End-to-end Interpretable Learning of Non-blind Image Deblurring

Thomas Eboli, Jian Sun, Jean Ponce

End-to-end Interpretable Learning of Non-blind Image Deblurring

Thomas Eboli[1], Jian Sun[2], and Jean Ponce[1]

[1] INRIA, Département d'informatique de l'ENS, ENS, CNRS, PSL University
{thomas.eboli,jean.ponce}@inria.fr
[2] Xi'an Jiaotong University
jiansun@xjtu.edu.cn
https://github.com/teboli/CPCR

Abstract. Non-blind image deblurring is typically formulated as a linear least-squares problem regularized by natural priors on the corresponding sharp picture's gradients, which can be solved, for example, using a half-quadratic splitting method with Richardson fixed-point iterations for its least-squares updates and a proximal operator for the auxiliary variable updates. We propose to precondition the Richardson solver using approximate inverse filters of the (known) blur and natural image prior kernels. Using convolutions instead of a generic linear preconditioner allows extremely efficient parameter sharing across the image, and leads to significant gains in accuracy and/or speed compared to classical FFT and conjugate-gradient methods. More importantly, the proposed architecture is easily adapted to learning both the preconditioner and the proximal operator using CNN embeddings. This yields a simple and efficient algorithm for non-blind image deblurring which is fully interpretable, can be learned end to end, and whose accuracy matches or exceeds the state of the art, quite significantly, in the non-uniform case.

Keywords: Non-blind deblurring, preconditioned fixed-point method, end-to-end learning.

1 Introduction

This presentation addresses the problem of non-blind image deblurring–that is, the recovery of a sharp image given its blurry version and the corresponding uniform or non-uniform motion blur kernel. Applications range from photography [17] to astronomy [34] and microscopy [15]. Classical approaches to this problem include least-squares and Bayesian models, leading to Wiener [40] and Lucy-Richardson [28] deconvolution techniques for example. Since many sharp images can lead to the same blurry one, blur removal is an ill-posed problem. To tackle this issue, variational methods [30] inject *a priori* knowledge over the set of solutions using penalized least-squares. Geman and Yang [12] introduce an auxiliary variable to solve this problem by iteratively evaluating a proximal operator [27] and solving a least-squares problem. The rest of this presentation builds on this *half-quadratic splitting* approach. Its proximal part has received a lot of attention through the design of complex model-based [21,36,43,46] or

learning-based priors [29]. Far less attention had been paid to the solution of the companion least-squares problem, typically relying on techniques such as conjugate gradient (CG) descent [2] or fast Fourier transform (FFT) [16, 42]. CG is relatively slow in this context, and it does not exploit the fact that the linear operator corresponds to a convolution. FFT exploits this property but is only truly valid under periodic conditions at the boundaries, which are never respected by real images.

We propose instead to use Richardson fixed-point iterations [18] to solve the least-squares problem, using approximate inverse filters of the (known) blur and natural image prior kernels as preconditioners. Using convolutions instead of a traditional linear preconditioner allows efficient parameter sharing across the image, which leads to significant gains in accuracy and/or speed over FFT and conjugate-gradient methods. To further improve performance and leverage recent progress in deep learning, several recent approaches to denoising and deblurring unroll a finite number of proximal updates and least-squares minimization steps [1, 6, 20, 22, 31]. Compared to traditional convolutional neural networks (CNNs), these algorithms use interpretable components and produce intermediate feature maps that can be directly supervised during training [22, 31].

We propose a solver for non-blind deblurring, also based on the splitting scheme of [12] but, in addition to learning the proximal operator as in [44], we also learn parameters in the fixed-point algorithm by embedding the preconditioner into a CNN whose bottom layer's kernels are the approximate filters discussed above. Unlike the algorithm of [44], our algorithm is trainable end to end, and achieves accuracy that matches or exceeds the state of the art. Furthermore, in contrast to other state-of-the-art CNN-based methods [22, 44] relying on FFT, it operates in the pixel domain and thus easily extends to non-uniform blurs scenarios.

1.1 Related work

Uniform image deblurring. Classical priors for natural images minimize the magnitude of gradients using the ℓ_2 [30], ℓ_1 [23] and hyper-Laplacian [21] (semi) norms or parametric potentials [36]. Instead of restoring the whole image at once, some works focus on patches by learning their probability distribution [46] or exploiting local properties in blurry images [25]. Handcrafted priors are designed so that the optimization is feasible and easy to carry out but they may ignore the characteristics of the images available. Data-driven priors, on the other hand, can be learned from a training dataset (*e.g.,* new regularizers). Roth and Black [29] introduce a learnable total variation (TV) prior whose parameters are potential/filter pairs. This idea has been extended in shallow neural networks in [31, 32] based on the splitting scheme of [12]. Deeper models based on Roth and Black's learnable prior had been proposed ever since [6, 20, 22]. The proximal operator can also be replaced by a CNN-based denoiser [24, 45] or a CNN specifically trained to mimic a proximal operator [44] or the gradient of the regularized [13]. More generally, CNNs are now used in various image restoration tasks, including non-blind deblurring by refining a low-rank decomposition of a deconvolution filter kernel [41], using a FFT-based solver [33], or to improve the accuracy of splitting techniques [1].

Non-uniform image deblurring. Non-uniform deblurring is more challenging than its uniform counterpart [5, 7]. Hirsch et al. [16] consider large overlapping patches and

suppose uniform motion on their supports before removing the local blur with an FFT-based uniform deconvolution method. Other works consider pixelwise locally linear motions [3,8,19,35] as simple elements representing complex global motions and solve penalized least-squares problems to restore the image. Finally, geometric non-uniform blur can be used in the case of camera shake to predict motion paths [38, 39].

1.2 Main contributions

Our contributions can be summarized as follows.

- We introduce a convolutional preconditioner for fixed-point iterations that efficiently solves the least-squares problem arising in splitting algorithms to minimize penalized energies. It is faster and/or more accurate than FFT and CG for this task, with theoretical convergence guarantees.
- We propose a new end-to-end trainable algorithm that implements a finite number of stages of half-quadratic splitting [12] and is fully interpretable. It alternates between proximal updates and preconditioned fixed-point iterations. The proximal operator and linear preconditioner are parameterized by CNNs in order to learn these functions from a training set of clean and blurry images.
- We evaluate our approach on several benchmarks with both uniform and non-uniform blur kernels. We demonstrate its robustness to significant levels of noise, and obtain results that are competitive with the state of the art for uniform blur and significantly outperforms it in the non-uniform case.

2 Proposed method

Let y and k respectively denote a blurry image and a known blur kernel. The *deconvolution* (or *non-blind deblurring*) problem can be formulated as

$$\min_x \frac{1}{2}||y - k \star x||_F^2 + \lambda \Omega(\sum_{i=1}^{n} k_i \star x), \tag{1}$$

where "$\star$" is the convolution operator, and x is the (unknown) sharp image. The filters k_i ($i = 1, \dots, n$) are typically partial derivative operators, and Ω acts as a regularizer on x, enforcing natural image priors. One often takes $\Omega(z) = ||z||_1$ (TV-ℓ_1 model). We propose in this section an end-to-end learnable variant of the method of *half-quadratic splitting* (or *HQS*) [12] to solve Eq. (1). As shown later, a key to the effectiveness of our algorithm is that all linear operations are explicitly represented by convolutions.

Let us first introduce notations that will simplify the presentation. Given some linear filters a_i and b_i ($i = 0, \dots, n$) with finite support (square matrices), we borrow the Matlab notation for "stacked" linear operators, and denote by $A = [a_0, \dots, a_n]$ and $B = [b_0; \dots; b_n]$ the (convolution) operators respectively obtained by stacking "horizontally" and "vertically" these filters, whose responses are

$$A \star x = [a_0 \star x, \dots, a_n \star x];\ B \star x = [(b_0 \star x)^\top, \dots, (b_n \star x)^\top]^\top; \tag{2}$$

We also define $A \star B = \sum_{i=0}^{n} a_i \star b_i$ and easily verify that $(A \star B) \star x = A \star (B \star x)$.

2.1 A convolutional HQS algorithm

Equation (1) can be rewritten as

$$\min_{x,z} \frac{1}{2}||y - k \star x||_F^2 + \lambda\Omega(z) \text{ such that } z = F \star x, \tag{3}$$

where $F = [k_1; \ldots; k_n]$. Let us define the energy function

$$E(x, z, \mu) = \frac{1}{2}||y - k \star x||_F^2 + \lambda\Omega(z) + \frac{\mu}{2}||z - F \star x||_F^2. \tag{4}$$

Given some initial guess x for the sharp image, (*e.g.* $x = y$) we can now solve our original problem using the *HQS* method [12] with T iterations of the form

$$\begin{array}{l} z \leftarrow \text{argmin}_z\, E(x, z, \mu); \\ x \leftarrow \text{argmin}_x\, E(x, z, \mu); \\ \mu \leftarrow \mu + \delta t. \end{array} \tag{5}$$

The μ update can vary with iterations but must be positive. We could also use the alternating direction method of multipliers (or *ADMM* [27]), for example, but this is left to future work. Note that the update in z has the form

$$z \leftarrow \underset{z}{\text{argmin}} \frac{\mu}{2}||z - F \star x||_F^2 + \lambda\Omega(z) = \varphi_{\lambda/\mu}(F \star x), \tag{6}$$

where $\varphi_{\lambda/\mu}$ is, by definition, the *proximal operator* [27] associated with Ω (a soft-thresholding function in the case of the ℓ_1 norm [9]) given λ and μ.

The update in x can be written as the solution of a linear least-squares problem:

$$x \leftarrow \underset{x}{\text{argmin}} \frac{1}{2}||u - L \star x||_F^2, \tag{7}$$

where $u = [y; \sqrt{\mu}z]$ and $L = [k; \sqrt{\mu}F]$.

2.2 Convolutional PCR iterations

Many methods are of course available for solving Eq. (7). We propose to compute x as the solution of $C \star (u - L \star x) = 0$, where $C = [c_0, \ldots, c_n]$ is composed of $n+1$ filters and is used in *preconditioned Richardson* (or *PCR*) fixed-point iterations [18].

Briefly, in the generic linear case, PCR is an iterative method for solving a square, nonsingular system of linear equations $Ax = b$. Given some initial estimate $x = x_0$ of the unknown x, it repeatedly applies the iterations

$$x \leftarrow x - C(Ax - b), \tag{8}$$

where C is a preconditioning square matrix. When C is an *approximate inverse* of A, that is, when the spectral radius η of $\text{Id} - CA$ is smaller than one, preconditioned Richardson iterations converge to the solution of $Ax = b$ with a linear rate proportional to η [18]. When A is an $m \times n$ matrix with $m \geq n$, and x and b are respectively

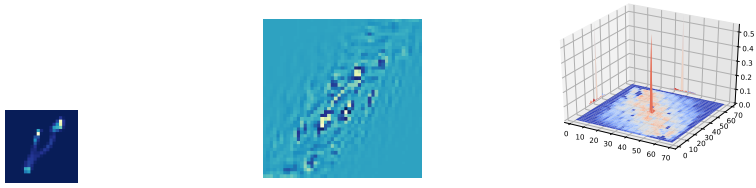

Fig. 1: From left ot right: An example of a blur kernel k from the Levin *et al.* dataset [23]; its approximate inverse kernel c_0; the resulting filter resulting from the convolution of k and c_0 (represented as a surface). It gives an approximate Dirac filter δ.

elements of $\mathbb{R}^n$ and $\mathbb{R}^m$, PCR can also be used in Cimmino's algorithm for linear least-squares, where the solution of $\min_x ||Ax - b||^2$ is found using $C = \rho A^\top$, with $\rho > 0$ sufficiently small, as the solution of $A^\top Ax - A^\top b = 0$, with similar guarantees. Finally, it is also possible to use a different $n \times m$ matrix. When the spectral radius η of the $n \times n$ matrix $\mathrm{Id} - CA$ is smaller than one, the PCR iterations converge once again at a linear rate proportional to η. However, they converge to the (unique in general) solution of $C(Ax - b) = 0$, which may of course be different from the least-squares solution.

This method is easily adapted to our context. Since L corresponds to a bank of filters of size $w_k \times w_k$, it is natural to take $C = [c_0, \ldots, c_n]$ to be another bank of $n + 1$ linear filters of size $w_c \times w_c$. Unlike a generic linear preconditioner satisfying $CA \approx \mathrm{Id}$ in matrix form, whose size depends on the square of the image size, C exploits the structure of L and is a linear operator with *much* fewer parameters, *i.e.* $n + 1$ times the size of the c_i's. Thus, C is an approximate inverse filter bank for L, in the sense that

$$\delta \approx L \star C = C \star L = c_0 \star k + \sqrt{\mu} \textstyle\sum_{i=1}^{n} c_i \star k_i, \tag{9}$$

where δ is the Dirac filter. In this setting, C is computed as the solution of

$$C = \mathrm{argmin}_C \, ||\delta - L \star C||_F^2 + \rho \textstyle\sum_{i=0}^{n} ||c_i||_F^2, \tag{10}$$

The classical solution using the pseudo inverse of L has cost $\mathcal{O}\left((w_k + w_c - 1)^{2\times 3}\right)$.

$$c_i = \mathcal{F}^{-1}\left(\frac{\tilde{K}_i^*}{\rho J + \sum_{j=0}^{n} |\tilde{K}_j|^2}\right) \quad \text{for } i = 0 \text{ to } n, \tag{11}$$

using the fast Fourier transform (FFT) with cost $\mathcal{O}(w_c^2 \log(w_c))$ [11]. $\mathcal{F}^{-1}$ is the inverse Fourier transform, J is a matrix full of ones, $\tilde{K}_i$ is the Fourier transform of k_i (with $k_0 = k$), $\tilde{K}_i^*$ is its complex conjugate and the division in the Fourier domain is entrywise. Note that the use of FFT in this context has nothing to do with its use as a deconvolution tool for solving Eq. (7). Figure 1 shows an example of a blur kernel from [23], its approximate inverse when $n = 0$ and the result of their convolution. Let us define $[A]_\star$ as the linear operator such that $[A]_\star B = A \star B$. Indeed, a *true* inverse filter bank such that equality holds in Eq. (9) does not exist in general (*e.g.,* a Gaussian filter cannot be inverted), but all that matters is that the linear operator associated with $\delta - C \star L$ has a spectral radius smaller than one [18]. We have the following result.

Lemma 1. *The spectral radius of the linear operator* $Id - [L]_\star[C]_\star$*, where* C *is the optimal solution of* (10) *given by* (11) *is always smaller than* 1 *when* $[L]_\star$ *has full rank.*

A detailed proof can be found in the supplemental material. We now have our basic non-blind deblurring algorithm, in the form of the Matlab-style CHQS (for *convolutional HQS*, primary) and CPCR (for *convolutional PCR*, auxiliary) functions below.

function $x = \text{CHQS}(y, k, F, \mu 0)$
$x = y$; $\mu = \mu 0$;
for $t = 0 : T - 1$ **do**
  $u = [y; \sqrt{\mu}\varphi_{\lambda/\mu}(F \star x)]$;
  $L = [k; \sqrt{\mu}F]$;
  $C = \text{argmin}_C \, ||\delta - C \star L||_F^2 + \rho \sum_{i=0}^{n} ||c_i||_F^2$;
  $x = \text{CPCR}(L, u, C, x)$;
  $\mu = \mu + \delta_t$;
end for
end function

function $x = \text{CPCR}(A, b, C, x_0)$
$x = x_0$;
for $s = 0 : S - 1$ **do**
  $x = x - C \star (A \star x - b)$;
end for
end function

2.3 An end-to-end trainable CHQS algorithm

To improve on this method, we propose to learn the proximal operator φ and the preconditioning operator C. The corresponding *learnable* CHQS (LCHQS) algorithm can now be written as a function with two additional parameters θ and ν as follows.

function $x = \text{LCHQS}(y, k, F, \mu_0, \theta, \nu)$
$x = y$; $\mu = \mu_0$;
for $t = 0 : T - 1$ **do**
  $u = [y; \sqrt{\mu}\varphi^{\theta}_{\lambda/\mu}(F \star x)]$;
  $L = [k; \sqrt{\mu}F]$;
  $C = \text{argmin}_C \, ||\delta - C \star L||_F^2 + \rho \sum_{i=0}^{n} ||c_i||_F^2$;
  $x = \text{CPCR}(L, u, \psi^{\nu}(C), x)$;
  $\mu = \mu + \delta_t$;
end for
end function

The function LCHQS has the same structure as CHQS but now uses two parameterized embedding functions φ^{θ}_{τ} and ψ^{ν} for the proximal operator and preconditioner. In practice, these functions are CNNs with learnable parameters θ and ν as detailed in Sec. 3. Note that θ actually determines the regularizer through its proximal operator. The function LCHQS is differentiable with respect to both its θ and ν parameters. Given a set of training triplets $(x^{(i)}, y^{(i)}, k^{(i)})$ (in $i = 1, \ldots, N$), the parameters θ and ν can thus be learned end-to-end by minimizing

$$F(\theta, \nu) = \sum_{i=1}^{N} ||x^{(i)} - \text{LCHQS}(y^{(i)}, k^{(i)}, F, \theta, \nu)||_1, \tag{12}$$

with respect to these two parameters by "unrolling" the HQS iterations and using back-propagation, as in [6, 44] for example. This can be thought of as the "compilation" of a fully interpretable iterative optimization algorithm into a CNN architecture. Empirically, we have found that the ℓ_1 norm gives better results than the ℓ_2 norm in Eq. (12).

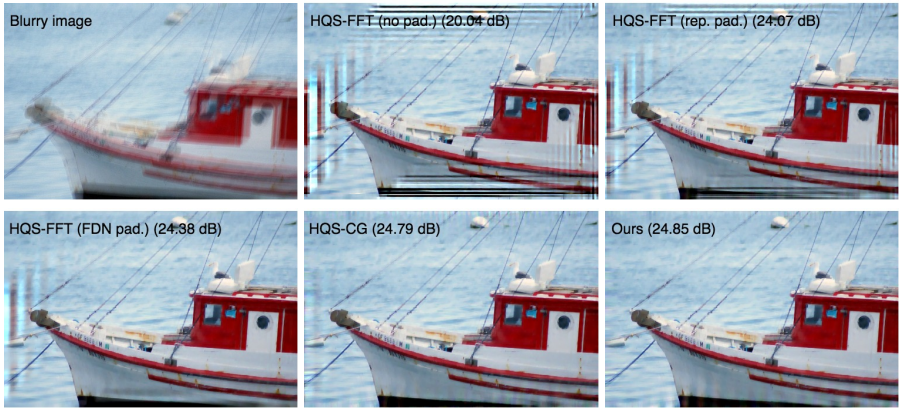

Fig. 2: A blurry image from our test set with 2% white noise and the solutions of Eq. (1) with TV-ℓ_1 regularization obtained with different HQS-based methods. From the same optimization problem, HQS-FFT displays boundary artifacts. HQS-CG and CHQS produce images with similar visual quality and PSNR values but HQS-CG is much slower.

3 Implementation and results

3.1 Implementation details

Network architectures. The global architecture of LCHQS shares the same pattern than FCNN [44], *i.e.*, $n = 2$ in Eq. (1) with $k_1 = [1, -1]$ and $k_2 = k_1^\top$, and the model repeats between 1 and 5 stages alternatively solving the proximal problem (6) and the linear least-squares problem (7). The proximal operator φ^θ is the same as the one introduced in [44], and it is composed of 6 convolutional layers with 32 channels and 3×3 kernels, followed by ReLU non-linearities, except for the last one. The first layer has 1 input channel and the last layer has 1 output channel. The network ψ^ν featured in LCHQS is composed of 6 convolutional layers with 32 channels and 3×3 kernels, followed by ReLU non-linearities, except for the last one. The first layer has $n + 1$ input channels (3 in practice with the setting detailed above) corresponding to the filtered versions of x with the c_i's, and the last layer has 1 output channel. The filters c_1 and c_2 are of size 31×31. This size is intentionally made relatively large compared to the sizes of k_1 and k_2 because inverse filters might have infinite support in principle. The size of c_0 is twice the size of the blur kernel k. This choice will be explained in Sec. 3.2. In our implementation, each LCHQS stage has its own θ and ν parameters. The non-learnable CHQS module solves a TV-ℓ_1 problem; the proximal step implements the soft-thresholding operation φ with parameter λ/μ and the least-squares step implements CPCR. The choice of μ will be detailed below.

Datasets. The training set for uniform blur is made of 3000 patches of size 180×180 taken from BSD500 dataset and as many random 41×41 blur kernels synthesized with the code of [4]. We compute ahead of time the corresponding inverse filters c_i and set the size of c_0 to be 83×83 with Eq. (11) where ρ is set to 0.05, a value we have chosen

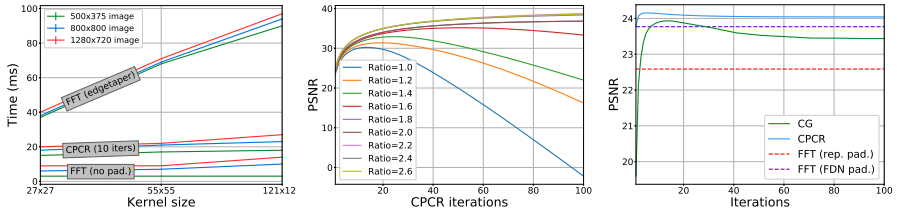

Fig. 3: From left to right: Computation times for CPCR (including computation of C) with FFT applied on images padded with "edgetaper" as recommended in [22] and non-padded images for three image formats; effect of the w_{c_0}/w_k ratio on performance; comparison of CG, FFT and CPCR for solving (7).

after cross-validation on a separate test set. We also create a training set for non-uniform motion blur removal made of 3000 180×180 images synthesized with the code of [14] with a locally linear motion of maximal magnitude of 35 pixels. For both training sets, the validation sets are made of 600 additional samples. In both cases, we add Gaussian noise with standard deviation matching that of the test data. We randomly flip and rotate by $90°$ the training samples and take 170×170 random crops for data augmentation.

Optimization. Following [22], we train our model in a two-step fashion: First, we supervise the sharp estimate output by each iteration of LCHQS in the manner of [22] with Eq. (12). We use an Adam optimizer with learning rate of 10^{-4} and batch size of 1 for 200 epochs. Second, we further train the network by supervising the final output of LCHQS with Eq. (12) on the same training dataset with an Adam optimizer and learning rate set to 10^{-5} for 100 more epochs *without* the per-layer supervision. We have obtained better results with this setting than using either of the two steps separately.

3.2 Experimental validation of CPCR and CHQS

In this section, we present an experimental sanity check of CPCR for solving (7) and CHQS for solving (1) in the context of a basic TV-ℓ_1 problem.

Inverse kernel size. We test different sizes for the $w_{c_0} \times w_{c_0}$ approximate inverse filter c_0 associated with a $w_k \times w_k$ blur kernel k, in the non-penalized case, with $\lambda = 0$. We use Eq. (11) with ρ set to 0.05. We use 160 images obtained by applying the 8 kernels of [23] to 20 images from the Pascal VOC 2012 dataset. As shown in Fig. 3 , the PSNR increases with increasing w_{c_0}/w_k ratios, but saturates when the ratio is larger than 2.2. We use a ratio of 2 which is a good compromise between accuracy and speed.

CPCR accuracy. We compare the proposed CPCR method to FFT-based deconvolution (FFT) and conjugate gradient descent (CG), to solve the least-squares problem of Eq. (7) in the setting of a TV-ℓ_1 problem. We follow [22] and, in order to limit boundary artifacts for FFT, we pad the images to be restored by replicating the pixels on the boundary with a margin of half the size of the blur kernel and then use the "edgetaper" routine. We also run FFT on images padded with the "replicate" strategy consisting in simply replicating the pixels on the boundary. We solve Eq. (7) with $\mu_0 = 0.008$,

	ker-1	ker-2	ker-3	ker-4	ker-5	ker-6	ker-7	ker-8	Aver.	Time (s)
HQS-FFT (no pad.)	21.14	20.51	22.31	18.21	23.36	20.01	19.93	19.02	20.69	0.07
HQS-FFT (rep. pad.)	26.45	25.39	**26.27**	22.75	27.64	27.26	24.84	23.54	25.53	0.07
HQS-FFT (FDN pad.)	**26.48**	25.89	**26.27**	23.79	27.66	27.23	25.26	25.02	25.96	0.15
HQS-CG	26.39	25.90	26.24	24.88	27.59	27.31	25.39	25.19	26.12	13
CHQS	26.45	**25.96**	26.26	**25.06**	**27.67**	**27.51**	**25.81**	**25.48**	**26.27**	0.26

Table 1: Comparison of different methods optimizing the same TV-ℓ_1 deconvolution model (1) on 160 synthetic blurry images with 2% white noise. We run all the methods on a GPU. The running times are for a 500×375 RGB image.

$\lambda = 0.003$ and z computed beforehand with Eq. (6). The 160 images previously synthesized are degraded with 2% additional white noise. Figure 3 shows the average PSNR scores for the three algorithms optimizing Eq. (7). After only 5 iterations, CPCR produces an average PSNR higher than the other methods and converges after 10 iterations. The "edgetaper" padding is crucial for FFT to compete with CG and CPCR by reducing the amount of border artifacts in the solution.

CPCR running time. CPCR relies on convolutions and thus greatly benefits from GPU acceleration. For instance, for small images of size 500×375 and a blur kernel of size 55×55, 10 iterations of CPCR are in the ballpark of FFT without padding: CPCR runs in 20ms, FFT runs in 3ms and FFT with "edgetaper" padding takes 40ms. For a high-resolution 1280×720 image and the same blur kernel, 10 iterations of CPCR run in 22ms, FFT without padding runs in 10ms and "edgetaper" padded FFT in 70ms. Figure 3 compares the running times of CPCR (run for 10 iterations) with padded/non-padded FFT for three image (resp. kernel) sizes: 500×375, 800×800 and 1280×720 (27×27, 55×55 and 121×121) pixels. Our method is marginally slower than FFT without padding in every configuration (within a margin of 20ms) but becomes much faster than FFT combined to "edgetaper" padding when the size of the kernel increases. FFT with "replicate" padding runs in about the same time as FFT (no pad) and thus is not shown in Fig. 3 . The times have been averaged over 1000 runs.

Running times for computing the inverse kernels with Eq. (11)**.** Computing the inverse kernels c_i, with an ratio w_c/w_k set to 2, takes 1.0 ± 0.2ms for a blur kernel k of size 27×27 and 5.4 ± 0.5ms (results averaged in 1000 runs) for a large 121×121 kernel. Thus, the time for inverting blur kernels is negligible in the overall pipeline.

CHQS validation. We compare several iterations of HQS using unpadded FFT (HQS-FFT (no pad.)), with "replicate" padding (HQS-FFT (rep. pad)), and the padding strategy proposed in [22] (HQS-FFT (FDN pad.)), CG (HQS-CG), or CPCR (CHQS) for solving the least-squares problem penalized with the TV-ℓ_1 regularized in Eq. (1) and use the same 160 blurry and noisy images than in previous paragraph as test set. We set the number of HQS iterations T to 10, run CPCR for 5 iterations and CG for at most 100 iterations. We use $\lambda = 0.003$ and $\mu_t = 0.008 \times 4^t$ ($t = 0, \dots, T-1$). Table 1 compares the average PSNR scores obtained with the different HQS algorithms over the test set. As expected, FDN padding greatly improves HQS-FFT results on larger kernels over naive "replicate" padding, *i.e.* "ker-4" and "ker-8", but overall does not perform

	FCNN [44]	EPLL [46]	RGCD [13]	FDN [22]	CHQS	LCHQS$_G$	LCHQS$_F$
Levin [23]	33.08	34.82	33.73	35.09	32.12	$\underline{35.11 \pm 0.05}$	**35.15** $\pm$ **0.04**
Sun [37]	32.24	32.46	31.95	32.67	30.36	$\underline{32.83 \pm 0.01}$	**32.93** $\pm$ **0.01**

Table 2: PSNR scores for Levin [23] and Sun [37] benchmarks, that respectively feature 0.5% and 1% noise. Best results are shown in bold, second-best underlined. The difference may not always be significant between FDN and LCHQS for the Levin dataset.

Fig. 4: Comparison of state-of-the-art methods and the proposed LCHQS for one sample of the Levin dataset [23] (better seen on a computer screen). FDN effectively removes the blur but introduces artifacts in flat areas, unlike EPLL and LCHQS.

as well as CHQS. For kernels 1, 2, 3 and 5, the four methods yield comparable results (within 0.1 dB of each other). FFT-based methods are significantly worse on the other four, whereas our method gives better results than HQS-CG in general, but is 100 times faster. This large speed-up is explained by the convolutional structure of CPCR whereas CG involves large matrix inversions and multiplications. Figure 2 shows a deblurring example from the test set. HQS-FFT (with FDN padding strategy), even with the refined padding technique of [22], produces a solution with boundary artifacts. Both HQS-CG and CHQS restore the image with a limited amount of artifacts, but CHQS does it much faster than HQS-CG. This is typical of our experiments in practice.

Discussion. These experiments show that CPCR always gives better results than CG in terms of PSNR, sometimes by a significant margin, and it is about 50 times faster. This suggests that CPCR may, more generally, be preferable to CG for linear least-squares problems when the linear operator is a convolution. CPCR also dramatically benefits from its convolutional implementation on a GPU with speed similar to FFT and is even faster than FFT with FDN padding for large kernels. These experiments also show that CHQS surpasses, in general, HQS-CG and HQS-FFT for deblurring.

Next, we further improve the accuracy of CHQS using supervised learning, as done in previous works blending within a single model variational methods and learning.

3.3 Uniform deblurring

We compare in this section CHQS and its learnable version LCHQS with the non-blind deblurring state of the art, including optimization-based and CNN-based algorithms.

Comparison on standard benchmarks. LCHQS is first trained by using the loss of Eq. (12) to supervise the output of each stage of the proposed model and second

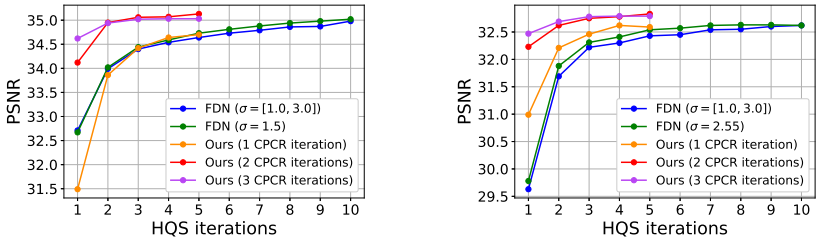

Fig. 5: Performance of FDN [22] and LCHQS on the Levin [23] (left) and Sun [37] (right) datasets.

trained by only supervising the output of the final layer, in the manner of [22]. The model trained in the first regime is named LCHQS_G and the one further trained with the second regime is named LCHQS_F. The other methods we compare our learnable model to are HQS algorithms solving a TV-ℓ_1 problem: HQS-FFT (with the padding strategy of [22]), HQS-CG and CHQS, an HQS algorithm with a prior over patches (EPLL) [46] and the state-of-the-art CNN-based deblurring methods FCNN [44] and FDN [22]. We use the best model provided by the authors of [22], denoted as FDN_T^{10} in their paper. Table 2 compares our method with these algorithms on two classical benchmarks. We use 5 HQS iterations and 2 CPCR iterations for CHQS and LCHQS. Except for EPLL that takes about 40 seconds to restore an image of the Levin dataset [23], all methods restore a 255×255 black and white image in about 0.2 second. The dataset of Sun *et al.* contains high-resolution images of size around 1000×700 pixels. EPLL removes the blur in 20 minutes on a CPU while the other methods, including ours, do it in about 1 second on a GPU. In this case, our learnable method gives comparable results to FDN [22], outputs globally much sharper results than EPLL [46] and is much faster. As expected, non-trainable CHQS is well behind its learned competitors (Tab. 2).

Number of iterations for LCHQS_G and CPCR. We investigate the influence of the number of HQS and CPCR iterations on the performance of LCHQS_G on the benchmarks of Levin *et al.* [23] and Sun *et al.* [37]. FDN implements 10 HQS iterations parameterized with CNNs but operates in the Fourier domain. Here, we compare LCHQS_G to the FDN model trained in a stage-wise manner (denoted as FDN_G^{10} in [22]). Figure 5 plots the mean PSNR values for the datasets of Levin *et al.* and Sun *et al.* [37] after each stage. FDN comes in two versions: one trained on a single noise level (green line) and one trained on noise levels within a given interval (blue line). We use up to 5 iterations of our learnable CHQS scheme, but it essentially converges after only 3 steps. When the number of CPCR iterations is set to 1, FDN and our model achieve similar results for the same number of HQS iterations. For 2/3 CPCR iterations, we do better than FDN for the same number of HQS iterations by a margin of +0.4/0.5dB on both benchmarks. For 3 HQS iterations and more, LCHQS saturates but systematically achieves better results than 10 FDN iterations: +0.15dB for [23] and +0.26dB for [37].

Robustness to noise. Table 3 compares our methods for various noise levels on the 160 RGB images introduced previously, dubbed from "PASCAL benchmark". FDN corresponds to the model called FDN_T^{10} in [22]. For this experiment, (L)CHQS uses 5

Fig. 6: Example of image deblurring with an additive noise of 3% (better seen on a computer screen). In this example, we obtain better PSNR scores than competitors and better visual results, for example details around the door or the leaves.

HQS iterations and 2 inner CPCR iterations. We add 1%, 3% and 5% Gaussian noise to these images to obtain three different test sets with gradually stronger noise levels. We train each model to deal with a specific noise level (non-blind setting) but also train a single model to handle multiple noise levels (blind setting) on images with 0.5 to 5% of white noise, as done in [22]. For each level in the non-blind setting, we are marginally above or below FDN results. In terms of average PSNR values, the margins are +0.12dB for 1%, +0.06dB for 3% and -0.05dB for 5% when comparing our models with FDN, but we are above the other competitors by margins between 0.3dB and 2dB. Compared to its noise-dependent version, the network trained in the blind setting yields a loss of 0.2dB for 1% noise, but gains of 0.14 and 0.27dB for 3 and 5% noises, showing its robustness and adaptability to various noises. Figure 6 compares results obtained on a blurry image with 3% noise.

3.4 Non-uniform motion blur removal

Typical non-uniform motion blur models assign to each pixel of a blurry image a local uniform kernel [5]. This is equivalent to replacing the uniform convolution in Eq. (1) by local convolutions for each overlapping patch in an image, as done by Sun *et al.* [35] when they adapt the solver of [46] to the non-uniform case. Note that FDN [22] and FCNN [44] operate in the Fourier domain and thus cannot be easily adapted to non-uniform deblurring, unlike (L)CHQS operating in the spatial domain. We handle non-uniform blur as follows to avoid computing different inverse filters at each pixel. As in [35], we model a non-uniform motion field with *locally* linear motions that can well approximate complex *global* motions such as camera rotations. We discretize the set of the linear motions by considering only those with translations (in pixels) in $\{1, 3 \ldots, 35\}$ and orientations in $\{0°, 6°, \ldots, 174°\}$. In this case, we know in ad-

	1% noise	3% noise	5% noise	Time (s)
HQS-FFT	26.48	23.90	22.15	0.2
HQS-CG	26.45	23.91	22.27	13
EPLL [46]	28.83	24.00	22.10	130
FCNN [44]	29.27	25.07	23.53	0.5
FDN [22]	29.42	25.53	23.97	0.6
CHQS	27.08	23.33	22.38	0.3
LCHQS_G (non-blind)	**29.54** $\pm$ **0.02**	25.59 $\pm$ 0.03	23.87 $\pm$ 0.06	0.7
LCHQS_F (non-blind)	**29.53** $\pm$ **0.02**	25.56 $\pm$ 0.03	23.95 $\pm$ 0.05	0.7
LCHQS_G (blind)	29.22 $\pm$ 0.02	25.55 $\pm$ 0.03	24.05 $\pm$ 0.02	0.7
LCHQS_F (blind)	29.35 $\pm$ 0.01	**25.71** $\pm$ **0.02**	**24.21** $\pm$ **0.01**	0.7

Table 3: Uniform deblurring on 160 test images with 1%, 3% and 5% white noise. Running times are for an 500×375 RGB image. The mention "blind" (resp. "non-blind) indicates that a single model handles the three (resp. a specific) noise level(s).

Fig. 7: Non-uniform motion deblurring example with 1 % additive Gaussian noise (better seen on a computer screen). The car and the helmet are sharper with our method than in the images produced by our competitors.

vance all the 511 35×35 local blur kernels and compute their approximate inverses ahead of time. During inference, we simply determine which one best matches the local blur kernel and use its approximate inverse in CPCR. This is a parallelizable operation on a GPU. Table 4 compares our approach (in non-blind setting) to existing methods for locally-linear blur removal on a test set of 100 images from PASCAL dataset non-uniformly blurred with the code of [14] and with white noise. For instance for 1% noise, LCHQS_G scores +0.99dB higher than CG-based method, and LCHQS_F pushes the margin up to +1.13dB while being 200 times faster. Figure 7 shows one non-uniform example from the test set.

3.5 Deblurring with approximated blur kernels

In practice one does not have the ground-truth blur kernel but instead an *approximate* version of it, obtained with methods such as [26, 39]. We show that (L)CHQS works well for approximate and/or large filters, different from the ones used in the training set and without any training or fine-tuning. We show in Figure 8 a deblurred image with an approximate kernel obtained with the code of [26] and of support of size 101×101 pixels. We obtain with LCHQS_F (blind) of Table 3 a sharper result than FCNN and do

	HQS-FFT	HQS-CG	EPLL [46]	CHQS	LCHQS_G	LCHQS_F
1% noise	23.49	25.84	25.49	25.11	26.83 ± 0.08	$\mathbf{26.98 \pm 0.08}$
3% noise	23.17	24.18	23.78	23.74	24.91 ± 0.05	$\mathbf{25.06 \pm 0.06}$
5% noise	22.44	23.10	23.34	22.65	23.97 ± 0.05	$\mathbf{24.14 \pm 0.05}$
Time (s)	13	212	420	0.8	0.9	0.9

Table 4: Non-uniform deblurring on 100 test images with 1%, 3% and 5% white noise. Running times are for an 500×375 RGB image.

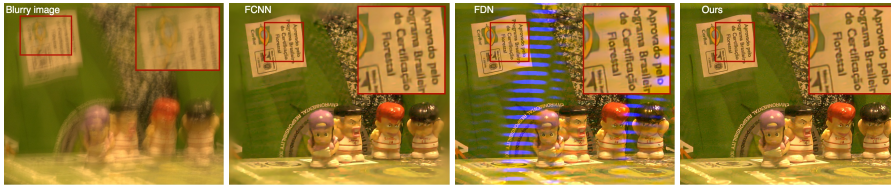

Fig. 8: Real-world blurry images deblurred with an 101×101 blur kernel estimated with [26]. We can restore fine details with approximate, large kernels.

not introduce artifacts as FDN, showing the robustness of CPCR and its embedding in HQS to approximate blur kernels. More results are shown in the supplemental material.

4 Conclusion

We have presented a new learnable solver for non-blind deblurring. It is based on the HQS algorithm for solving penalized least-squares problems but uses preconditioned iterative fixed-point iterations for the x-update. Without learning, this approach is superior both in terms of speed and accuracy to classical solvers based on the Fourier transform and conjugate gradient descent. When the preconditioner and the proximal operator are learned, we obtain results that are competitive with or better than the state of the art. Our method is easily extended to non-uniform deblurring, and it outperforms the state of the art by a significant margin in this case. We have also demonstrated its robustness to important amounts of white noise. Explicitly accounting for more realistic noise models [10] and other degradations such as downsampling is left for future work.

Acknowledgments. This works was supported in part by the INRIA/NYU collaboration and the Louis Vuitton/ENS chair on artificial intelligence. In addition, this work was funded in part by the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR19-P3IA-0001 (PRAIRIE 3IA Institute). Jian Sun was supported by NSFC under grant numbers 11971373 and U1811461.

References

1. Aljadaany, R., Pal, D.K., Savvides, M.: Douglas-rachford networks: Learning both the image prior and data fidelity terms for blind image deconvolution. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 10235–10244 (2019)
2. Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2014)
3. Brooks, T., Barron, J.T.: Learning to synthesize motion blur. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 6840–6848 (2019)
4. Chakrabarti, A.: A neural approach to blind motion deblurring. In: Proceedings of the European Conference on Computer Vision. pp. 221–235 (2016)
5. Chakrabarti, A., Zickler, T.E., Freeman, W.T.: Analyzing spatially-varying blur. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 2512–2519 (2010)
6. Chen, Y., Pock, T.: Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(6), 1256–1272 (2017)
7. Cho, S., Matsushita, Y., Lee, S.: Removing non-uniform motion blur from images. In: Proceedings of the International Conference on Computer Vision. pp. 1–8 (2007)
8. Couzinie-Devy, F., Sun, J., Alahari, K., Ponce, J.: Learning to estimate and remove non-uniform image blur. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 1075–1082 (2013)
9. Elad, M.: Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing. Springer Publishing Company, Incorporated (2010)
10. Foi, A., Trimeche, M., Katkovnik, V., Egiazarian, K.O.: Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. IEEE Transactions in Image Processing **17**(10), 1737–1754 (2008)
11. Folland, G.B.: Fourier Analysis and its Applications. Wadsworth (1992)
12. Geman, D., Yang, C.: Nonlinear image recovery with half-quadratic regularization. IEEE Transactions in Image Processing **4**(7), 932–946 (1995)
13. Gong, D., Zhang, Z., Shi, Q., van den Hengel, A., Shen, C., Zhang, Y.: Learning deep gradient descent optimization for image deconvolution. IEEE Transactions on Neural Networks and Learning Systems pp. 1–15 (2020)
14. Gong, D., Yang, J., Liu, L., Zhang, Y., Reid, I.D., Shen, C., van den Hengel, A., Shi, Q.: From motion blur to motion flow: A deep learning solution for removing heterogeneous motion blur. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 3806–3815 (2017)
15. Goodman, J.: Introduction to Fourier optics. McGraw-Hill (1996)
16. Hirsch, M., Sra, S., Schölkopf, B., Harmeling, S.: Efficient filter flow for space-variant multiframe blind deconvolution. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 607–614 (2010)
17. Hu, Z., Yuan, L., Lin, S., Yang, M.: Image deblurring using smartphone inertial sensors. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 1855–1864 (2016)
18. Kelley, T.: Iterative Methods for Linear and Nonlinear Equations. SIAM (1995)
19. Kim, T.H., Lee, K.M.: Segmentation-free dynamic scene deblurring. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 2766–2773 (2014)
20. Kobler, E., Klatzer, T., Hammernik, K., Pock, T.: Variational networks: Connecting variational methods and deep learning. In: Proceedings of the German Conference on Pattern Recognition. pp. 281–293 (2017)
21. Krishnan, D., Fergus, R.: Fast image deconvolution using hyper-laplacian priors. In: Advances in Neural Information Processing Systems. pp. 1033–1041 (2009)

22. Kruse, J., Rother, C., Schmidt, U.: Learning to push the limits of efficient FFT-based image deconvolution. In: Proceedings of the International Conference on Computer Vision. pp. 4596–4604 (2017)
23. Levin, A., Weiss, Y., Durand, F., Freeman, W.T.: Understanding and evaluating blind deconvolution algorithms. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 1964–1971 (2009)
24. Meinhardt, T., Möller, M., Hazirbas, C., Cremers, D.: Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In: Proceedings of the International Conference on Computer Vision. pp. 1799–1808 (2017)
25. Michaeli, T., Irani, M.: Blind deblurring using internal patch recurrence. In: Proceedings of the European Conference on Computer Vision. pp. 783–798 (2014)
26. Pan, J., Sun, D., Pfister, H., Yang, M.: Deblurring images via dark channel prior. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(10), 2315–2328 (2018)
27. Parikh, N., Boyd, S.P.: Proximal algorithms. Foundations and Trends in Optimization **1**(3) (2014)
28. Richardson, W.H.: Bayesian-based iterative method of image restoration. Journal of the Optical Society of America **62**(1), 55–59 (1972)
29. Roth, S., Black, M.J.: Fields of experts. International Journal on Computer Vision **82**(2) (2009)
30. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D **60**, 259–268 (1992)
31. Schmidt, U., Roth, S.: Shrinkage fields for effective image restoration. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 2774–2784 (2014)
32. Schmidt, U., Rother, C., Nowozin, S., Jancsary, J., Roth, S.: Discriminative non-blind deblurring. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 604–611 (2013)
33. Schuler, C.J., Hirsch, M., Harmeling, S., Schölkopf, B.: Learning to deblur. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(7), 1439–1451 (2016)
34. Starck, J., Murtagh, F.: Astronomical Image and Data Analysis, Second Edition. Astronomy and Astrophysics Library, Springer (2006)
35. Sun, J., Cao, W., Xu, Z., Ponce, J.: Learning a convolutional neural network for non-uniform motion blur removal. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 769–777 (2015)
36. Sun, J., Xu, Z., Shum, H.: Image super-resolution using gradient profile prior. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 1–8 (2008)
37. Sun, L., Cho, S., Wang, J., Hays, J.: Edge-based blur kernel estimation using patch priors. In: Proceedings of International Conference on Computational Photography. pp. 1–8 (2013)
38. Tai, Y., Tan, P., Brown, M.S.: Richardson-lucy deblurring for scenes under a projective motion path. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(8), 1603–1618 (2011)
39. Whyte, O., Sivic, J., Zisserman, A., Ponce, J.: Non-uniform deblurring for shaken images. International Journal on Computer Vision **98**(2), 168–186 (2012)
40. Wiener, N.: The Extrapolation, Interpolation, and Smoothing of Stationary Time Series. John Wiley & Sons, Inc. (1949)
41. Xu, L., Ren, J.S.J., Liu, C., Jia, J.: Deep convolutional neural network for image deconvolution. In: Advances in Neural Information Processing Systems. pp. 1790–1798 (2014)
42. Xu, L., Tao, X., Jia, J.: Inverse kernels for fast spatial deconvolution. In: Proceedings of the European Conference on Computer Vision. pp. 33–48 (2014)
43. Xu, L., Zheng, S., Jia, J.: Unnatural L0 sparse representation for natural image deblurring. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 1107–1114 (2013)

44. Zhang, J., Pan, J., Lai, W., Lau, R.W.H., Yang, M.: Learning fully convolutional networks for iterative non-blind deconvolution. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 6969–6977 (2017)
45. Zhang, K., Zuo, W., Gu, S., Zhang, L.: Learning deep CNN denoiser prior for image restoration. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 2808–2817 (2017)
46. Zoran, D., Weiss, Y.: From learning models of natural image patches to whole image restoration. In: Proceedings of the International Conference on Computer Vision. pp. 479–486 (2011)