tag{B.5}$$

   where $N_k = \sum_{i=1}^{n} \Gamma_{i,k}$.
4. Re-evaluate the log likelihood. Iterate E-Step and M-Step until the log likelihood or the parameters are not changing much.

Appendix C. Design of the adapted radius distribution. The choice of the probability distribution Δ to draw the frequencies is a key ingredient in designing the sketching operator. In our work, we choose the frequency distribution called the adapted radius [27] with a heuristic. In this appendix, we describe the design of the adapted radius distribution.

Assuming that we want to estimate a P-dimensional Gaussian $g = \mathcal{N}(0, I_P)$, we can compute the characteristic function $\psi_g(\omega)$ associated with g:

$$\psi_g(\omega) = e^{-\frac{1}{2}\omega^T\omega}. \tag{C.1}$$

The adapted radius heuristic proposes not to sample ω directly but rather to sample the radius of the P-dimensional Gaussian $R = \sqrt{\omega^T\omega}$. Thus, we draw the frequency $\omega \in \mathbb{R}^P$ as

$$\omega = R\varphi \tag{C.2}$$

where the radius $R \in \mathbb{R}_+$ is chosen with a radius distribution $R \sim p_R(R;\eta)$, and the direction $\varphi \in \mathbb{R}^P$ is uniformly generated on the l_2 unit sphere S_{P-1}, i.e. $\varphi \sim \mathcal{U}(S_{P-1})$. Then, the characteristic function $\psi_g(\omega)$ reduces to

$$\psi_g(\omega) = \psi_g(R\varphi) = e^{-\frac{1}{2}R^2} = \psi(R). \tag{C.3}$$

We obtain a one-dimensional Gaussian function for R. To design the radius distribution, we consider the estimation of a Gaussian $g = \mathcal{N}(0,1)$. We aim at sampling the radius R leading to large variations of the characteristic function when the parameters are closed to the true parameters. In other words, when parameters (μ, σ^2) are closed to (0,1), we want have a large $\left|\psi_{(\mu,\sigma^2)}(R) - \psi_{(0,1)}(R)\right|$. This can be accomplished by promoting the radius R which makes the norm of the gradient $\left\|\nabla\psi_{(\mu,\sigma^2)}(R)\right\|_2$ large. Recall that $\psi_{(\mu,\sigma^2)}(R) = e^{-i\mu R}e^{-\frac{1}{2}\sigma^2R^2}$ and the norm of the gradient is:

$$\left\|\nabla\psi_{(\mu,\sigma^2)}(R)\right\|_2^2 = \left|-iR\psi_{(\mu,\sigma^2)}(R)\right|^2 + \left|-\frac{1}{2}R^2\psi_{(\mu,\sigma^2)}(R)\right|^2 = (R^2 + \frac{1}{4}R^4)e^{-\sigma^2R^2}. \tag{C.4}$$

Therefore, $\|\nabla\psi_{(0,1)}(R)\|_2 = (R^2 + \frac{1}{4}R^4)^{\frac{1}{2}}e^{-\frac{1}{2}R^2}$. It yields the density of a radius distribution :

$$p_R(R;\eta) = ((\eta R)^2 + \frac{1}{4}(\eta R)^4)^{\frac{1}{2}}e^{-\frac{1}{2}(\eta R)^2}. \tag{C.5}$$

Here the scale parameter η should be chosen to probe the spectral content of the true GMM model.

Appendix D. Proof of Proposition 5.1.

Proof. Let $\Phi_k^* = (\Sigma_k^*, \alpha_k^*)$ be the minimizer of the problem (5.5), i.e.

$$\Phi_k^* \in \underset{\substack{\Sigma_k \in \mathbb{R}^{P\times P} \\ \alpha_k \geq 0, \sum_{k=1}^K \alpha_k = 1}}{\arg\min} \sum_{l=1}^m \left| \sum_{k=1}^K \alpha_k e^{-\frac{1}{2}\omega_l^T\Sigma_k\omega_l} - \hat{y}_l \right|^2 \tag{D.1}$$

and suppose that there exists a minimizer $\hat{\Phi}_k = (\hat{X}_k, \hat{\alpha}_k)$ for the problem (5.8):

$$\hat{\Phi}_k \in \underset{\substack{X_k \in \mathbb{R}^{P\times r} \\ \alpha_k \geq 0, \sum_{k=1}^K \alpha_k = 1}}{\arg\min} \sum_{l=1}^m \left| \sum_{k=1}^K \alpha_k e^{-\frac{1}{2}\omega_l^T X_k X_k^T \omega_l} - \hat{y}_l \right|^2 . \tag{D.2}$$

Let $\tilde{\Sigma}_k$ be the best rank-r approximation of Σ_k^* with the rank r i.e.

$$\tilde{\Sigma}_k \in \underset{\Sigma, \mathrm{rank}(\Sigma)=r}{\arg\min} \|\Sigma_k^* - \Sigma\|_F^2 . \tag{D.3}$$

Define $\tilde{\Phi} = (\tilde{\Sigma}_k, \alpha_k^*)$. According to the definition (D.2) and the triangle inequality, we have

$$\begin{aligned} \left\|\mathcal{S}f_{\hat{\Phi}} - y\right\|_2 &\leq \left\|\mathcal{S}f_{\tilde{\Phi}} - \hat{y}\right\|_2 \\ &= \left\|\mathcal{S}f_{\tilde{\Phi}} - \mathcal{S}f_{\Phi^*} + \mathcal{S}f_{\Phi^*} - \hat{y}\right\|_2 \\ &\leq \left\|\mathcal{S}f_{\tilde{\Phi}} - \mathcal{S}f_{\Phi^*}\right\|_2 + \left\|\mathcal{S}f_{\Phi^*} - y\right\|_2 \end{aligned} . \tag{D.4}$$

The first term is

$$\begin{aligned} \left\|\mathcal{S}f_{\tilde{\Phi}} - \mathcal{S}f_{\Phi^*}\right\|_2^2 &= \left\| \sum_{k=1}^K \alpha_k^* \mathcal{S}(f_{\tilde{\Sigma}_k} - f_{\Sigma_k^*}) \right\|_2^2 = \sum_{l=1}^m \left| \sum_{k=1}^K \alpha_k^* \left(e^{-\frac{1}{2}\omega_l^T \tilde{\Sigma}_k \omega_l} - e^{-\frac{1}{2}\omega_l^T \Sigma_k^* \omega_l} \right) \right|^2 \\ &= \sum_{l=1}^m \left| \sum_{k=1}^K \alpha_k^* e^{-\frac{1}{2}\omega_l^T \tilde{\Sigma}_k \omega_l} \left(1 - e^{-\frac{1}{2}\omega_l^T (\Sigma_k^* - \tilde{\Sigma}_k) \omega_l} \right) \right|^2 . \end{aligned} \tag{D.5}$$

Using the convexity inequality $|1 - e^{-x}| \leq |x|$ and Cauchy–Schwarz inequality, we have

$$\begin{aligned} \left| e^{-\frac{1}{2}\omega_l^T \tilde{\Sigma}_k \omega_l} (1 - e^{-\frac{1}{2}\omega_l^T (\Sigma_k^* - \tilde{\Sigma}_k) \omega_l}) \right| &\leq \left| 1 - e^{-\frac{1}{2}\omega_l^T (\Sigma_k^* - \tilde{\Sigma}_k) \omega_l} \right| \\ &\leq \frac{1}{2} \left| \omega_l^T (\Sigma_k^* - \tilde{\Sigma}_k) \omega_l \right| = \frac{1}{2} \left| \langle \omega_l, (\Sigma_k^* - \tilde{\Sigma}_k) \omega_l \rangle \right| . \end{aligned} \tag{D.6}$$

By the Eckart-Young-Mirsky theorem, we have that the largest singular value of $\Sigma_k^* - \tilde{\Sigma}_k$ is $\sigma_{r+1}(\Sigma_k^*)$ and

$$\left| \langle \omega_l, (\Sigma_k^* - \tilde{\Sigma}_k) \omega_l \rangle \right| \leq \|\omega_l\|_2^2 \|(\Sigma_k^* - \tilde{\Sigma}_k)\|_{\mathrm{op}} = \|\omega_l\|_2^2 \sigma_{r+1}(\Sigma_k^*). \tag{D.7}$$

Therefore, using $\sum_{k=1}^K \alpha_k = \|\alpha\|_1 = 1$ and Hölder's inequality, and upper bound on (D.5) reads

$$\begin{aligned} \left\|\mathcal{S}f_{\tilde{\Phi}} - \mathcal{S}f_{\Phi^*}\right\|_2^2 &\leq \frac{1}{4} \sum_{l=1}^m \left| \|\alpha\|_1 \max_k \left(\|\omega_l\|_2^2 \sigma_{r+1}(\Sigma_k^*) \right) \right|^2 \\ &= \left(\frac{1}{4} \sum_{l=1}^m \|\omega_l\|_2^4 \right) \max_k \left(\sigma_{r+1}(\Sigma_k^*) \right)^2 . \end{aligned} \tag{D.8}$$

Denoting $C = \frac{1}{2}\sqrt{\sum_{l=1}^{m} \|\omega_l\|_2^4}$, we have from (D.4) that:

$$\left\|\mathcal{S}f_{\hat{\Phi}} - \hat{y}\right\|_2 \leq C \max_k(\sigma_{r+1}(\Sigma_k^*)) + \|\mathcal{S}f_{\Phi^*} - \hat{y}\|_2 \,. \tag{D.9}$$

■

Appendix E. Calculation of the gradient. Denote $F(X_k) = -\frac{v(X_k)^T \hat{r}}{\|v(X_k)\|_2}$, where $r \in \mathbb{R}^m$ is the real part of $\hat{r}$. We compute the gradient of F as follows:

$$\begin{aligned}
\nabla_{X_k} F(X_k) &= -\frac{1}{\|v(X_k)\|_2^2}\left((\nabla_{X_k} v(X_k))^T r \|v(X_k)\|_2 - \frac{v(X_k)^T r (\nabla_{X_k} v(X_k))^T v(X_k)}{\|v(X_k)\|_2}\right) \\
&= -\frac{(\nabla_{X_k} v(X_k))^T}{\|v(X_k)\|_2}\left(r + \frac{v(X_k)^T r v(X_k)}{\|v(X_k)\|_2^2}\right) \\
&= \frac{(\nabla_{X_k} v(X_k))^T}{\|v(X_k)\|_2}\left(\frac{F(X_k) v(X_k)}{\|v(X_k)\|_2} - r\right).
\end{aligned} \tag{E.1}$$

For each component $v_l(X_k) = e^{-\frac{1}{2}\omega_l^T {X_k}{X_k}^T \omega_l}$, we have

$$\frac{\partial v_l(X_k)}{\partial X_k} = -v_l(X_k) {X_k}^T \omega_l \omega_l^T \,. \tag{E.2}$$

Then for a given vector $z \in \mathbb{R}^m$

$$\langle \nabla_{X_k} v(X_k), z\rangle = -\sum_{l=1}^{m} z_l v_l(X_k) {X_k}^T \omega_l \omega_l^T. \tag{E.3}$$

In practice, we compute the scalar product with

$$\langle \nabla_{X_k} v(X_k), z\rangle = -W(W^T X_k \dot{*} (v(X_k) \dot{*} z)) \tag{E.4}$$

where $W = [\omega_1, ..., \omega_m] \in M_{P,m}(\mathbb{R})$ the frequency matrix and $\dot{*}$ the multiplication element by element in MATLAB. We compute the matrix G as

$$G = -\frac{1}{\|v(X_k)\|_2} W\left(W^T X_k \dot{*}\left(v(X_k) \dot{*}\left(\frac{F(X_k) v(X_k)}{\|v(X_k)\|_2} - r\right)\right)\right). \tag{E.5}$$

As a consequence, we reshape the matrix G to a vector which corresponds the gradient $\nabla_{X_k} F(X_k)$.

Appendix F. The expression of (5.25). Denoting $\hat{U}_k^r \in \mathbb{R}^{P\times r}$ (resp. $\hat{U}_k^c \in \mathbb{R}^{P\times(P-r)}$)the matrix formed by the first r (resp. the last $P-r$) columns of $\hat{U}_k$ and $\hat{\Lambda}_k^r$ the matrix formed with the first r rows and r columns of $\hat{\Lambda}_k$. We use the bloc matrix multiplication:

$$\begin{aligned}
\left(\Sigma_k + \frac{1}{\beta} I_P\right)^{-1} &= \hat{U}_k (\hat{\Lambda}_k + \frac{1}{\beta} I)^{-1} \hat{U}_k^T \\
&= \hat{U}_k^r (\hat{\Lambda}_k^r + \frac{1}{\beta} I_r)^{-1} \hat{U}_k^{rT} + (\mu + \frac{1}{\beta} I_{P-r})^{-1} \hat{U}_k^c \hat{U}_k^{cT}.
\end{aligned} \tag{F.1}$$

We have $\hat{U}_k^c \hat{U}_k^{cT} = (I_p - \hat{U}_k^r \hat{U}_k^{rT})$, thus

$$\left(\Sigma_k + \frac{1}{\beta} I_P\right)^{-1} = \hat{U}_k^r (\hat{\Lambda}_k^r + \frac{1}{\beta} I_r)^{-1} \hat{U}_k^{rT} + \frac{\beta}{\beta\mu + 1}(I_p - \hat{U}_k^r \hat{U}_k^{rT}). \tag{F.2}$$

Acknowledgments. The authors acknowledge the support of the French National Research Agency (ANR) under reference ANR-20-CE40-0001 (EFFIREG project). Experiments presented in this paper were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see https://www.plafrim.fr/). The authors are indebted to anonymous reviewers for providing insightful comments which have resulted in this paper.

REFERENCES

[1] C. Aguerrebere, A. Almansa, Y. Gousseau, J. Delon, and P. Musé, *Single shot high dynamic range imaging using piecewise linear estimators*, in 2014 IEEE International Conference on Computational Photography (ICCP), IEEE, 2014, pp. 1–10.
[2] C. Bouveyron, S. Girard, and C. Schmid, *High-dimensional data clustering*, Computational statistics & data analysis, 52 (2007), pp. 502–519.
[3] A. Buades, B. Coll, and J.-M. Morel, *A non-local algorithm for image denoising*, in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 2, IEEE, 2005, pp. 60–65.
[4] S. Burer and R. D. Monteiro, *Local minima and convergence in low-rank semidefinite programming*, Mathematical Programming, 103 (2005), pp. 427–444.
[5] E. Byrne, A. Chatalic, R. Gribonval, and P. Schniter, *Sketched clustering via hybrid approximate message passing*, IEEE Transactions on Signal Processing, 67 (2019), pp. 4556–4569.
[6] N. Cai, Y. Zhou, S. Wang, B. W.-K. Ling, and S. Weng, *Image denoising via patch-based adaptive gaussian mixture prior method*, Signal, Image and Video Processing, 10 (2016), pp. 993–999.
[7] A. Chatalic, R. Gribonval, and N. Keriven, *Large-scale high-dimensional clustering with fast sketching*, in 2018 International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 4714–4718.
[8] Y. Chi, Y. M. Lu, and Y. Chen, *Nonconvex optimization meets low-rank matrix factorization: An overview*, IEEE Transactions on Signal Processing, 67 (2019), pp. 5239–5269.
[9] G. Cormode and S. Muthukrishnan, *An improved data stream summary: the count-min sketch and its applications*, Journal of Algorithms, 55 (2005), pp. 58–75.
[10] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, *Image denoising by sparse 3-d transform-domain collaborative filtering*, IEEE Transactions on image processing, 16 (2007), pp. 2080–2095.
[11] C.-A. Deledalle, S. Parameswaran, and T. Q. Nguyen, *Image denoising with generalized gaussian mixture model patch priors*, SIAM Journal on Imaging Sciences, 11 (2018), pp. 2568–2609.
[12] J. Feng, L. Song, X. Huo, X. Yang, and W. Zhang, *Image restoration via efficient gaussian mixture model learning*, in 2013 IEEE International Conference on Image Processing, IEEE, 2013, pp. 1056–1060.
[13] M. Fontaine, C. Vanwynsberghe, A. Liutkus, and R. Badeau, *Scalable source localization with multichannel α-stable distributions*, in 2017 25th European Signal Processing Conference (EUSIPCO), IEEE, 2017, pp. 11–15.
[14] M. Fontaine, C. Vanwynsberghe, A. Liutkus, and R. Badeau, *Sketching for nearfield acoustic imaging of heavy-tailed sources*, in International Conference on Latent Variable Analysis and Signal Separation, Springer, 2017, pp. 80–88.
[15] S. Foucart and H. Rauhut, *A mathematical introduction to compressive sensing*, Springer, 2013.
[16] D. Geman and C. Yang, *Nonlinear image recovery with half-quadratic regularization*, IEEE transactions on Image Processing, 4 (1995), pp. 932–946.

[17] R. Gribonval, G. Blanchard, N. Keriven, and Y. Traonmilin, *Compressive statistical learning with random feature moments*, Mathematical Statistics and Learning, 3 (2021), pp. 113–164.
[18] R. Gribonval, G. Blanchard, N. Keriven, and Y. Traonmilin, *Statistical learning guarantees for compressive clustering and compressive mixture modeling*, Mathematical Statistics and Learning, 3 (2021), pp. 165–257.
[19] R. Gribonval, A. Chatalic, N. Keriven, V. Schellekens, L. Jacques, and P. Schniter, *Sketching datasets for large-scale learning (long version)*, arXiv preprint arXiv:2008.01839, (2020).
[20] R. Gribonval, A. Chatalic, N. Keriven, V. Schellekens, L. Jacques, and P. Schniter, *Sketching data sets for large-scale learning: Keeping only what you need*, IEEE Signal Processing Magazine, 38 (2021), pp. 12–36.
[21] J. Hertrich, D. P. L. Nguyen, J.-F. Aujol, D. Bernard, Y. Berthoumieu, A. Saadaldin, and G. Steidl, *PCA reduced gaussian mixture models with applications in superresolution*, arXiv preprint arXiv:2009.07520, (2020).
[22] A. Houdard, C. Bouveyron, and J. Delon, *High-dimensional mixture models for unsupervised image denoising (HDMI)*, SIAM Journal on Imaging Sciences, 11 (2018), pp. 2815–2846.
[23] P. Jain, A. Tewari, and I. S. Dhillon, *Orthogonal matching pursuit with replacement*, arXiv preprint arXiv:1106.2774, (2011).
[24] N. Keriven, *SketchMLbox – A MATLAB toolbox for large-scale mixture learning*, Mar. 2018, https://hal.inria.fr/hal-02960718.
[25] N. Keriven, A. Bourrier, R. Gribonval, and P. Pérez, *Sketching for large-scale learning of mixture models*, Information and Inference: A Journal of the IMA, 7 (2018), pp. 447–508.
[26] N. Keriven, A. Deleforge, and A. Liutkus, *Blind source separation using mixtures of alpha-stable distributions*, in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 771–775.
[27] N. Keriven, N. Tremblay, Y. Traonmilin, and R. Gribonval, *Compressive k-means*, in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2017, pp. 6369–6373.
[28] F. Laus, M. Nikolova, J. Persch, and G. Steidl, *A nonlocal denoising algorithm for manifold-valued images using second order statistics*, 2016.
[29] M. Lebrun, *An analysis and implementation of the bm3d image denoising method*, Image Processing On Line, 2012 (2012), pp. 175–213.
[30] M. Lebrun, A. Buades, and J.-M. Morel, *Implementation of the "non-local bayes" (nl-bayes) image denoising algorithm*, Image Processing On Line, 2013 (2013), pp. 1–42.
[31] M. Lebrun, A. Buades, and J.-M. Morel, *A nonlocal bayesian image denoising algorithm*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 1665–1688.
[32] A. Levin and B. Nadler, *Natural image denoising: Optimality and inherent bounds*, in CVPR 2011, IEEE, 2011, pp. 2833–2840.
[33] E. Luo, S. H. Chan, and T. Q. Nguyen, *Adaptive image denoising by mixture adaptation*, IEEE transactions on image processing, 25 (2016), pp. 4489–4503.
[34] S. G. Mallat and Z. Zhang, *Matching pursuits with time-frequency dictionaries*, IEEE Transactions on signal processing, 41 (1993), pp. 3397–3415.
[35] D. Martin, C. Fowlkes, D. Tal, and J. Malik, *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*, in Proceedings 8th International Conference on Computer Vision. ICCV 2001, vol. 2, IEEE, 2001, pp. 416–423.
[36] G. J. McLachlan and T. Krishnan, *The EM algorithm and extensions*, vol. 382, John Wiley & Sons, 2007.
[37] V. Papyan and M. Elad, *Multi-scale patch-based image restoration*, IEEE Transactions on image processing, 25 (2015), pp. 249–261.
[38] S. Parameswaran, C.-A. Deledalle, L. Denis, and T. Q. Nguyen, *Accelerating gmm-based patch priors for image restoration: Three ingredients for a* 100× *speed-up*, IEEE Transactions on Image Processing, 28 (2018), pp. 687–698.
[39] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, *Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition*, in Proceedings of 27th Asilomar conference on signals, systems and computers, IEEE, 1993, pp. 40–44.

[40] O. Permiakova and T. Burger, *Sketched stochastic dictionary learning for large-scale data and application to high-throughput mass spectrometry*, Statistical Analysis and Data Mining: The ASA Data Science Journal, (2021).
[41] Y. Ren, Y. Romano, and M. Elad, *Example-based image synthesis via randomized patch-matching*, IEEE Transactions On Image Processing, 27 (2017), pp. 220–235.
[42] F. Renna, R. Calderbank, L. Carin, and M. R. Rodrigues, *Reconstruction of signals drawn from a gaussian mixture via noisy compressive measurements*, IEEE Transactions on Signal Processing, 62 (2014), pp. 2265–2277.
[43] S. Roth and M. J. Black, *Fields of experts: A framework for learning image priors*, in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 2, IEEE, 2005, pp. 860–867.
[44] A. Saint-Dizier, J. Delon, and C. Bouveyron, *A unified view on patch aggregation*, Journal of Mathematical Imaging and Vision, 62 (2020), pp. 149–168.
[45] V. Schellekens and L. Jacques, *Compressive classification (machine learning without learning)*, arXiv preprint arXiv:1812.01410, (2018).
[46] V. Schellekens and L. Jacques, *Compressive learning of generative networks*, arXiv preprint arXiv:2002.05095, (2020).
[47] V. Schellekens and L. Jacques, *When compressive learning fails: blame the decoder or the sketch?*, arXiv preprint arXiv:2009.08273, (2020).
[48] M. P. Sheehan, M. S. Kotzagiannidis, and M. E. Davies, *Compressive independent component analysis*, in 2019 27th European Signal Processing Conference (EUSIPCO), IEEE, 2019, pp. 1–5.
[49] M. P. Sheehan, J. Tachella, and M. E. Davies, *A sketching framework for reduced data transfer in photon counting lidar*, arXiv preprint arXiv:2102.08732, (2021).
[50] H. Shi, https://github.com/shihui1224/sketching-for-denoising.
[51] H. Shi, Y. Traonmilin, and J.-F. Aujol, *Sketched learning for image denoising*, in The Eighth International Conference on Scale Space and Variational Methods in Computer Vision (SSVM), Cabourg, France, May 2021.
[52] J. Sulam and M. Elad, *Expected patch log likelihood with a sparse prior*, in Energy Minimization Methods in Computer Vision and Pattern Recognition, X.-C. Tai, E. Bae, T. F. Chan, and M. Lysaker, eds., Cham, 2015, Springer International Publishing, pp. 99–111.
[53] M. E. Tipping and C. M. Bishop, *Mixtures of probabilistic principal component analyzers*, Neural computation, 11 (1999), pp. 443–482.
[54] D.-V. Tran, S. Li-Thiao-Té, M. Luong, T. Le-Tien, and F. Dibos, *Number of useful components in gaussian mixture models for patch-based image denoising*, in Image and Signal Processing, A. Mansouri, A. El Moataz, F. Nouboud, and D. Mammass, eds., Cham, 2018, Springer International Publishing, pp. 108–116.
[55] Y. Traonmilin, J. Aujol, and A. Leclaire, *The basins of attraction of the global minimizers of non-convex inverse problems with low-dimensional models in infinite dimension*, Preprint, abs/2009.08670 (2020).
[56] R. Varadhan and C. Roland, *Simple and globally convergent methods for accelerating the convergence of any em algorithm*, Scandinavian Journal of Statistics, 35 (2008), pp. 335–353.
[57] Y.-Q. Wang and J.-M. Morel, *Sure guided gaussian mixture image denoising*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 999–1034.
[58] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, *Image quality assessment: from error visibility to structural similarity*, IEEE Transactions on Image Processing, 13 (2004), pp. 600–612.
[59] J. Xu, L. Zhang, W. Zuo, D. Zhang, and X. Feng, *Patch group based nonlocal self-similarity prior learning for image denoising*, in Proceedings of the IEEE international conference on computer vision, 2015, pp. 244–252.
[60] J. Yang, X. Yuan, X. Liao, P. Llull, D. J. Brady, G. Sapiro, and L. Carin, *Video compressive sensing using gaussian mixture models*, IEEE Transactions on Image Processing, 23 (2014), pp. 4863–4878.
[61] G. Yu, G. Sapiro, and S. Mallat, *Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity*, IEEE Transactions on Image Processing, 21 (2011), pp. 2481–2499.

[62] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, *Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising*, IEEE transactions on image processing, 26 (2017), pp. 3142–3155.
[63] P. Zhang and Y. Gao, *Matrix multiplication on high-density multi-gpu architectures: Theoretical and experimental investigations*, in IEEE International Conference on High Performance Computing, Data, and Analytics, 2015.
[64] D. Zoran and Y. Weiss, *From learning models of natural image patches to whole image restoration*, in 2011 International Conference on Computer Vision, IEEE, 2011, pp. 479–486.