Unrolled Variational Bayesian Algorithm for Image Blind Deconvolution

Yunshi Huang, Emilie Chouzenoux, Jean-Christophe Pesquet

Unrolled Variational Bayesian Algorithm for Image Blind Deconvolution

Yunshi Huang, Emilie Chouzenoux, *Senior Member, IEEE* and Jean-Christophe Pesquet, *Fellow, IEEE*

***Abstract*—In this paper, we introduce a variational Bayesian algorithm (VBA) for image blind deconvolution. Our VBA generic framework incorporates smoothness priors on the unknown blur/image and possible affine constraints (e.g., sum to one) on the blur kernel, integrating the VBA within a neural network paradigm following an unrolling methodology. The proposed architecture is trained in a supervised fashion, which allows us to optimally set two key hyperparameters of the VBA model and leads to further improvements in terms of resulting visual quality. Various experiments involving grayscale/color images and diverse kernel shapes, are performed. The numerical examples illustrate the high performance of our approach when compared to state-of-the-art techniques based on optimization, Bayesian estimation, or deep learning.**

***Index Terms*—Variational Bayesian approach, Kullback-Leibler divergence, Majorization-Minimization, blind deconvolution, image restoration, neural network, unrolling, deep learning.**

I. Introduction

Problem statement: The image blind deconvolution problem appears in many fields of image processing, such as astronomy [1], biology [2] and medical imaging [3]. Given a degraded, blurred, and noisy image, the aim is to restore a clean image along with an estimate of the blur kernel. Blind deconvolution is a very difficult problem to engage with, as there exists an infinite number of pairs (image/blur) that lead to the same observed image.

State-of-the-art review: Blind deconvolution methods available in literature adopt either a sequential identification process [4] or a joint estimation approach [5]. In the former, the blur kernel is identified first, possibly through a calibration step [6], [7], [8]. Then the unknown image is then inferred using a non-blind image restoration method. In the latter, the blur kernel and unknown image are simultaneously estimated. Since the problem is highly ill-posed, it is mandatory to incorporate prior knowledge on the sought unknowns. The retained prior strongly influences the choice for the solver.

Three main categories of joint blind deconvolution approaches can be distinguished. A first category consists of formulating the problem as the minimization of a cost function that gathers both a data fidelity term (e.g., least-squares discrepancy) and penalties/constraints acting on the image and kernel variables. Imposing normalization and sparsity enhancing constraints on the kernel coefficients is a standard method for avoiding scale ambiguity inherent to the blind deconvolution model [9], [10], [11]. The smoothness of the image can be easily imposed, by adopting total-variation based regularization [12]. Several other efficient choices have been proposed in the literature, along with suitable iterative optimization methods to solve the resulting problems [13], [14], [15], [16], [17], [18]. The methods' flexibility is among its main advantages, although this advantage comes at the price of heavy parameter tuning.

The second category involves resorting to a Bayesian formulation to express the model and a priori knowledge on the variables. The estimates are then defined from the estimation of the moments (typically the mean) of a posterior distribution given the observed data and prior. This action of defining the estimates uses either sampling [19], [20] or approximation [21] strategies that typically involve the evaluation of intractable integrals. Markov chain Monte Carlo (MCMC) methods have been widely used for blind deconvolution involving 1D sparse signals [22], [23], [24], but it is rarely employed in large-scale problems [25], due to computational reasons. Another family of methods consists in adopting the so-called variational Bayesian approximation paradigm [26], [27]. A simpler (usually separable) approximation to the posterior is then built through the minimization of a suitable divergence. This approach leads to fast Bayesian-based algorithms, whose great performance has been assessed in the context of non-blind [28], [29] and blind [30] image restoration. Bayesian-based techniques usually require less parameters than optimization-based ones. Moreover, they can also provide higher-order moments estimates, such as covariance matrices, which are of high interest for assessing probabilistically the uncertainty of the results. However, dealing with complex noise models and priors in such methods may be tricky, and the algorithms may be quite computationally heavy as they possess hyperparameters to be tuned. A recent study is to insert optimization-based steps in Bayesian sampling/approximation methods for more versatile modalities and faster computations, particularly in the context of large-scale image processing [28], [31], [32].

The third category of methods relies on deep learning methodologies that have been cultivated during the last decade [33], [34], [35]. More precisely, such methodologies adopt a supervised learning strategy to learn (implicitly) some prior information on the image/kernel from a so-called training set. These methodologies then build a highly non-linear and multi-layers architecture. The parameters of the architecture (i.e., neuron weights) are estimated by back-propagation to minimize a given loss function associated with the task at

The authors acknowledge support from the *Agence Nationale de la Recherche* of France under MAJIC (ANR-17-CE40-0004-01) project, and from the European Research Council Starting Grant MAJORIS ERC-2019-STG-850925.

The authors are with Université Paris-Saclay, Inria, CentraleSupélec, Centre de Vision Numérique (France).

hand (e.g., image visual quality). Several recent works propose neural network architectures dedicated to the problem of image blind deconvolution. For example, DeblurGAN [33] is based on conditional generative adversarial networks and a multi-component loss function. SRCNN [34] and its extended version, DBSRCNN [35] rely on a CNN architecture. SelfDeblur [36] combines an optimization-based method with two generative networks for modeling deep priors on the image and the blur kernel, respectively. Other image deblurring problems have been studied using neural network techniques. For example, [37] presents an unsupervised method based on multi-adversarial CycleGAN [38] for high-resolution image generation and [39] proposes a dynamic scene deblurring method for unconstrained dual-lens (DL) cameras. Most of the methods proposed in the above works are supervised. They can perform quite well, as long as the training set is large and representative enough. Moreover, they are well suited to GPU-based implementation. However, such methods have traditionally suffered from lack of interpretability and robustness [40].

(Hyper)parameter tuning in particular presents a significant challenge found in most traditional image restoration methods. The parameters that require tuning can arise from the model itself (e.g., prior weight, noise level) or from the resolution method (e.g., algorithm stepsize). Several strategies based on empirical search as well as more advanced statistical tools can be used for the tuning, but these strategies are often time-consuming with not guaranteed success. However, an ambitious emerging set of methods in the field of inverse problems in signal/image processing resolves these issues by performing algorithm unrolling [41]. An iterative method (e.g., an optimization algorithm) is unrolled as layers of a neural network. The reduced set of parameters of this network are learnt by supervised training. Such unrolling algorithms have been developed in [42], [43], [44], [45] in the context of image deconvolution, as well as in [46] in other image restoration applications, with promising results. One key advantage of unrolling the algorithm is that cumbersome parameter tuning can be avoided by simply embedding the sought parameters into the network and learning them at training phase. Theoretical results demonstrating the stability and robustness of unrolling techniques can be found in [44], [47], [48]. These methods are also closely related to plug-and-play techniques where a trained neural network is employed as the denoiser [44], [49], [50]. Deep unrolling can also be combined with a bilevel optimization framework while maintaining the aim of efficient hyperparameter learning (see [51] for a recent survey).

Contributions: In this paper, we propose a novel approach for blind image deconvolution that aims at gathering the best of the aforementioned methods. We first introduce a variational Bayesian algorithm (VBA) enhanced by optimization-based ideas from [28]. The advantages of VBA is to cope with a large set of priors on the kernel and the image, as well as to present a reduced computational cost. Then, we apply the unrolling paradigm to create a deep neural network architecture, where VBA iterations are integrated as layers. This allows us to (i) learn the hyperparameters (in particular, the noise level) of VBA in an automatic supervised fashion, (ii) improve further the quality of the results by choosing a dedicated loss in the training phase, (iii) implement the method by taking full advantage of possible GPU resources, thus considerably reducing the processing time during the test phase. In contrast to standard deep learning methods for blind deconvolution, all these benefits come along with a preservation of the interpretability of the method, thanks to the unrolling technique. It should be emphasized that variational Bayesian methods often appear in deep learning context. Indeed, they are backbones of variational autoencoders [52] and also constitute methods of choice for training Bayesian neural networks [53]. However, up to our knowledge, our work is the first to investigate the unrolling of a variational Bayesian technique.

Organization of the paper: The rest of our paper is organized as follows. Section II introduces the image degradation model as well as our Bayesian modeling; this section also provides the background for deriving our algorithm. Section III explicitly describes the iterative updates of the proposed VBA. The unrolling of VBA is presented in Section IV. Numerical results, including comparisons with various methods, are presented in Section V. Finally, Section VI concludes this paper by addressing the advantages brought by our proposed algorithm.

II. Problem statement

A. Observation model

We focus on the restoration of an original image $\widetilde{\mathbf{x}} \in \mathbb{R}^N$, from a degraded version of it $\mathbf{y} \in \mathbb{R}^N$, related to $\widetilde{\mathbf{x}}$ according to the following model:

$$\mathbf{y} = \widetilde{\mathbf{H}}\widetilde{\mathbf{x}} + \mathbf{n}. \tag{1}$$

Hereabove, $\mathbf{n} \in \mathbb{R}^N$ models some additive random perturbation on the observation. Moreover, $\widetilde{\mathbf{H}} \in \mathbb{R}^{N\times N}$ is a linear operator modeling the effect of a blur kernel $\widetilde{\mathbf{h}} \in \mathbb{R}^M$. In this work, we focus on the generalized blind deconvolution problem where the matrix associated with a given kernel $\mathbf{h} = [h_1, \ldots, h_M]^\top$ reads

$$\mathbf{H} = \sum_{m=1}^{M} h_m \mathbf{S}_m, \tag{2}$$

with $\{\mathbf{S}_1, \ldots, \mathbf{S}_M\}$ is a set of M known sparse $N \times N$ real-valued matrices. This model allows to retrieve the standard image deblurring model, as a special case when $\mathbf{H}$ identifies with a 2D discrete convolution matrix with suitable padding. The considered problem amounts to retrieving an estimate $(\widehat{\mathbf{x}}, \widehat{\mathbf{h}})$ of the pair of variables $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{h}})$ given $\mathbf{y}$. Due to the ill-posedness of this inverse problem, assumptions are required on the sought image / kernel and on the noise statistics to reach satisfying results. In the sequel, we will assume that $\mathbf{n}$ is a realization of an additive Gaussian noise with zero mean and standard deviation σ. In the remainder of the paper, it will be convenient to set $\beta = \sigma^{-2}$. Furthermore, we introduce a linear equality constraint on the blur kernel estimate $\mathbf{h}$. A general expression of such a constraint is as follows:

$$\mathbf{h} = \mathbf{T}\mathbf{z} + \mathbf{t}, \tag{3}$$

where $\mathbf{T} = (T_{m,p})_{1\leq m\leq M, 1\leq p\leq P} \in \mathbb{R}^{M\times P}$ is a matrix of rank $P \in \{1,\dots,M\}$ and $\mathbf{t} = [t_1,\dots,t_M]^\top \in \mathbb{R}^M$ is some vector of predefined constants. Vector $\mathbf{z} = [z_1,\dots,z_P]^\top \in \mathbb{R}^P$ becomes the new unknown of the problem, along with the image $\mathbf{x}$. A typical linear equality constraint in such context is the sum-to-one constraint, i.e. $\sum_{m=1}^M h_m = 1$. Other examples will be provided in the experimental section. We can thus rewrite (2) as

$$\mathbf{H} = \sum_{p=1}^{P} z_p \mathbf{K}_p + \mathbf{K}_0 = \mathcal{H}(\mathbf{z}), \tag{4}$$

with

$$(\forall p \in \{1,\dots,P\}) \quad \mathbf{K}_p = \sum_{m=1}^{M} T_{m,p}\mathbf{S}_m \in \mathbb{R}^{N\times N}, \tag{5}$$

and

$$\mathbf{K}_0 = \sum_{m=1}^{M} t_m \mathbf{S}_m \in \mathbb{R}^{N\times N}. \tag{6}$$

B. Hierarchical Bayesian Modeling

Let us now introduce the hierarchical Bayesian model on which our VBA method will be grounded.

1) Likelihood: First, we express the likelihood $p(\mathbf{y}|\mathbf{x},\mathbf{z})$ of the observed data, given the unknowns $(\mathbf{x},\mathbf{z})$. Since the noise is assumed to be Gaussian distributed, the likelihood can be expressed as follows:

$$p(\mathbf{y}|\mathbf{x},\mathbf{z}) = \beta^{\frac{N}{2}} \exp\left(-\frac{\beta}{2}||\mathbf{y}-\mathcal{H}(\mathbf{z})\mathbf{x}||^2\right), \tag{7}$$

where we recall that β denotes the inverse of the noise variance.

2) Prior: As already mentioned, it is necessary to incorporate suitable prior knowledge on the sought quantities to limit the problem ill-posedness. We here consider a wide range of sparsity enhancing prior for the image $\mathbf{x}$, by adopting the generic model,

$$p(\mathbf{x}|\gamma) \propto \gamma^{\frac{N}{2\kappa}} \exp\Big(-\gamma \sum_{j=1}^{J} ||\mathbf{D}_j\mathbf{x}||^{2\kappa}\Big), \tag{8}$$

with $\kappa \in (0,1]$ a scale parameter and $(\mathbf{D}_j)_{1\leq j\leq J} \in (\mathbb{R}^{S\times N})^J$ both assumed to be known. For instance, an isotropic total variation prior is obtained by setting $\kappa = 1/2$, $S = 2$, $J = N$ and for every $j \in \{1,\dots,N\}$, $\mathbf{D}_j\mathbf{x} = [[\nabla^h\mathbf{x}]_j, [\nabla^v\mathbf{x}]_j] \in \mathbb{R}^2$ gathers the horizontal and vertical gradients of $\mathbf{x}$ at pixel j. Other relevant choices are discussed in [28]. Hereabove, $\gamma > 0$ is a regularization hyperparameter that we incorporate in our hierarchical model. We assume a Gamma distribution on γ,

$$p(\gamma) \propto \gamma^{\alpha-1}\exp(-\eta\gamma), \tag{9}$$

where $\alpha \geq 0$ and $\eta \geq 0$ are the (known) shape and inverse scale parameters of the Gamma distribution. Such choice for the hyperparameter is rather standard in the context of Bayesian image restoration.

Regarding the blur $\mathbf{h}$, we adopt the so-called SAR model, successfully used for Bayesian-based blind deconvolution in [30]. The model relies on the following Gaussian model,

$$p(\mathbf{h}|\xi) \propto \xi^{\frac{M}{2}} \exp\Big(-\frac{\xi}{2}||\mathbf{A}(\mathbf{h}-\mathbf{m})||^2\Big), \tag{10}$$

where $\mathbf{A} \in \mathbb{R}^{Q\times M}$ with $Q \in \mathbb{N}\setminus\{0\}$ denotes a matrix of rank M. $\mathbf{m} \in \mathbb{R}^M$ is the mean of the underlying Gaussian distribution, and $\xi > 0$ is such that $\xi\mathbf{A}^\top\mathbf{A}$ is its inverse covariance matrix. If $\mathbf{h}$ follows this distribution, the projection of $\mathbf{h}$ onto the affine space defined by (3) is also Gaussian as well as the vector $\mathbf{z}$ associated with each projected vector. More precisely, $\mathbf{z}$ follows a Gaussian distibution with mean $\boldsymbol{\mu} = \mathbf{T}^{-1}(\mathbf{m}-\mathbf{t})$ and covariance matrix $\xi^{-1}\mathbf{T}^{-1}(\mathbf{A}^\top\mathbf{A})^{-1}(\mathbf{T}^{-1})^\top$ where $\mathbf{T}^{-1}$ is the left inverse of $\mathbf{T}$, i.e. $\mathbf{T}^{-1} = (\mathbf{T}^\top\mathbf{T})^{-1}\mathbf{T}^\top$. This yields the following prior for the variable of interest $\mathbf{z}$:

$$p(\mathbf{z}|\xi) \propto \xi^{\frac{P}{2}} \exp\Big(-\frac{\xi}{2}(\mathbf{z}-\boldsymbol{\mu})^\top\mathbf{L}(\mathbf{z}-\boldsymbol{\mu})\Big), \tag{11}$$

where $\mathbf{L} = \mathbf{T}^\top\mathbf{T}(\mathbf{T}^\top\big(\mathbf{A}^\top\mathbf{A}\big)^{-1}\mathbf{T}\big)^{-1}\mathbf{T}^\top\mathbf{T}$. We will consider $(\mathbf{L},\boldsymbol{\mu})$ to be predefined by the user, so as to be adapted to the sought properties of the blur kernel to estimate. The hyperparameter ξ will be learned during a training phase, as we will explain in Section IV.

3) Hierarchical model: Let us assume that $(\mathbf{x},\gamma)$ and $\mathbf{z}$ are mutually independent. According to Bayes formula, the posterior distribution of the unknowns $\Theta = (\mathbf{x},\mathbf{z},\gamma)$ given the observed data $\mathbf{y}$ is defined as

$$p(\Theta|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x},\mathbf{z})p(\mathbf{x}|\gamma)p(\mathbf{z}|\xi)p(\gamma), \tag{12}$$

where the four factors on the right side have been defined above.

C. Variational Bayesian Inference

The Bayesian inference paradigm seeks for solving the blind restoration problem through the exploration of the posterior $p(\Theta|\mathbf{y})$. Typically, one would be interested in the posterior mean, its covariance, or its modes (i.e., maxima). Let us make (12) explicit:

$$\begin{aligned} p(\Theta|\mathbf{y}) \propto \exp\left(-\gamma\sum_{j=1}^{J}||\mathbf{D}_j\mathbf{x}||^{2\kappa} - \frac{\beta}{2}||\mathbf{y}-\mathcal{H}(\mathbf{z})\mathbf{x}||^2\right) \\ \times \gamma^{\frac{N}{2\kappa}+\alpha-1}\exp(-\eta\gamma)\xi^{\frac{P}{2}}\exp\Big(-\frac{\xi}{2}(\mathbf{z}-\boldsymbol{\mu})^\top\mathbf{L}(\mathbf{z}-\boldsymbol{\mu})\Big). \end{aligned} \tag{13}$$

Unfortunately, neither $p(\Theta|\mathbf{y})$, nor its moments (e.g., mean, covariance), nor its mode positions have a closed form. In particular $p(\mathbf{y})$, which acts as a normalization constant, cannot be calculated analytically. We thus resort to the variational Bayesian framework to approximate this distribution by a more tractable one, denoted by $q(\Theta)$, for which the estimators are easier to compute. The approximation is computed with the aim to minimize the Kullback-Leibler (KL) divergence between the target posterior and its approximation, which amounts to determining

$$\begin{aligned} q^{\text{opt}}(\Theta) &= \text{argmin}_q \ \mathcal{KL}(q(\Theta)||p(\Theta|\mathbf{y})), \\ &= \text{argmin}_q \int q(\Theta)\ln\left(\frac{q(\Theta)}{p(\Theta|\mathbf{y})}\right)\text{d}\Theta, \end{aligned} \tag{14}$$

where the equality holds only when $q(\Theta) = p(\Theta|\mathbf{y})$. In order to make the solution of the above minimization problem tractable, a typical strategy is to make use of a variational Bayesian algorithm (VBA) based on a so-called mean field

approximation of the posterior, combined with an alternating minimization procedure.

The mean field approximation reads as a factorized structure $q(\Theta) = \prod_{r=1}^{R} q_r(\Theta_r)$, which is assumed for the distribution q. Each of the R factors are then obtained by minimizing the KL divergence by iterative update of a given factor q_r while holding the others unchanged. This procedure takes advantage of the property that the minimizer of the KL divergence with respect to each factor can be expressed as

$$(\forall r \in \{1,...,R\}) \quad q_r^{\text{opt}}(\Theta_r) \propto \exp\left(< \ln p(\mathbf{y},\Theta) >_{\prod_{i\neq r} q_i^{\text{opt}}(\Theta_i)} \right) \tag{15}$$

where $< \cdot >_{\prod_{i\neq r} q_i(\Theta_i)} = \int \prod_{i\neq r} q_i(\Theta_i) \mathrm{d}\Theta_i$. Here, we will consider the following factorization:

$$q(\Theta) = q_{\mathbf{X}}(\mathbf{x}) q_{\mathbf{Z}}(\mathbf{z}) q_{\Gamma}(\gamma). \tag{16}$$

In Section III, we describe the steps of VBA for this particular choice. Due to the intricate form of the chosen prior on the image, we introduce an extra approximation step, relying on a majoration-minimization (MM) strategy, reminescent from [28]. In addition, we propose a strategy to reduce the time complexity of VBA, so as to deal with medium to large size images. As we will emphasize, the method requires the setting of two cumbersome hyperparameters, namely the regularization weight ξ and the noise level β. Then, in Section IV, we show how to unroll the VBA method as a neural network structure, so as to learn the parameters (ξ, β) in a supervised fashion.

III. VBA FOR BLIND IMAGE DECONVOLUTION

We now describe our proposed implementation of the VBA when applied to the approximation to the posterior in (13). We first present an MM-based procedure to handle the complicated form of the prior term on variable $\mathbf{x}$. Then, we give the explicit expressions of the updates performed in the alternating minimization method.

A. MM-based approximation

Let us focus on the prior term in (8). This distribution is difficult to deal with as soon as κ is different from 1 (in which case a Gaussian distribution is retrieved). We thus propose to construct a surrogate for the prior on $\mathbf{x}$. We use the tangent inequality for concave functions, which yields the following majorant function for the ℓ_κ-function with $\kappa \in (0,1]$:

$$(\forall u > 0)(\forall v \geq 0) \quad v^{\kappa} \leq (1-\kappa) u^{\kappa} + \kappa u^{\kappa-1} v. \tag{17}$$

Let us introduce the vector of auxiliary positive variables $\boldsymbol{\lambda} = (\lambda_j)_{1\leq j\leq J}$. From the previous inequality, we then deduce the following majorant function for the negative logarithm of the prior distribution:

$$(\forall \mathbf{x} \in \mathbb{R}^N) \quad \gamma \sum_{j=1}^{J} ||\mathbf{D}_j \mathbf{x}||^{2\kappa} \leq \sum_{j=1}^{J} F_j(\mathbf{D}_j \mathbf{x}, \lambda_j; \gamma), \tag{18}$$

where, for every $j \in \{1,\ldots,J\}$,

$$F_j(\mathbf{D}_j \mathbf{x}, \lambda_j; \gamma) = \gamma \frac{\kappa ||\mathbf{D}_j \mathbf{x}||^2 + (1-\kappa)\lambda_j}{\lambda_j^{1-\kappa}}. \tag{19}$$

This majorant function can be understood as a Gaussian lower bound on the prior distribution on $\mathbf{x}$, which will appear more tractable in the VBA implementation. We will also show that the update of the auxiliary variables remains rather simple, thus not impacting the complexity of the whole procedure.

In a nutshell, using (13), and (18), we obtain the following inequality:

$$p(\Theta|\mathbf{y}) \geq \mathcal{F}(\Theta|\mathbf{y};\boldsymbol{\lambda}) \tag{20}$$

where the lower bound on the posterior distribution is

$$\mathcal{F}(\Theta|\mathbf{y};\boldsymbol{\lambda}) = C\gamma^{\frac{N}{2\kappa}} \exp\left(-\frac{\beta}{2} ||\mathbf{y} - \mathcal{H}(\mathbf{z})\mathbf{x}||^2 - F(\mathbf{x},\boldsymbol{\lambda};\gamma) \right) p(\gamma) p(\mathbf{z}|\xi). \tag{21}$$

Hereabove we have introduced the shorter notation

$$F(\mathbf{x},\boldsymbol{\lambda};\gamma) = \sum_{j=1}^{J} F_j(\mathbf{D}_j \mathbf{x}, \lambda_j; \gamma) \tag{22}$$

and C is a multiplicative constant independent from Θ. Inequality (20) leads to the following majorization of the $\mathcal{KL}$ divergence involved in (14):

$$\mathcal{KL}(q(\Theta)||p(\Theta|\mathbf{y})) \leq \mathcal{KL}(q(\Theta)||\mathcal{F}(\Theta|\mathbf{y};\boldsymbol{\lambda})). \tag{23}$$

By minimizing the upper bound in (23) with respect to $\boldsymbol{\lambda}$, we can keep it as tight as possible, so as to guarantee the good performance of the VBA. To summarize, we propose to solve Problem (14) through the following four iterative steps:

1) Minimizing $\mathcal{KL}(q(\Theta)||\mathcal{F}(\Theta|\mathbf{y};\boldsymbol{\lambda}))$ w.r.t. $q_{\mathbf{X}}(\mathbf{x})$.
2) Minimizing the upper bound $\mathcal{KL}(q(\Theta)||\mathcal{F}(\Theta|\mathbf{y};\boldsymbol{\lambda}))$ in (23) w.r.t. $q_{\mathbf{Z}}(\mathbf{z})$.
3) Update the auxiliary variables $(\lambda_j)_{1\leq j\leq J}$ to minimize $\mathcal{KL}(q(\Theta)||\mathcal{F}(\Theta|\mathbf{y};\boldsymbol{\lambda}))$.
4) Minimizing $\mathcal{KL}(q(\Theta)||\mathcal{F}(\Theta|\mathbf{y};\boldsymbol{\lambda}))$ w.r.t. $q_{\Gamma}(\gamma)$.

Subsequently, at a given iteration k of the proposed algorithm, the corresponding estimated variables will be indexed by k.

B. VBA updates

Let us now describe the four steps of the proposed VBA, starting from a given iteration k associated with the current approximated distributions $q_{\mathbf{X}}^k(\mathbf{x}), q_{\mathbf{Z}}^k(\mathbf{z})$, and $q_{\Gamma}^k(\gamma)$, and the auxiliary parameter estimate $\boldsymbol{\lambda}^k$. We also denote by $(\mathbf{x}^k, \mathbf{z}^k, \gamma^k)$ the estimates of the means of $q_{\mathbf{X}}^k$, $q_{\mathbf{Z}}^k$, and q_{Γ}^k, and $(\mathbf{C}_{\mathbf{x}}^k, \mathbf{C}_{\mathbf{z}}^k)$ the covariance estimates for $q_{\mathbf{X}}^k$ and $q_{\mathbf{Z}}^k$.

1) Update of $q_{\boldsymbol{X}}(\boldsymbol{x})$: By definition,

$$q_{\mathbf{X}}^{k+1}(\mathbf{x}) = \text{argmin}_{q_{\mathbf{X}}} \mathcal{KL}(q_{\mathbf{X}}(\mathbf{x}) q_{\Gamma}^k(\gamma) q_{\mathbf{Z}}^k(\mathbf{z}) || \mathcal{F}(\Theta|\mathbf{y};\boldsymbol{\lambda}^k)). \tag{24}$$

The standard solution provided by (15) remains valid, by replacing the joint distribution by a lower bound chosen proportional to $\mathcal{F}(\Theta|\mathbf{y};\boldsymbol{\lambda}^k)$:

$$\begin{aligned} q_{\mathbf{x}}^{k+1}(\mathbf{x}) &\propto \exp\left(< \ln \mathcal{F}(\mathbf{x},\mathbf{z},\gamma \mid \mathbf{y};\boldsymbol{\lambda}^k) >_{q_{\Gamma}^k(\gamma), q_{\mathbf{Z}}^k(\mathbf{z})} \right) \\ &\propto \exp\left(\int\int \ln \mathcal{F}(\mathbf{x},\mathbf{z},\gamma \mid \mathbf{y};\boldsymbol{\lambda}^k) q_{\Gamma}^k(\gamma) q_{\mathbf{Z}}^k(\mathbf{z}) \mathrm{d}\gamma \mathrm{d}\mathbf{z} \right). \end{aligned} \tag{25}$$

By decomposing the different terms and using (4),

$$q_{\mathbf{x}}^{k+1}(\mathbf{x}) \propto \exp\Bigg\{ -\frac{1}{2}\mathbf{x}^\top \Bigg(\beta \Big(\mathbb{E}_{q_{\mathbf{Z}}^k(\mathbf{z})}(\mathbf{H})^\top \mathbb{E}_{q_{\mathbf{Z}}^k(\mathbf{z})}(\mathbf{H}) + \sum_{p=1}^{P}\sum_{q=1}^{P} \mathbf{e}_p^\top \mathrm{cov}_{q_{\mathbf{Z}}^k(\mathbf{z})}(\mathbf{z}) \mathbf{e}_q \mathbf{K}_p^\top \mathbf{K}_q \Big) + 2\mathbb{E}_{q_\Gamma^k(\gamma)}(\gamma)\mathbf{D}^\top \Lambda^k \mathbf{D} \Bigg) \mathbf{x} + \beta \mathbf{x}^\top \mathbb{E}_{q_{\mathbf{Z}}^k(\mathbf{z})}(\mathbf{H})^\top \mathbf{y} \Bigg\} \quad (26)$$

where

$$\mathbb{E}_{q_{\mathbf{Z}}^k(\mathbf{z})}(\mathbf{H}) = \sum_{p=1}^{P} \mathbf{e}_p^\top \mathbb{E}_{q_{\mathbf{Z}}^k(\mathbf{z})}(\mathbf{z}) \mathbf{K}_p + \mathbf{K}_0, \quad (27)$$

$$\mathbf{D} = [\mathbf{D}_1^\top, \ldots, \mathbf{D}_J^\top]^\top, \quad (28)$$

Λ^k is the block diagonal matrix whose diagonal elements are $(\kappa(\lambda_j^k)^{\kappa-1}\mathbf{I}_S)_{1\leq j\leq J}$, and $(\mathbf{e}_1, \ldots, \mathbf{e}_P)$ is the canonical basis of $\mathbb{R}^P$. We thus obtain a Gaussian distribution:

$$q_{\mathbf{X}}^{k+1}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \check{\mathbf{x}}^{k+1}, \check{\mathbf{C}}_{\mathbf{x}}^{\ k+1}), \quad (29)$$

parametrized by

$$(\check{\mathbf{C}}_{\mathbf{x}}^{\ k+1})^{-1} = \beta \Bigg((\mathbf{H}^k)^\top \mathbf{H}^k + \sum_{p=1}^{P}\sum_{q=1}^{P} \mathbf{e}_p^\top \mathbf{C}_{\mathbf{z}}^k \mathbf{e}_q \mathbf{K}_p^\top \mathbf{K}_q \Bigg) + 2\gamma^k \mathbf{D}^{\mathrm{T}} \mathbf{\Lambda}^k \mathbf{D}, \quad (30)$$

$$\check{\mathbf{x}}^{k+1} = \beta \check{\mathbf{C}}_{\mathbf{x}}^{\ k+1} (\mathbf{H}^k)^\top \mathbf{y}, \quad (31)$$

with $\mathbf{H}^k = \mathcal{H}(\mathbf{z}^k)$.

In image restoration applications, dimension N can be rather large (typically greater than 10^6 variables), so that the storage of the full covariance matrix $\check{\mathbf{C}}_{\mathbf{x}}^{\ k+1}$ is neither desirable nor usually possible. We thus propose to resort to a diagonal approximation to this matrix when required, so that the update finally reads:

$$q_{\mathbf{X}}^{k+1}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{x}^{k+1}, \mathbf{C}_{\mathbf{x}}^{k+1}), \quad (32)$$

with

$$\mathbf{C}_{\mathbf{x}}^{k+1} = \mathrm{Diag}\left(\delta_{\mathbf{x}}^{k+1}\right) \quad (33)$$

$$\mathbf{x}^{k+1} = \mathrm{CG}\Big((\check{\mathbf{C}}_{\mathbf{x}}^{\ k+1})^{-1}, \beta (\mathbf{H}^k)^\top \mathbf{y} \Big), \quad (34)$$

where $\delta_{\mathbf{x}}^{k+1} \in \mathbb{R}^N$ is the vector of the inverses of the diagonal elements of $(\check{\mathbf{C}}_{\mathbf{x}}^{\ k+1})^{-1}$, and $\mathrm{CG}(\mathtt{A}, \mathtt{b})$ denotes the application of a linear conjugate gradient solver to the linear system $\mathtt{Ax=b}$.

2) Update of $q_{\mathbf{Z}}(\mathbf{z})$: According to the VBA principle,

$$q_{\mathbf{Z}}^{k+1}(\mathbf{z}) = \mathrm{argmin}_{q_{\mathbf{Z}}} \mathcal{KL}(q_{\mathbf{X}}^{k+1}(\mathbf{x}) q_\Gamma^k(\gamma) q_{\mathbf{Z}}(\mathbf{z}) || \mathcal{F}(\Theta|\mathbf{y}; \boldsymbol{\lambda}^k)). \quad (35)$$

Using (15) and the previously introduced bound $\mathcal{F}(\Theta \mid \mathbf{y}; \boldsymbol{\lambda}^k)$, we have

$$q_{\mathbf{Z}}^{k+1}(\mathbf{z}) \propto \exp\left(\int\int \ln \mathcal{F}(\mathbf{x}, \mathbf{z}, \gamma \mid \mathbf{y}; \boldsymbol{\lambda}^k) q_\Gamma^k(\gamma) q_{\mathbf{X}}^{k+1}(\mathbf{x}) \mathrm{d}\gamma \mathrm{d}\mathbf{x} \right). \quad (36)$$

Replacing the involved quantities by their expression yields

$$q_{\mathbf{Z}}^{k+1}(\mathbf{z}) \propto \exp\Bigg\{ -\frac{1}{2}\mathbf{z}^\top \Big(\beta \mathbf{B}^{k+1} + \xi \mathbf{L} \Big) \mathbf{z} + \mathbf{z}^\top \Big(\beta \mathbf{a}^{k+1} + \xi \mathbf{L} \boldsymbol{\mu} \Big) \Bigg\}, \quad (37)$$

where $\mathbf{a}^{k+1} = (a_p^{k+1})_{1\leq p\leq P} \in \mathbb{R}^P$ and $\mathbf{B}^{k+1} = (B_{p,q}^{k+1})_{1\leq p,q\leq P} \in \mathbb{R}^{P\times P}$ are such that, for every $(p,q) \in \{1, \ldots, P\}^2$,

$$\begin{aligned} a_p^{k+1} &= \mathbb{E}_{q_{\mathbf{X}}^{k+1}(\mathbf{x})}(\mathbf{x})^\top \mathbf{K}_p^\top \mathbf{y} - \mathbb{E}_{q_{\mathbf{X}}^{k+1}}(\mathbf{x}^\top \mathbf{K}_p^\top \mathbf{K}_0 \mathbf{x}) \\ &= (\mathbf{x}^{k+1})^\top \mathbf{K}_p^\top \mathbf{y} - \mathbf{B}_{p,0}^{k+1}, \quad (38) \\ B_{p,q}^{k+1} &= \mathbb{E}_{q_{\mathbf{X}}^{k+1}}(\mathbf{x}^\top \mathbf{K}_p^\top \mathbf{K}_q \mathbf{x}) \\ &= \mathrm{trace}\left(\mathbf{K}_p \mathbf{C}_{\mathbf{x}}^{k+1} \mathbf{K}_q^\top \right) + (\mathbf{x}^{k+1})^\top \mathbf{K}_p^\top \mathbf{K}_q \mathbf{x}^{k+1} \quad (39) \end{aligned}$$

with

$$\begin{aligned} B_{p,0}^{k+1} &= \mathbb{E}_{q_{\mathbf{X}}^{k+1}}(\mathbf{x}^\top \mathbf{K}_p^\top \mathbf{K}_0 \mathbf{x}) \\ &= \mathrm{trace}\left(\mathbf{K}_p \mathbf{C}_{\mathbf{x}}^{k+1} \mathbf{K}_0^\top \right) + (\mathbf{x}^{k+1})^\top \mathbf{K}_p^\top \mathbf{K}_0 \mathbf{x}^{k+1}. \quad (40) \end{aligned}$$

Thus, the update for the distribution $q_{\mathbf{Z}}$ reads

$$q_{\mathbf{Z}}^{k+1}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{z}^{k+1}, \mathbf{C}_{\mathbf{z}}^{k+1}), \quad (41)$$

with

$$(\mathbf{C}_{\mathbf{z}}^{k+1})^{-1} = \beta \mathbf{B}^{k+1} + \xi \mathbf{L}, \quad (42)$$

$$\mathbf{z}^{k+1} = \mathbf{C}_{\mathbf{z}}^{k+1} \left(\beta \mathbf{a}^{k+1} + \xi \mathbf{L} \boldsymbol{\mu} \right). \quad (43)$$

3) Update of $\boldsymbol{\lambda}$: Let us now express the update of the auxiliary variable. We aim at finding

$$\boldsymbol{\lambda}^{k+1} = \mathrm{argmin}_{\boldsymbol{\lambda}} \mathcal{KL}(q_{\mathbf{X}}^{k+1}(\mathbf{x}) q_\Gamma^k(\gamma) q_{\mathbf{Z}}^{k+1}(\mathbf{z}) || \mathcal{F}(\Theta|\mathbf{y}; \boldsymbol{\lambda})). \quad (44)$$

This amounts to finding, for every $j \in \{1, \ldots, J\}$,

$$\begin{aligned} \lambda_j^{k+1} &= \mathrm{argmin}_{\lambda_j \in [0,+\infty)} \int q_{\mathbf{X}}^{k+1}(\mathbf{x}) q_\Gamma^k(\gamma) q_{\mathbf{Z}}^{k+1}(\mathbf{z}) \\ &\qquad \times \log \frac{q_{\mathbf{X}}^{k+1}(\mathbf{x}) q_\Gamma^k(\gamma) q_{\mathbf{Z}}^{k+1}(\mathbf{z})}{\mathcal{F}(\Theta|\mathbf{y}, \boldsymbol{\lambda})} \mathrm{d}\Theta, \\ &= \mathrm{argmin}_{\lambda_j \in [0,+\infty)} \sum_{i=1}^{J} \int\int q_{\mathbf{X}}^{k+1}(\mathbf{x}) q_\Gamma^k(\gamma) \\ &\qquad \times F_i(\mathbf{D}_i \mathbf{x}, \lambda_i; \gamma) \mathrm{d}\mathbf{x} \mathrm{d}\gamma, \\ &= \mathrm{argmin}_{\lambda_j \in [0,+\infty)} \frac{\kappa \mathbb{E}_{q_{\mathbf{x}}^{k+1}(\mathbf{x})} \left[||\mathbf{D}_j \mathbf{x}||^2 \right] + (1-\kappa)\lambda_j}{\lambda_j^{1-\kappa}}. \end{aligned} \quad (45)$$

The explicit solution to the above minimization problem yields the following update:

$$\begin{aligned} \lambda_j^{k+1} &= \mathbb{E}_{q_{\mathbf{x}}^{k+1}(\mathbf{x})} \left[||\mathbf{D}_j \mathbf{x}||^2 \right] \\ &= ||\mathbf{D}_j \mathbf{x}^{k+1}||^2 + \mathrm{trace}\left(\mathbf{D}_j^{\mathrm{T}} \mathbf{D}_j \mathbf{C}_{\mathbf{x}}^{k+1} \right). \quad (46) \end{aligned}$$

4) Update of $q_\Gamma(\gamma)$: Finally, the update related to the hyperparameter γ is expressed as

$$q_\Gamma^{k+1}(\gamma) = \text{argmin}_{q_\Gamma} \mathcal{KL}(q_{\mathbf{X}}^{k+1}(\mathbf{x}) q_\Gamma(\gamma) q_{\mathbf{Z}}^{k+1}(\mathbf{z}) || \mathcal{F}(\Theta|\mathbf{y}; \boldsymbol{\lambda}^{k+1})). \tag{47}$$

Using (15), we have

$$q_\Gamma^{k+1}(\gamma) \propto \exp\left(\int\int \ln \mathcal{F}(\mathbf{x}, \mathbf{z}, \gamma \,|\, \mathbf{y}; \boldsymbol{\lambda}^{k+1}) \times q_{\mathbf{X}}^{k+1}(\mathbf{x}) q_{\mathbf{Z}}^{k+1}(\mathbf{z}) \, \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z} \right). \tag{48}$$

The above integral has the following closed form expression:

$$q_\Gamma^{k+1}(\gamma) \propto \gamma^{\frac{N}{2\kappa}+\alpha-1} \exp(-\eta\gamma) \times \exp\left(-\gamma \sum_{j=1}^{J} \frac{\kappa \mathbb{E}_{q_{\mathbf{x}}^{k+1}(\mathbf{x})}\left[||\mathbf{D}_j \mathbf{x}||^2 \right] + (1-\kappa)\lambda_j^{k+1}}{(\lambda_j^{k+1})^{1-\kappa}} \right). \tag{49}$$

It thus follows from (46) that the update of q_Γ is

$$q_\Gamma^{k+1}(\gamma) = \Gamma(d, b^{k+1}), \tag{50}$$

that is the Gamma distribution with parameters

$$d = \frac{N}{2\kappa} + \alpha, \quad b^{k+1} = \sum_{j=1}^{J} (\lambda_j^{k+1})^\kappa + \eta. \tag{51}$$

The mean of q_Γ^{k+1} is finally given by

$$\gamma^{k+1} = \frac{d}{b^{k+1}}. \tag{52}$$

Note that parameter d is not iteration dependent and can thus be precomputed from the beginning of the VBA.

C. Overview of VBA

Algorithm 1 provides a summary of the resulting VBA for solving the blind deconvolution problem introduced in Section II. We also specify our initialization strategy. More practical details about the latter will be discussed in the experimental section. As a result, the optimal posterior distributions for both variables $\mathbf{x}$ and $\mathbf{z}$ will be approximated as Gaussian distributions, while the one for hyperparameter γ is approximated by a Gamma distribution. In particular, after K iterations, it is direct to extract from VBA outputs an estimate for the posterior mean of the image and the kernel, through variable $\mathbf{x}^K$ and $\mathbf{T}\mathbf{z}^K + \mathbf{t}$. The associated covariance matrices are given by $\mathbf{C}_{\mathbf{x}}^K$ and $\mathbf{T}\mathbf{C}_{\mathbf{z}}^K\mathbf{T}^\top$. These matrices can be useful to perform uncertainty quantification of the results. The VBA also allows us to estimate easily the hyperparameter γ involved in the image prior. Nonetheless, it appears difficult to find an efficient manner to estimate the hyperparameter ξ using a variational Bayesian approach, as this value highly fluctuates from one image/kernel pair to the other so that a simple prior modeling of does not appear obvious. Moreover, the VBA requires the knowledge of the noise level, through the parameter β. This is limitating, and one might prefer to have this quantity estimated in an automatic manner. Thus, we propose in the next section, to resort to a supervised learning strategy to learn both ξ and β along the iterates of VBA, in the spirit of recent works [43] on the unrolling (also called unfolding) of iterative algorithms.

Algorithm 1 VBA approach for image blind deconvolution

Initialization. Set hyperparameters $(\xi, \beta, \alpha, \eta)$. Define initial values for $(\mathbf{x}^0, \mathbf{C}_{\mathbf{x}}^0, \mathbf{z}^0, \mathbf{C}_{\mathbf{z}}^0)$. Compute $\boldsymbol{\lambda}^0$ and γ^0 using (46) and (52), respectively.

Iterative steps. For $k = 0, 1, \ldots, K$:

1: Update the mean $\mathbf{x}^{k+1}$ and the covariance matrix $\mathbf{C}_{\mathbf{x}}^{k+1}$ of $q_{\mathbf{X}}^{k+1}(\mathbf{x})$ using (33)-(34).
2: Update the mean $\mathbf{z}^{k+1}$ and the covariance matrix $\mathbf{C}_{\mathbf{z}}^{k+1}$ of $q_{\mathbf{Z}}^{k+1}(\mathbf{z})$ using (42)-(43).
3: Update λ_j^{k+1} using (46), for every $j \in \{1, \ldots, J\}$.
4: Update the mean γ^{k+1} of $q_\Gamma^{k+1}(\gamma)$ using (51)-(52).

IV. Supervised learning of VBA hyperparameters

A. Overview

We introduce a supervised learning strategy to estimate the hyperparameter ξ and the inverse of the noise variance β, that are required to run VBA. We adopt the so-called *unrolling* (or unfolding) methodology [41]. The idea is to view each iteration of an iterative algorithm as one layer of a neural network structure. Each layer can be parametrized by some quantities that are learned from a training database so as to minimize a task-oriented loss function. The advantage of the unrolling approach is threefold: (i) each layer mimics one iteration of the algorithm and thus it is highly interpretable, (ii) the choice of the loss is directly related to the task at the end, which is beneficial for the quality of the results, (iii) once trained, the network can be applied easily and rapidly on a large set of test data without any further tuning. In particular, its implementation can make use of GPU-accelerated frameworks. Several recent examples in the field of image processing have shown the benefits of unrolling [54], [55], [56], [57] when compared to standard black-box deep learning techniques or more classical restoration methods based on Bayesian or optimization tools. Let us in particular mention the works [42], [58] for the application of unrolling in the context of blind image restoration.

Let us now specify the unrolling procedure in the context of VBA. Let $K > 0$ be the number of iterations of the VBA described in Algorithm III-C, thus corresponding to K layers of a neural network architecture. Iteration $k \in \{0, \ldots, K-1\}$ of our unrolled VBA can be conceptually expressed as

$$\begin{aligned} &(\mathbf{x}^{k+1}, \mathbf{C}_{\mathbf{x}}^{k+1}, \mathbf{z}^{k+1}, \mathbf{C}_{\mathbf{z}}^{k+1}, \boldsymbol{\lambda}^{k+1}, \gamma^{k+1}) \\ &= \mathcal{A}(\mathbf{x}^k, \mathbf{C}_{\mathbf{x}}^k, \mathbf{z}^k, \mathbf{C}_{\mathbf{z}}^k, \boldsymbol{\lambda}^k, \gamma^k, \xi^k, \beta^k). \end{aligned} \tag{53}$$

The initialization procedure for $(\mathbf{x}^0, \mathbf{C}_{\mathbf{x}}^0, \mathbf{z}^0, \mathbf{C}_{\mathbf{z}}^0, \boldsymbol{\lambda}^0, \gamma^0)$ is detailed in Algorithm III-C. For $k \in \{0, \ldots, K-1\}$, the expressions of $(\mathbf{x}^{k+1}, \mathbf{C}_{\mathbf{x}}^{k+1}, \mathbf{z}^{k+1}, \mathbf{C}_{\mathbf{z}}^{k+1}, \gamma^{k+1}, \boldsymbol{\lambda}^{k+1})$ as a function of the input arguments of $\mathcal{A}(\cdot)$ are given respectively by (33)-(34), (42)-(43), (46), and (51)-(52). Furthermore, $(\xi^k, \beta^k)_{0 \le k \le K-1}$ are now learned, instead of being constant and preset by the user. This leads to the *unfoldedVBA* architecture depicted in Fig. 1, which can be summarized into the

composition of K layers $\mathcal{L}_{K-1} \circ \cdots \circ \mathcal{L}_0$. Each layer $\mathcal{L}_k$ with $k \in \{0,\dots,K-1\}$ is made of three main blocks, that are two neural networks, namely NN_σ^k and NN_ξ^k, and the core VBA block $\mathcal{A}(\cdot)$. There remains to specify our strategy for building the two inner networks, with the aim to learn $(\xi^k, \beta^k)_{0\le k\le K-1}$.

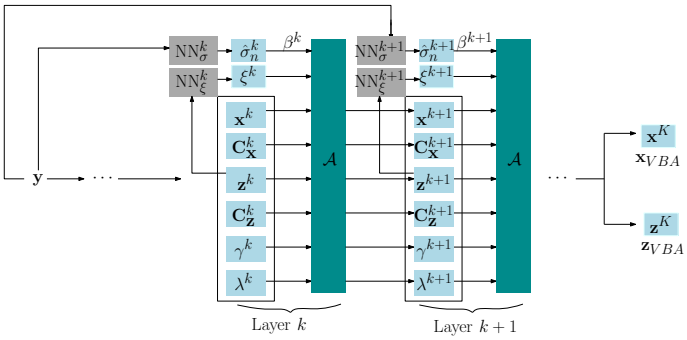

Fig. 1. Architecture of *unfoldedVBA* network.

B. Learning hyperparameter ξ

For every $k \in \{0,...,K-1\}$, neural network NN_ξ^k takes as input the current kernel estimate $\mathbf{h}^k = \mathbf{T}\mathbf{z}^k + \mathbf{t}$ and delivers ξ^k as an output. The architecture of the neural network is shown in Fig. 2. Note that the Softplus function, defined as

$$(\forall x \in \mathbb{R}) \quad \mathrm{Softplus}(x) = \ln(1+\exp(x)), \tag{54}$$

is used as a last layer, in order to enforce the strict positivity of the output hyperparameter ξ^k.

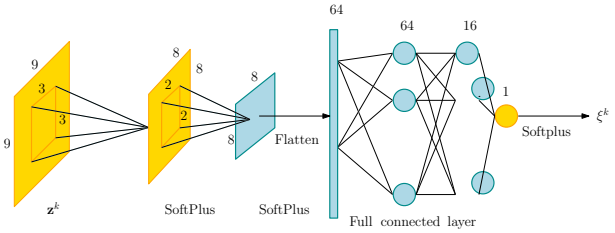

Fig. 2. Neural network architecture NN_ξ^k for estimating ξ^k, for $k \in \{0,...,K-1\}$.

C. Learning noise parameter β

When the noise parameter β is unknown, it might be useful to include a procedure to learn it automatically, again in a supervised fashion. In this case, we propose to introduce simple nonlinear mappings such that, for every $k \in \{0,\dots,K-1\}$,

$$\begin{aligned}\sigma^k &= \mathrm{NN}_\sigma^k(\mathbf{y}),\\ &= \mathrm{Softplus}(\rho_k)\widehat{\sigma}(\mathbf{y}) + \mathrm{Softplus}(\tau_k),\end{aligned} \tag{55}$$

and $\beta^k = (\sigma^k)^{-2}$. Hereabove, $\mathbf{y}$ is the observed degraded image, from which we deduce the wavelet-based variance estimator (also used in [43]),

$$\widehat{\sigma}(\mathbf{y}) = \frac{\mathrm{median}(|\mathbf{W}_H\mathbf{y}|)}{0.6745}, \tag{56}$$

where $|\mathbf{W}_H\mathbf{y}|$ gathers the absolute value of the diagonal coefficients of the first level Haar wavelet decomposition of the degraded image $\mathbf{y}$. Moreover, $(\rho_k, \tau_k)_{0\le k\le K-1}$ are two scalar parameters to be learned during the training phase.

D. Complete architecture

We now present our complete blind deconvolution architecture for grayscale images and color images in Fig. 3. First, let us notice that VBA and its unrolled variant is designed for grayscale images. We thus generalized the architecture from Fig. 3(top), to process color images. To this end, we first transform the input RGB image to its YUV representation, which takes human perception into consideration. The network NN_σ^k is first applied to the luminance part $\mathbf{y}_Y$ of the image. After applying the *unfoldedVBA* network (see Fig. 1), we obtain $\mathbf{z}_{\mathrm{VBA}}$ and $\mathbf{x}_{\mathrm{VBA}}$ as outputs. The latter is a restored version of the luminance channel. The remaining (U,V) color channels are simply obtained by median filtering of $(\mathbf{y}_U, \mathbf{y}_V)$. Both architectures in Fig. 3 additionally involve post-processing layers. More precisely, we first include a linear layer so as to encode the linear transformation (3), and then deduce the estimated blur kernel $\widehat{\mathbf{h}}$. Second, we also allow a post-processing layer $\mathcal{L}_{\mathrm{pp}}$ acting on the image, so as to reduce possible residual artifacts, finally yielding $\widehat{\mathbf{x}}$. In the case of color images, the post-processing is applied on the RGB representation to avoid chromatic artifacts.

E. Training procedure

The training of both proposed architectures from Fig. 3 requires to define a loss function, measuring the discrepancy between the output $(\widehat{\mathbf{x}}, \widehat{\mathbf{h}})$ and the ground truth $(\overline{\mathbf{x}}, \overline{\mathbf{h}})$, that we denote hereafter by $\ell(\widehat{\mathbf{x}}, \widehat{\mathbf{h}}, \overline{\mathbf{x}}, \overline{\mathbf{h}})$. In the blind deconvolution application, one can for instance consider a loss function related to the error reconstruction on the kernel, or to the image quality, or a combination of both. Two training procedures will be distinguished and discussed in our experimental section, namely:

Greedy training The parameters of the unfolded VBA are learned in a greedy fashion so as to minimize the kernel reconstruction error at each layer. Then, the post-processing network is learned in a second step, so as to maximize an image quality metric such as the SSIM [59].

End-to-end training The parameters of the complete architecture are learned end-to-end so as to maximize the image quality metric.

Whatever the chosen training procedure, it is necessary to make use of a back-propagation step, that is to differentiate the loss function with respect to all the parameters of the network. Most operations involved in Fig. 3 can be differentiated efficiently using standard auto-differentiation tools. However, we observed in our experiments that it is beneficial (and sometimes even necessary) for a stable training phase to avoid using such tools for differentiating the VBA layer $\mathcal{A}(\cdot)$ involved in Fig. 1. In practice, we used the explicit expressions for the partial derivatives of it. Note that we followed the approach from [60] to obtain the expression of the derivatives for the CG solver.

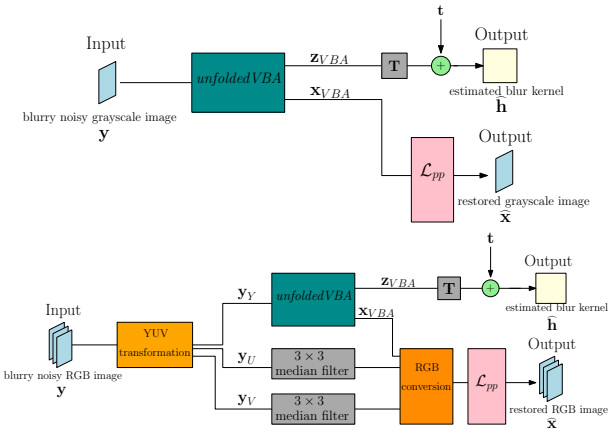

Fig. 3. Proposed blind image restoration pipeline for grayscale (top) and color (bottom) images.

V. Numerical experiments

A. Problem formulation and settings

1) Problem overview: We focus on the resolution of the blind image deconvolution problem, where $\widetilde{\mathbf{x}} \in \mathbb{R}^N$ is an original image, either grayscale or colored one. We come back to the model presented in Section II-A, where the linear operator $\widetilde{\mathbf{H}} \in \mathbb{R}^{N \times N}$ models the application of a blur kernel $\widetilde{\mathbf{h}} \in \mathbb{R}^M$ to the image. The noise $\mathbf{n}$ is assumed to be an additive white Gaussian noise with zero mean and standard deviation σ. In the case of color images, we assume that the same kernel, and the same noise level, is applied to each of the three RGB channels.

2) Datasets: Let us introduce the three datasets used to train and test our network, and compare our approach with state-of-the-art techniques. In all the cases, the training set is made of 100 images from the COCO training set. The validation set contains 40 images from the BSD500 validation set. For *Dataset 1* and *Dataset 2*, the test set consists of 30 images from the Flickr30 test set. For *Dataset 3*, we adopted the same testing strategy as in [36], using the 4 images from [5]. Each image is center-cropped using a window of size $N = 256 \times 256$. Each original image $\widetilde{\mathbf{x}}$ is associated to a degraded version of it, $\mathbf{y}$, built from Model (1). Various blur kernels and noise levels are used, as detailed hereafter.

Dataset 1: All involved images are converted in grayscale. Each image of the database is blurred with 10 randomly generated Gaussian blurs, and then corrupted by additive noise. Thus in total, we have 1000 ($= 100 \times 10$) training images, 400 ($= 40 \times 10$) validation images, and 300 ($= 30 \times 10$) test images for Dataset 1. The Gaussian blurs are of size 9×9. Two of them are isotropic with standard deviation randomly generated following a uniform law within $[0.2, 0.4]$. Eight of them are anisotropic with orientation either $\pi/4$ or $3\pi/4$ (with equal probability) and vertical/horizontal widths (i.e., standard deviations of the 2D Gaussian shape) uniformly drawn within $[0.15, 0.4]$. On this dataset, the noise standard deviation is set to $\sigma = 0.01$, and assumed to be known (so that blocks $(\mathrm{NN}_\sigma^k)_{1 \leq k \leq K}$ of our architecture are overlooked).

Dataset 2: All the images are then colored ones. We degraded each of them with 15 different blurs, namely 10 Gaussian blurs (simulated using the same procedure as above), two uniform blurs with width 5×5 and 7×7 pixels, and 3 out-of-focus blurs. For the latter, the vertical and horizontal widths are set randomly within $[0.2, 0.5]$, and the orientation is either $\pi/4$ or $3\pi/4$ (with equal probability). Furthermore, for each blurred image, zero-mean Gaussian noise is added, with standard deviation σ randomly chosen, with uniform distribution over $[0.005, 0.05]$. The noise level is not assumed to be known and is estimated using the proposed NN_σ architecture. In total, we have 1500 ($= 100 \times 15$) training images, 600 ($= 40 \times 15$) validation images, and 450 ($= 30 \times 15$) test images, on this dataset. Examples of blurs involved in *Dataset 2* are depicted in Fig. 4.

Dataset 3: All the images are converted in grayscale. We degraded each of them with 8 different real-world motion blurs from[1] [5]. As the kernels have different sizes ranging from 13×13 to 27×27, we pad them with 0 values, so to reach a common large size of 27×27. For each blurred image, zero-mean Gaussian noise is added, with standard deviation σ randomly chosen, following a uniform distribution within $[5 \times 10^{-6}, 5 \times 10^{-5}]$. The noise level is not assumed to be known and is estimated using the proposed NN_σ architecture. In total, we have 800 ($= 100 \times 8$) training images, 320 ($= 40 \times 8$) validation images, and 32 ($= 4 \times 8$) test images, on this dataset. The motion blurs involved in *Dataset 3* are illustrated in Fig. 5.

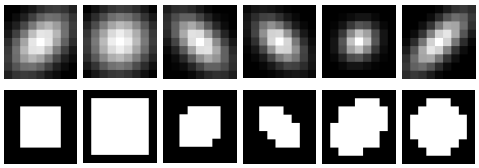

Fig. 4. Examples of blur kernels used to construct *Dataset 2*.

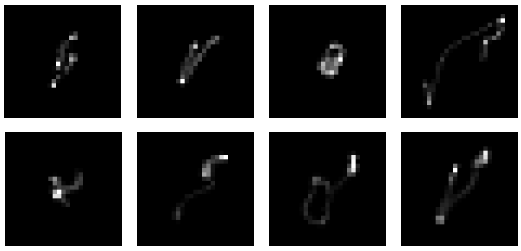

Fig. 5. Motion blur kernels from *Dataset 3*.

3) VBA settings: Different settings are adopted for the implementation of the VBA step on each dataset, that we specified hereafter. For all considered datasets, we seek for kernels whose entries satisfy a sum-to-one constraint. For the datasets involving synthetic blurs (namely, *Dataset 1* and *Dataset 2*), we impose additionally an axial symmetry along

[1] https://webee.technion.ac.il/people/anat.levin/

the main diagonal axis. In all cases, the constraints can be easily translated into the affine constraint (3). For *Dataset 1* and *Dataset 2*, the degree of freedom of the kernel model is equal to $P = \frac{(\sqrt{M}+1)\sqrt{M}}{2} - 1$, while we have $P = M - 1$ for *Dataset 3*. Regarding the choice of the blur prior, hereagain, a distinction is made between the first two scenarios and the last one. The synthetic blurs from *Dataset 1* and *Dataset 2* are either very smooth (e.g., Gaussian blurs) or present large constant areas (e.g., defocus blurs). In order to incorporate such prior knowledge, for both these datasets, $\mathbf{A} \in \mathbb{R}^{(2M+1)\times M}$ is set to the matrix that computes the horizontal and vertical differences between pixels, augmented with an additional first row corresponding to an averaging operation, i.e. $[1,\ldots,1]/M$. This choice allows us to promote smooth variations in the kernels, while satisfying the required full column rank assumption on $\mathbf{A}$. A constant vector with entries equal to $\frac{1}{M}$ is set for the prior mean $\mathbf{m}$. In contrast, the motion blurs of *Dataset 3* depicted in Fig. 5 present complex shapes, with no specific property on their gradient. We thus simply set $\mathbf{A}$ to the identity matrix in the prior, while a rough estimation of the blur obtained from [2] [10] is used for $\mathbf{m}$. Matrices $(\mathbf{D}_j)_{1\leq j\leq J}$ and parameter κ, involved in (8), are set in such a way that the chosen prior on the image yields an isotropic total-variation regularization (see our comment in Sec. II-B2). We must now specify the initialization for VBA iterates/layers. Our initial guess $\mathbf{x}^0$ for the image is the degraded one. The associated covariance matrix $\mathbf{C}_\mathbf{x}^0$ is initialized using the identity matrix. For *Dataset 1* and *Dataset 2*, the blur is initialized with a 5×5 uniform kernel, from which we deduce the corresponding $\mathbf{z}^0$, while we use the prior mean for initializing the method for *Dataset 3*. In all VBA experiments, the covariance matrix $\mathbf{C}_\mathbf{z}^0$ is set to a multiple of identity matrix. As the blurs involved in the *Dataset 3* are of large size, a diagonal approximation of $\mathbf{C}_\mathbf{z}^k$ is used in (30) so as to cope with memory saturation issues. A similar strategy was adopted for instance in [28] and was not observed to yield any degradation of the resulting quality. The hyperparameters (α,η) involved in the prior law on parameter γ are set to zero in practice which is equivalent to impose a non-informative Jeffrey improper prior. Finally, the conjugate gradient solver used for the update of the image is run over 10 iterations which appears sufficient to reach practical stability. The solver is initialized with the degraded image.

4) Training specifications: We present results obtained by adopting the two training strategies described in Section IV-E. For the *greedy training*, we make use of the mean squared error on the estimated kernel, as a loss function for the *unfoldedVBA* layers, defined as $\ell(\widehat{\mathbf{x}},\widehat{\mathbf{h}},\widetilde{\mathbf{x}},\widetilde{\mathbf{h}}) = \|\widetilde{\mathbf{h}} - \widehat{\mathbf{h}}\|^2$. The SSIM loss ([59]), between the output image $\widehat{\mathbf{x}}$ and the ground truth $\widetilde{\mathbf{x}}$ is used to train the post-processing layer $\mathcal{L}_{\text{pp}}$. For the *end-to-end training*, we use again SSIM between $\widehat{\mathbf{x}}$ and $\widetilde{\mathbf{x}}$. We use warm initialization for *end-to-end training*, that is we initialize with the weights learnt during the *greedy training* phase, associated with a weight decay procedure. ADAM optimizer, with mini-batch size equal to 10, is used for all the training procedures. Its parameters such as learning rate (lr), weight decay (wd) and epochs number are finetuned, so as to obtain stable performance on each validation set. The number of layers K (i.e., number of VBA iterations) is set during the *greedy training*, and kept the same for the *end-to-end training*. In practice, we increase K as long as a significant decrease in the averaged MSE over the training set was observed. We specify in Table I all the retained settings. The train/validation/test phase are conducted with a code implemented in Pytorch (version 1.7.0) under Python (version 3.6.10) environment, and run on an Nvidia DGX wokstation using one Tesla V100 SXM2 GPU (1290 MHz frequency, 32GB of RAM). Our code is made available at https://github.com/yunshihuang/unfoldedVBA.

	Dataset 1	Dataset 2	Dataset 3
Greedy training	*UnfoldedVBA*		
	$K = 6$, epoch $= 10$ lr $= 5\times 10^{-3}$	$K = 21$, epoch $= 10$ lr $= 5\times 10^{-3}$ (for $\mathcal{L}_0$) lr $= 10^{-3}$ (for other layers)	$K = 10$, epoch $= 10$ lr $= 10^{-4}$ (for $\mathcal{L}_0$) lr $= 10^{-3}$ (for other layers)
	Post-processing $\mathcal{L}_{pp}$		
	U-net [61] epoch $= 30$, lr $= 10^{-3}$	Residual network [43, Fig.4] epoch $= 200$, lr $= 10^{-3}$	U-net [61] epoch $= 200$, lr $= 10^{-3}$
End-to-end training	$K = 6$ epoch $= 6$ lr $= 5\times 10^{-5}$ wd $= 10^{-4}$	$K = 21$ epoch $= 6$ lr $= 5\times 10^{-5}$ wd $= 10^{-4}$	$K = 10$ epoch $= 6$ lr $= 5\times 10^{-5}$ wd $= 10^{-4}$

TABLE I
SETTINGS FOR THE TRAINING PHASES IN OUR EXPERIMENTS

5) Comparison to other methods: The proposed method is compared to several blind deconvolution approaches available in the literature:

Optimization-based methods: We first evaluate the VBA described in Section III, in the favorable situation where the noise level σ is assumed to be known, and parameter ξ is finetuned empirically (see more details hereafter). VBA is run until reaching practical convergence, i.e. when the relative squared distance between two consecutive image iterates gets lower than 10^{-5}. We also test two optimization-based approaches for blind deconvolution. The first one is called *deconv2D*. It makes use of the proximal alternating algorithm from [17], to minimize a least-squares data fidelity term combined with various priors, namely total variation and positivity constraint on the image, sum-to-one and quadratic constraint on the kernel. This method is implemented in Matlab, and inherits some of the software accelerations discussed in [16] for blind video deconvolution. The second competitor in this category is the *blinddeconv* approach [3] from [11]. For the sake of fair comparisons, for both datasets, we finetune the hyperparameters of these three methods on 40% of the training set and apply an average of the found values on the test set. Moreover, following the use of these three methods, we perform a non-blind deconvolution step BM3D-DEB [4] [62], which uses their respective estimated blur kernel to restore the image.

Deep learning methods: We perform comparisons with three recent deep learning architectures for blind deconvolution.

[2]http://zoi.utia.cas.cz/deconv_sparsegrad

[3]Matlab code: https://dilipkay.wordpress.com/blind-deconvolution/

[4]Matlab code: https://webpages.tuni.fi/foi/GCF-BM3D/index.html#ref_software

SelfDeblur [5] [36] is an unsupervised approach able to jointly perform the image restoration and kernel estimation tasks. DBSRCNN [6] [35] and DeblurGAN [7] [33] are two supervised deep learning techniques. In contrast with SelfDeblur, they both only provide the estimated image, but do not estimate the kernel. Both these methods have been retrained on each of our datasets, using the same settings as in their initial implementation. Moreover, we adapted DBSRCNN to color images using the same pipeline as for our method, that is applying DBSRCNN on the luminance channel while simply nonlinearly filtering the chrominance ones. Finally, the kernels provided as outputs of the SelfDeblur method are normalized as a post-processing step to satisfy the sum-to-one constraint, to make the comparisons more faithful.

6) Evaluation metrics: All the methods are evaluated in terms of their performance on the blur kernel estimation (when available) and on the image restoration. Different metrics are used. For the blur kernels, we evaluate (*i*) the MSE, (*ii*) the so-called $\mathcal{H}_\infty$ error defined as the ℓ_∞ norm of the difference between the 2D discrete Fourier coefficients (with suitable padding) of the estimated and of the true kernel, and (*iii*) the mean absolute error (MAE) defined as the ℓ_1 norm of the difference between $\widetilde{\mathbf{h}}$ and $\widehat{\mathbf{h}}$. For evaluating the image quality, we compute (*i*) the SSIM, (*ii*) the PSNR (Peak-Signal-to-Noise Ratio), and (*iii*) the PieAPP value [63], between the estimated image $\widehat{\mathbf{x}}$ and the ground truth $\widetilde{\mathbf{x}}$.

B. Experimental results

1) Dataset 1: In Tables II and III, we report the results of kernel estimation and image restoration, computed on the test set, using the different methods. As could be expected, the greedy approach tends to give more weight to the kernel quality than the end-to-end training. Our two training approaches yield great performance, when compared to all the other tested approaches. One can notice that the VBA with finetuned value for ξ performs quite well, showing the validity of our Bayesian formulation. The proposed unrolled VBA technique allows us to avoid a manual tuning of this parameter, and further increases the resulting quality. This is a direct outcome of the supervised training procedure aiming at maximizing quality scores, and also to the introduction of a post-processing step on the images. DBSRCNN shows fair performance in terms of image quality in this dataset, especially in terms of PSNR. Here, we must emphasize that this method uses MSE as a loss function during training, which tends to favor high PSNR scores. However, DBSRCNN is not capable of estimating the blur kernel, which might be useful for various applications. We display two examples of results in Fig 7, extracted from our test set. One can notice, by visual inspection of these results, the high quality of the restored images. No artifacts can be observed, which is confirmed by a low average value of the PieAPP index on the test set. Moreover, the kernels are generally estimated quite accurately, as shown by the low

[5]Python/Pytorch code: https://github.com/csdwren/SelfDeblur

[6]Python/Pytorch code: https://github.com/Fatma-ALbluwi/DBSRCNN

[7]Python/Pytorch (training) and Matlab C-mex (testing) codes: https://github.com/KupynOrest/DeblurGAN

Method	MSE	$\mathcal{H}_\infty$ error	MAE
VBA	0.0017 (0.0022)	0.1674 (0.1003)	0.0472 (0.0317)
deconv2D	0.0025 (0.0035)	0.1483 (0.1037)	0.0489 (0.0395)
blinddeconv	0.0013 (0.0011)	0.1553 (0.0660)	0.0417 (0.0203)
SelfDeblur	0.0143 (0.0181)	0.3253 (0.1105)	0.1350 (0.0663)
Proposed (greedy)	**0.0008** (0.0012)	**0.1165** (0.0677)	**0.0281** (0.0168)
Proposed (end-to-end)	0.0009 (0.0013)	0.1188 (0.0672)	0.0289 (0.0170)

TABLE II
QUANTITATIVE ASSESSMENT OF THE RESTORED KERNELS. MEAN (STANDARD DEVIATION) VALUES COMPUTED OVER THE TEST SETS OF *Dataset 1*.

MSE score and the good retrieval of their general structure. In the few cases when the unfolded VBA algorithm fails to give a perfect recovery of the blur kernel as in Fig. 7(bottom), the estimation is still accurate enough to yield a good recovery of the image whatever *greedy training* or *end-to-end training* is used. One can also notice that our method tends to provide better contrasted images, compared to its closest competitor in the image restoration task that is DBSRCNN. We display in Fig. 6(left) the evolution of the SSIM loss during the end-to-end training of the proposed architecture, showing the increase of the loss, then its stabilization, for both training and validation set, thus confirming an appropriate setting of ADAM optimizer parameters. Finally, Table IV(left) displays the average test time for each methods, that is the computational time required to restore one example of the dataset, once the method is finetuned/trained. We displayed CPU time for a fair comparison between methods, for codes ran on a Dell workstation equipped with an Xeon(R) W-2135 processor (3.7 GHz clock frequency and 12 GB of RAM). GPU time is also indicated when available. The fastest method is DBSRCNN, though we must emphasize that, in contrast with all the other methods based on Matlab/Python softwares, DBSRCNN makes use of an optimized C implementation, for its test phase on CPU. DeblurGAN is also very fast, but the resulting quality was quite poor in our experiments. Our method reaches a reasonable computational time on CPU. It becomes quite competitive when making use of GPU implementation, as the unrolled architecture is well suited for that purpose. This allows to drop the test time per image to few seconds, making it advantageous, with the addition benefit of better quality results in average, and of an available kernel estimate.

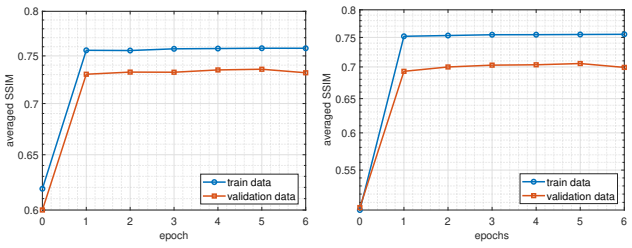

Fig. 6. Evolution of SSIM loss along epochs of *end-to-end training* phase, averaged either on training or on validation sets of *Dataset 1* (left) and *Dataset 2* (right).

2) Dataset 2: The results of kernel estimation and image restoration on Dataset 2 using the various methods are shown in Tables V and VI, respectively. This dataset is more

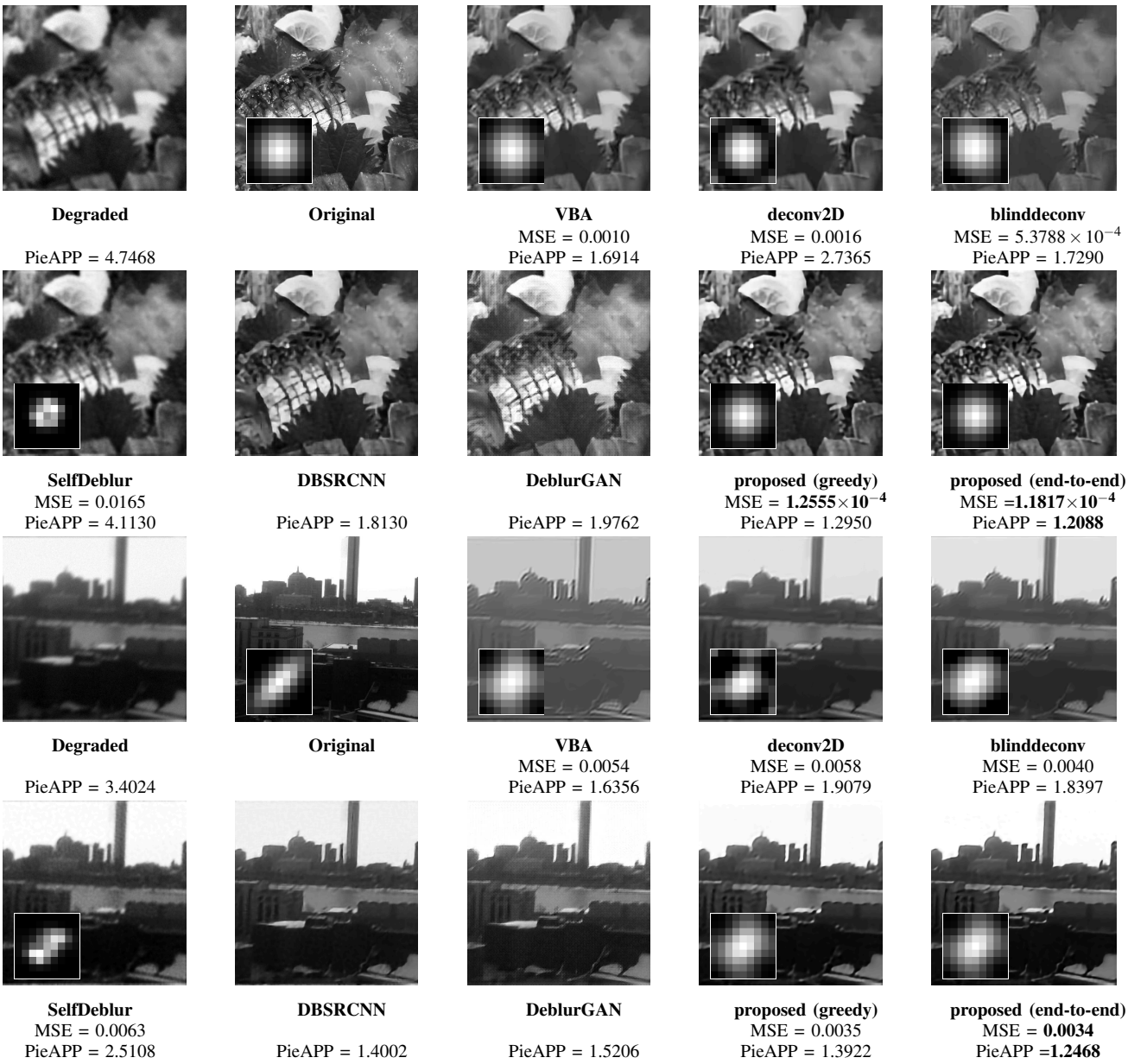

Fig. 7. Ground-truth image/blur, degraded image, restored images (with PieAPP index) and estimated blurs (with MSE score) when available, for various methods, on two examples in the test set of *Dataset 1*.

Method	SSIM	PSNR	PieAPP
Blurred	0.6542 (0.1072)	22.2254 (2.3779)	4.1794 (0.9005)
VBA	0.7603 (0.0814)	23.7332 (2.5672)	1.5109 (0.6184)
deconv2D	0.7668 (0.0912)	24.5970 (2.8656)	1.9289 (0.4959)
blinddeconv	0.7528 (0.0963)	23.9347 (2.4299)	1.9170 (0.6630)
SelfDeblur	0.6948 (0.1006)	22.2704 (2.1255)	3.3178 (0.7291)
DBSRCNN	0.7780 (0.0895)	**24.9561** (2.9800)	1.5959 (0.6463)
DeblurGAN	0.6613 (0.0731)	22.4388 (2.4074)	1.8937 (0.7630)
Proposed (greedy)	0.7945 (0.0890)	**24.7093** (2.9351)	1.4047 (0.6437)
Proposed (end-to-end)	**0.7989** (0.0886)	**24.6638** (3.0711)	**1.1976** (0.5433)

TABLE III
QUANTITATIVE ASSESSMENT OF THE RESTORED IMAGES. MEAN (STANDARD DEVIATION) VALUES COMPUTED OVER THE TEST SETS OF *Dataset 1*.

Method	Dataset 1	Dataset 2	Dataset 3
VBA	153s (15s)	156s (18s)	606s (62s)
deconv2D	16s	19s	333s
blinddeconv	19s	22s	188s
SelfDeblur	452s (51s)	455s (55s)	2232s (279s)
DBSRCNN	1s	2s	1s
DeblurGAN	2s (1s)	3s (2s)	2s (1s)
Proposed	36s (4s)	113s (12s)	286s (31s)

TABLE IV
AVERAGE TEST TIME PER IMAGE, USING CPU (RESP. GPU).

challenging than the former one, as it includes color images, various blur shapes, and various noise levels. The latter are not assumed to be known anymore. Hereagain, we can observe that the *greedy training* yields the best performance in terms of

Method	MSE	$\mathcal{H}_\infty$ error	MAE
VBA	0.0148 (0.0139)	0.4492 (0.1638)	0.1339 (0.0627)
deconv2D	0.0099 (0.0160)	0.2796 (0.1692)	0.0869 (0.0576)
blinddeconv	0.0245 (0.0264)	0.3113 (0.1409)	0.1596 (0.1106)
SelfDeblur	0.4336 (0.3188)	0.8720 (0.2274)	0.6975 (0.3047)
Proposed (greedy)	**0.0037** (0.0079)	**0.1888** (0.1061)	**0.0570** (0.0414)
Proposed (end-to-end)	0.0039 (0.0079)	0.1960 (0.1056)	0.0588 (0.0411)

TABLE V
QUANTITATIVE ASSESSMENT OF THE RESTORED KERNELS. MEAN (STANDARD DEVIATION) VALUES COMPUTED OVER THE TEST SETS OF *Dataset 2*.

Method	SSIM	PSNR	PieAPP
Blurred	0.5427 (0.1150)	21.7994 (2.1679)	4.2378 (0.8539)
VBA	0.4024 (0.1571)	16.0371 (4.1798)	2.4218 (0.5545)
deconv2D	0.6880 (0.1065)	23.1940 (2.8986)	2.2245 (0.6721)
blinddeconv	0.6961 (0.1034)	23.2663 (2.7229)	2.3259 (0.8080)
SelfDeblur	0.5107 (0.1305)	19.9943 (2.1467)	5.9269 (1.4066)
DBSRCNN	0.6948 (0.1688)	**23.6041** (4.2073)	1.9474 (0.7171)
DeblurGAN	0.3370 (0.0740)	17.2781 (1.2909)	3.6581 (1.0040)
Proposed (greedy)	0.7454 (0.1015)	23.2169 (2.4442)	**1.7250** (0.5324)
Proposed (end-to-end)	**0.7518** (0.1025)	**23.5631** (2.5959)	**1.7681** (0.5502)

TABLE VI
QUANTITATIVE ASSESSMENT OF THE RESTORED IMAGES. MEAN (STANDARD DEVIATION) VALUES COMPUTED OVER THE TEST SETS OF *Dataset 2*.

kernel estimation for the three considered metrics. In contrast, *end-to-end training* tends to favor the restored image quality while still providing a good kernel quality compared to other methods. In this more complicated context, standard VBA does not perform very well, as setting ξ becomes tedious for such an heterogeneous dataset. Let us note that the noise level is assumed to be known for this particular method, putting it in a quite favorable situation, compared to the other competitors, including our proposed approach. DBSRCNN provides again a good image recovery in terms of PSNR, but our proposed approach still outperforms it for both SSIM and PieAPP metrics while maintaining fairly high PSNR scores. We display two examples of restoration in Fig 8, when the sought blur is uniform, and out-of-focus, respectively. Such blur shapes are challenging and the MSE on the estimated blur might appear not excellent. Nevertheless, our method remains the best among the compared ones. The visual quality of the image generated by the proposed method is also very satisfying. We display in Fig. 6(right) the evolution of the SSIM loss during the *end-to-end training*, witnessing the absence of any overfitting issue. Moreover, we present in Fig 9 the evolution of the MSE loss on the kernel estimate, along the $K = 21$ layers of the architecture trained in an *end-to-end* manner. The MSE was averaged on test set examples associated to either Gaussian or out-of-focus blurs, respectively. These plots show that, for our choice of K (finetuned on the validation set), the MSE values are close to minimal. Larger K implied an increase of memory and training time, while not necessarily improving the results quality. One can also notice more fluctuations in the case of out-of-focus blur, which turns out to be more challenging to restore. A similar curve was obtained for uniform blurs, not shown by lack of space. Finally, Table IV(right) presents the average test time of the different methods. Again, our method appears competitive in terms of running time.

3) Dataset 3: The results of kernel estimation and image restoration on *Dataset 3* using the various methods are shown in Tables VII and VIII, respectively. The level of complexity of the blind deconvolution task again increases compared to the former experiments, as the blurs are of larger size and have less smooth kernel. The noise level also has to be estimated. Regarding the performance of kernel estimation, we can observe that the best performance are reached by the optimization-based method blinddeconv, followed closely by our method. SelfDeblur and deconv2D are slightly behind, in terms of quantitative metrics. Regarding the image restoration, the results are more contrasted. Hereagain, the proposed method with *end-to-end training* gives the best performance regarding the quality of the restored image, and at the same time keeps the quality of blur kernel estimation compared to *greedy training*. The standard VBA procedure reaches rather fair quality on this dataset. We must however point out that, as for *Dataset 2*, this method was run under the simplifying assumption that the noise level is known. Both deep learning methods DBSRCNN and DeblurGAN fail to recover a good image quality. Let us emphasize that DBSRCNN has been explicitely developed to tackle Gaussian blurs, following a patch-based strategy, which might explain its poor performance for deciphering complex non-localized blur shapes. We display two examples of restoration in Fig. 10 where the sought kernels correspond to two different motion blurs from the dataset. We can notice that most of the methods struggle in the estimation of the blur kernel. Although SelfDeblur yields fairly good results according to a visual inspection, this method seems to suffer from a shift ambiguity, typical in blind deconvolution, which explains its rather limited quantitative scores compared to the other methods. As for the image quality, our proposed method is the most satisfying visually, which is aligned with its best PieAPP score by far among all the methods. We display in Fig. 11(left) the evolution of the SSIM loss during the *end-to-end training*. The evolution of the MSE loss on the kernel estimate is displayed in Fig. 11(right), across the $K = 10$ layers of the architecture trained in an *end-to-end* manner. Slight fluctuations can be observed compared to the plots obtained for both synthetic blur datasets, which shows the high challenge of estimating such complex motion blur kernels. Table IV(right) displays the average test time for each methods. With the exception of DBSRCNN and DeblurGAN (but yielding rather low quality results), all the methods see their computational time increasing, compared to the other two datasets. This is directly related to the increase of the blur kernel size. This being said, the computational time for our proposed method remains competitive, especially when making use of GPU.

VI. CONCLUSION

This paper proposes a novel method for blind image deconvolution that combines a variational Bayesian algorithm with a neural network architecture. Our experiments illustrate the excellent performance of this new method on three datasets,

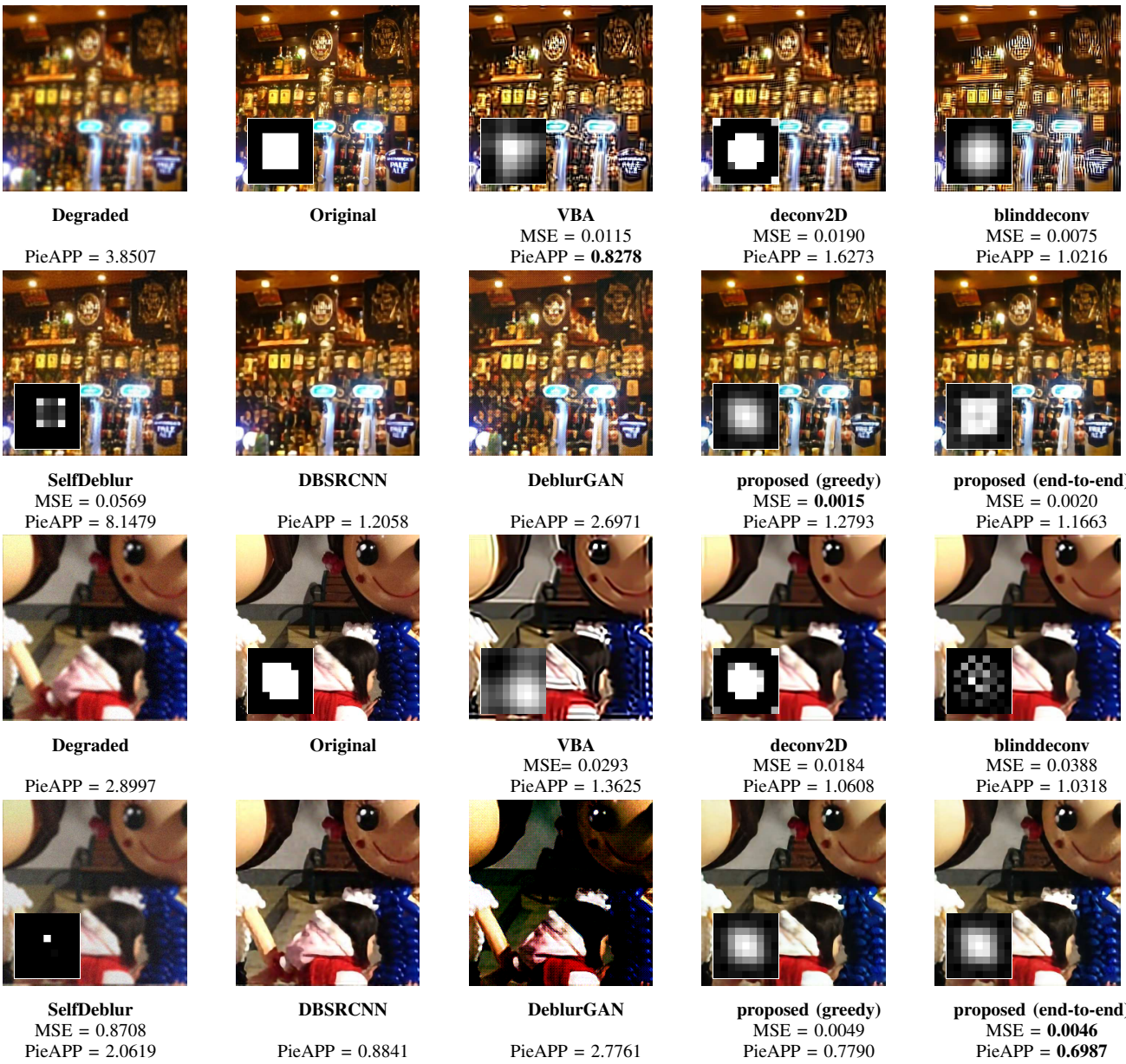

Fig. 8. Ground-truth image/blur, degraded image, restored images (with PieAPP index) and estimated blurs (with MSE score) when available, for various methods, on two examples in the test set of *Dataset 2*.

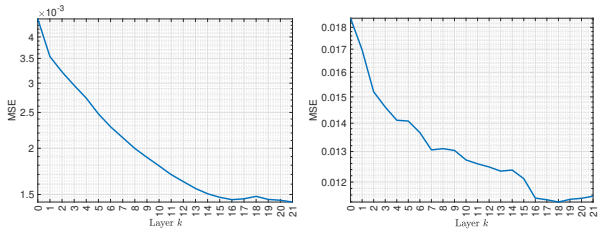

Fig. 9. MSE loss along the layers for proposed method using *end-to-end training*. Average over *Dataset 2* test examples involving either Gaussian (left), or out-of-focus (right) blur shapes.

Method	MSE	$\mathcal{H}_\infty$ error	MAE
VBA	0.0376 (0.0206)	0.6817 (0.2969)	0.2518 (0.0898)
deconv2D	0.0592 (0.0162)	1.0446 (0.3027)	0.2753 (0.0457)
blinddeconv	**0.0303** (0.0085)	**0.6249** (0.2088)	**0.2119** (0.0583)
SelfDeblur	0.0500 (0.0260)	0.7362 (0.3044)	0.2522 (0.0926)
Proposed (greedy)	**0.0320** (0.0159)	0.6564 (0.2918)	0.2189 (0.0697)
Proposed (end-to-end)	**0.0321** (0.0160)	0.6583 (0.2930)	0.2197 (0.0701)

TABLE VII
QUANTITATIVE ASSESSMENT OF THE RESTORED KERNELS. MEAN (STANDARD DEVIATION) VALUES COMPUTED OVER THE TEST SETS OF *Dataset 3*.

comprised of grayscale and color images, which are degraded with various synthetic and real-world kernel types. Compared to state-of-the-art variational and deep learning approaches, our method delivers a more accurate estimation of both the images and the blur kernels. It also includes an automatic noise estimation step and it requires minimal hyperparameter tuning. The proposed method is very competitive in terms of computational time during the test phase while showing

Fig. 10. Ground-truth image/blur, degraded image, restored images (with PieAPP index) and estimated blurs (with MSE score) when available, for various methods, on two examples in the test set of *Dataset 3*.

Method	SSIM	PSNR	PieAPP
Blurred	0.5830 (0.1500)	21.1404 (2.8626)	3.4647 (0.6608)
VBA	0.7344 (0.1242)	24.3958 (3.0088)	1.7028 (0.4914)
deconv2D	0.5440 (0.1422)	20.4229 (2.6241)	2.2811 (0.6407)
blinddeconv	0.4847 (0.1669)	16.0690 (2.7140)	0.6957 (0.5241)
SelfDeblur	0.7381 (0.1897)	24.3973 (4.6249)	0.6893 (0.3982)
DBSRCNN	0.5327 (0.1111)	19.7017 (2.2976)	2.3438 (0.5672)
DeblurGAN	0.5536 (0.1727)	21.0929 (3.2151)	2.3692 (0.7155)
Proposed (greedy)	0.9197 (0.0353)	29.5730 (1.9817)	0.1778 (0.2408)
Proposed (end-to-end)	**0.9295** (0.0342)	**31.0749** (2.0658)	**0.2227** (0.2309)

TABLE VIII
QUANTITATIVE ASSESSMENT OF THE RESTORED IMAGES. MEAN (STANDARD DEVIATION) VALUES COMPUTED OVER THE TEST SETS OF *Dataset 3*.

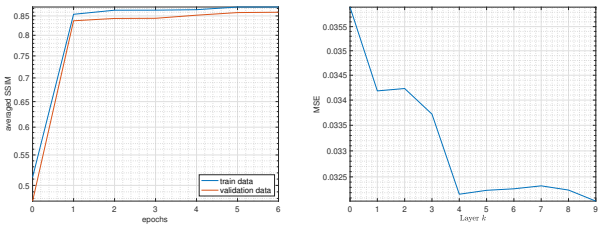

Fig. 11. (left) Evolution of SSIM loss along epochs of *end-to-end training* phase, averaged either on training or on validation sets of *Dataset 3*. (right) MSE loss along the layers for proposed method using *end-to-end training*. Average over *Dataset 3* test examples involving motion blur shapes.

similar train time to its deep learning competitors. The main core of the proposed architecture is highly interpretable, as it implements unrolled iterates of a well sounded Bayesian-based

blind deconvolution method. As a byproduct, it also outputs estimates for the covariance matrices of both sought quantities (image/kernel). This information could be of interest for uncertainty quantification and model selection tasks (see for instance [7], [64]). More generally, our work demonstrates that unrolling VBA algorithms constitutes a promising research direction for solving challenging problems arising in Data Science.

REFERENCES

[1] F. Murtagh, E. Pantin, and J.-L. Starck, *Deconvolution and blind deconvolution in astronomy*, pp. 277–316. Taylor and Francis, 2007.

[2] T. J. Holmes, D. Biggs, and A. Abu-Tarif, *Blind Deconvolution*, pp. 468–487. Boston, MA: Springer US, 2006.

[3] O. Michailovich and A. Tannenbaum, "Blind deconvolution of medical ultrasound images: A parametric inverse filtering approach," *IEEE Transactions on Image Processing*, vol. 16, no. 12, pp. 3005–3019, 2007.

[4] A. S. Carasso, "Direct blind deconvolution," *SIAM Journal on Applied Mathematics*, vol. 61, no. 6, pp. 1980–2007, 2001.

[5] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding and evaluating blind deconvolution algorithms," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, (Miami, FL, USA), pp. 1964–1971, 20-25 Jun. 2009.

[6] T. Bell, J. Xu, and S. Zhang, "Method for out-of-focus camera calibration," *Applied Optics*, vol. 55, no. 9, pp. 2346–2352, 2016.

[7] Y. Huang, E. Chouzenoux, and V. Elvira, "Probabilistic modeling and inference for sequential space-varying blur identification," *IEEE Transactions on Computational Imaging*, vol. 7, pp. 531–546, 2021.

[8] E. Chouzenoux, T.-K. Lau, C. Lefort, and J.-C. Pesquet, "Optimal multivariate Gaussian fitting with applications to psf modeling in two-photon microscopy imaging," *Journal of Mathematical Imaging and Vision*, vol. 61, no. 7, pp. 1037–1050, 2019.

[9] N. Komodakis and N. Paragios, "MRF-based blind image deconvolution," in *Proceedings of the Asian Conference on Computer Vision (ACCV 2012)*, (Daejeon, Korea), pp. 361–374, 5-9 Nov. 2012.

[10] J. Kotera, F. Šroubek, and P. Milanfar, "Blind deconvolution using alternating maximum a posteriori estimation with heavy-tailed priors," *Computer Analysis of Images and Patterns*, vol. 8048, pp. 59–66, 2013.

[11] D. Krishnan, T. Tay, and R. Fergus, "Blind deconvolution using a normalized sparsity measure," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, (Colorado Springs, CO, USA), pp. 233 – 240, 20-25 Jun. 2011.

[12] T. Chan and C.-K. Wong, "Total variation blind deconvolution," *IEEE transactions on image processing*, vol. 7, no. 3, pp. 370–375, 1998.

[13] A. Levin, "Blind motion deblurring using image statistics," in *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS 2006)*, vol. 19, (Vancouver, British Columbia, Canada), pp. 841–848, 4-7 Dec. 2006.

[14] Q. Shan, J. Jia, and A. Agarwala, "High-quality motion deblurring from a single image," *ACM Transactions on Graphics*, vol. 27, pp. 1–10, Aug. 2008.

[15] N. Joshi, R. Szeliski, and D. J. Kriegman, "PSF estimation using sharp edge prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, (Anchorage, AK, USA), pp. 1–8, 23-28 Jun. 2008.

[16] F. Abboud, Émilie Chouzenoux, J.-C. Pesquet, J.-H. Chenot, and L. Laborelli, "An alternating proximal approach for blind video deconvolution," *Signal Processing: Image Communication*, vol. 70, pp. 21–36, 2019.

[17] J. Bolte, P. L. Combettes, and J.-C. Pesquet, "Alternating proximal algorithm for blind image recovery," in *Proceedings of the IEEE International Conference on Image Processing (ICIP 2010)*, (Honk Kong, China), pp. 1673–1676, 26-29 Sep. 2010.

[18] J. Liu, M. Yan, and T. Zeng, "Surface-aware blind image deblurring," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 3, pp. 1041–1055, 2021.

[19] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer, 2004.

[20] D. Luengo, L. Martino, M. Bugallo, V. Elvira, and S. Särkkä, "A survey of monte carlo methods for parameter estimation," *EURASIP Journal on Advances in Signal Processing*, vol. 2020, pp. 1–62, 2020.

[21] M. Evans and T. Swartz, "Methods for approximating integrals in statistics with special emphasis on bayesian integration problems," *Statistical Science*, vol. 10, no. 3, pp. 254–272, 1995.

[22] O. Rosec, J.-M. Boucher, B. Nsiri, and T. Chonavel, "Blind marine seismic deconvolution using statistical MCMC methods," *IEEE Journal of Oceanic Engineering*, vol. 28, no. 3, pp. 502–512, 2003.

[23] D. Ge, J. Idier, and É. Le Carpentier, "Enhanced sampling schemes for MCMC based blind Bernoulli-Gaussian deconvolution," *Signal Processing*, vol. 91, pp. 759–772, Apr. 2011.

[24] G. Kail, J.-Y. Tourneret, F. Hlawatsch, and N. Dobigeon, "Blind deconvolution of sparse pulse sequences under a minimum distance constraint: A partially collapsed gibbs sampler method," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 2727–2743, 2012.

[25] T. E. Bishop, R. Molina, and J. R. Hopgood, "Blind restoration of blurred photographs via ar modelling and mcmc," in *Proceedings of the 15th IEEE International Conference on Image Processing (ICIP 2008)*, pp. 669–672, 2008.

[26] C. Fox and S. Roberts, "A tutorial on variational bayesian inference," *Artificial Intelligence Review*, vol. 38, p. 85–95, 2012.

[27] D. Blei, A. Kucukelbir, and J. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, Jan. 2016.

[28] Y. Marnissi, Y. Zheng, E. Chouzenoux, and J.-C. Pesquet, "A variational Bayesian approach for image restoration—application to image deblurring with Poisson–Gaussian noise," *IEEE Transactions on Computational Imaging*, vol. 3, no. 4, pp. 722–737, 2017.

[29] Y. Zheng, A. Fraysse, and T. Rodet, "Efficient variational Bayesian approximation method based on subspace optimization," *IEEE Transactions on Image Processing*, vol. 24, no. 2, pp. 681–693, 2015.

[30] S. Babacan, R. Molina, and A. Katsaggelos, "Variational Bayesian blind deconvolution using a total variation prior," *IEEE Transactions On Image Processing*, vol. 18, pp. 12–26, Feb. 2009.

[31] M. Pereyra, P. Schniter, E. Chouzenoux, J.-C. Pesquet, J.-Y. Tourneret, A. O. Hero, and S. McLaughlin, "A survey of stochastic simulation and optimization methods in signal processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, p. 224–241, Mar. 2016.

[32] Y. Marnissi, E. Chouzenoux, A. Benazza-Benyahia, and J.-C. Pesquet, "Majorize–minimize adapted Metropolis–Hastings algorithm," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2356–2369, 2020.

[33] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "DeblurGAN: Blind motion deblurring using conditional adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2018)*, (Salt Lake City, UT, USA), pp. 8183–8192, 18-22 Jun. 2018.

[34] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[35] F. Albluwi, V. A. Krylov, and R. Dahyot, "Image deblurring and super-resolution using deep convolutional neural networks," in *Proceedings of the IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP 2018)*, (Aalborg, Denmark), pp. 1–6, 17-20 Sep. 2018.

[36] D. Ren, K. Zhang, Q. Wang, Q. Hu, and W. Zuo, "Neural blind deconvolution using deep priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, (Seattle, WA, USA), pp. 3338–3347, 16-18 Jun. 2020.

[37] Y. Wen, J. Chen, B. Sheng, Z. Chen, P. Li, P. Tan, and T.-Y. Lee, "Structure-aware motion deblurring using multi-adversarial optimized cyclegan," *IEEE Transactions on Image Processing*, vol. 30, pp. 6142–6155, 2021.

[38] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV 2017)*, pp. 2242–2251, 2017.

[39] M. R. M. Mohan, G. K. Nithin, and A. N. Rajagopalan, "Deep dynamic scene deblurring for unconstrained dual-lens cameras," *IEEE Transactions on Image Processing*, vol. 30, pp. 4479–4491, 2021.

[40] A. Bietti, G. Mialon, and J. Mairal, "On regularization and robustness of deep neural networks." arXiv:1810.00363, 2018.

[41] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, 2021.

[42] Y. Li, M. Tofighi, J. Geng, V. Monga, and Y. C. Eldar, "Efficient and interpretable deep blind image deblurring via algorithm unrolling," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 666–681, 2020.

[43] C. Bertocchi, E. Chouzenoux, M.-C. Corbineau, J.-C. Pesquet, and M. Prato, "Deep unfolding of a proximal interior point method for image restoration," *Inverse Problems*, vol. 36, Sep. 2019.

[44] J.-C. Pesquet, A. Repetti, M. Terris, and Y. Wiaux, "Learning maximally monotone operators for image recovery," *SIAM Journal on Imaging Sciences*, vol. 14, no. 3, pp. 1206–1237, 2021.
[45] M. Galinier, M. Prato, E. Chouzenoux, and J.-C. Pesquet, "A hybrid interior point - deep learning approach for Poisson image deblurring," in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2020)*, (Espoo, Finland), p. 6p, Sept. 2020.
[46] K. Zhang, L. Van Gool, and R. Timofte, "Deep unfolding network for image super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, pp. 3217–3226, 2020.
[47] P. L. Combettes and J.-C. Pesquet, "Deep neural network structures solving variational inequalities," *Set-Valued and Variational Analysis*, vol. 28, pp. 491–518, Sep. 2020.
[48] E. Chouzenoux, C. D. Valle, and J.-C. Pesquet, "Inversion of integral models: a neural network approach." arXiv:2105.15044, 2021.
[49] Y. Sun, Z. Wu, X. Xu, B. Wohlberg, and U. S. Kamilov, "Scalable plug-and-play ADMM with convergence guarantees," *IEEE Transactions on Computational Imaging*, vol. 7, pp. 849–863, 2021.
[50] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, "Plug-and-play image restoration with deep denoiser prior," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
[51] C. Crockett and J. A. Fessler, "Bilevel methods for image reconstruction," *Foundations and Trends in Signal Processing*, vol. 15, no. 2-3, pp. 121–289, 2022.
[52] D. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proceedings of the International Conference on Learning Representations (ICLR 2014)*, (Banff, AB, Canada), 14-16 Apr. 2014.
[53] L. V. Jospin, W. L. Buntine, F. Boussaid, H. Laga, and M. Bennamoun, "Hands-on Bayesian neural networks - a tutorial for deep learning users," *ArXiv*, vol. abs/2007.06823, 2020.
[54] D. Gilton, G. Ongie, and R. Willett, "Model adaptation for inverse problems in imaging," *IEEE Transactions on Computational Imaging*, vol. 7, pp. 661–674, 2021.
[55] Y. Li, O. Bar-Shira, V. Monga, and Y. C. Eldar, "Deep algorithm unrolling for biomedical imaging," *Computing Research Repository*, vol. abs/2108.06637, 2021.
[56] B. Tolooshams, S. Mulleti, D. Ba, and Y. C. Eldar, "Unfolding neural networks for compressive multichannel blind deconvolution." arXiv:2010.1139 [Online], 2021.
[57] Y. Nan, Y. Quan, and H. Ji, "Variational-EM-based deep learning for noise-blind image deblurring," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, (Seattle, WA, USA), 16-18 Jun. 2020.
[58] Y. Li, M. Tofighi, V. Monga, and Y. C. Eldar, "An algorithm unrolling approach to deep image deblurring," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)*, (Brighton, UK), pp. 7675–7679, 12-17 May 2019.
[59] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
[60] B. Charlier, J. Feydy, J. Glaunès, F.-D. Collin, and G. Durif, "Kernel operations on the GPU, with autodiff, without memory overflows," *Journal of Machine Learning Research*, vol. 22, no. 74, pp. 1–6, 2021.
[61] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015)*, vol. 9351, (Munich, Germany), pp. 234–241, 5-9 Oct. 2015.
[62] M. Lebrun, "An analysis and implementation of the BM3D image denoising method," *Image Processing On Line*, vol. 2, pp. 175–213, Aug. 2012.
[63] E. Prashnani, H. Cai, Y. Mostofi, and P. Sen, "PieAPP: Perceptual image-error assessment through pairwise preference," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2018)*, (Salt Lake City, UT, USA), pp. 1808–1817, 18-22 Jun. 2018.
[64] A. Repetti, M. Pereyra, and Y. Wiaux, "Scalable Bayesian uncertainty quantification in imaging inverse problems via convex optimization," *SIAM Journal on Imaging Sciences*, vol. 12, no. 1, pp. 87–118, 2019.