DeepRLS: A Recurrent Network Architecture with Least Squares Implicit Layers for Non-blind Image Deconvolution

Iaroslav Koshelev
Skoltech, Moscow
{Iaroslav.Koshelev, Daniil.Selihanovich}@skoltech.ru

Daniil Selihanovich
Skoltech, Moscow

Stamatios Lefkimmiatis
Huawei Noah’s Ark Lab, Moscow
stamatios.lefkimmiatis@huawei.com

Abstract

In this work, we study the problem of non-blind image deconvolution and propose a novel recurrent network architecture that leads to very competitive restoration results of high image quality. Motivated by the computational efficiency and robustness of existing large scale linear solvers, we manage to express the solution to this problem as the solution of a series of adaptive non-negative least-squares problems. This gives rise to our proposed Recurrent Least Squares Deconvolution Network (RLSDN) architecture, which consists of an implicit layer that imposes a linear constraint between its input and output. By design, our network manages to serve two important purposes simultaneously. The first is that it implicitly models an effective image prior that can adequately characterize the set of natural images, while the second is that it recovers the corresponding maximum a posteriori (MAP) estimate. Experiments on publicly available datasets, comparing recent state-of-the-art methods, show that our proposed RLSDN approach achieves the best reported performance both for grayscale and color images for all tested scenarios. Furthermore, we introduce a novel training strategy that can be adopted by any network architecture that involves the solution of linear systems as part of its pipeline. Our strategy eliminates completely the need to unroll the iterations required by the linear solver and, thus, it reduces significantly the memory footprint during training. Consequently, this enables the training of deeper network architectures which can further improve the reconstruction results.

1. Introduction

Image deconvolution belongs to the category of inverse imaging problems [5] and it appears in a host of applications ranging from computational photography and microscopy to remote sensing and astronomical imaging. The goal of deconvolution is to recover the sharp latent image from a blurry and noisy captured version. The blurring effect is typically modeled as the convolution of the underlying image with the point spread function (psf) of the imaging system. It can be caused by several factors including motion during image acquisition, which can be due to camera shake or due to moving objects in the captured scene when long exposure times are used, out-of-focus optics, scattered light distortion in confocal microscopy, atmospheric turbulence in astronomy, etc. [21].

Image deconvolution is a very challenging problem and a plethora of methods have been proposed in the literature to address it. We can classify all these methods into two categories, those that deal with the estimation of both the underlying sharp image and the blur kernel (psf) [9,12,42,49,54] and those that work under the assumption that the blur kernel is given and aim to estimate only the underlying image [13,14,20,29,55]. The methods in the first class are referred to as blind deconvolution methods while those in

Figure 1. Real image deblurring result by the proposed RLSDN network. The blurred input was obtained from [26], while the blur kernel was estimated using the method of [40].

Input / Estimated blur kernel

RLSDN (ours)
the second as non-blind deconvolution methods. In general, though, when dealing with blind deconvolution problems the common strategy is to follow a two phase approach. In the first phase an accurate estimate of the blur kernel is obtained, while in the second phase a non-blind deconvolution method is applied to recover the underlying image (for a comprehensive review of blind methods see [30] and references therein). Therefore, the development of efficient and effective non-blind image deconvolution methods is still of high significance, and is the main focus of this work.

The presence of blur and noise usually leads to an acquired image that has suffered a significant loss of information. As a result the recovery of the sharp underlying image is unattainable without further taking into account additional available information. There are several ways one can incorporate such information and exploit it during the image restoration process. One way to achieve this is using model-based methods that adopt certain image priors, which are able to encode statistical or physical properties of the latent image [12, 32, 52]. Then, the image deconvolution is re-casted to a constrained optimization problem. However, the most recent paradigm, which has shown great potentials, involves deep-learning methods which are able both to implicitly encode prior image information and obtain the deblurred result relying on specific network architectures [13, 14, 29, 42].

In this work, we follow an approach that combines ideas both from model-based and deep-learning methods. In particular, motivated by model-based methods and large-scale efficient optimization techniques we manage to express the solution of the deblurring problem as the solution of a series of adaptive non-negative least-squares (NNLS) problems. Then, based on this result we design a recurrent deconvolution network which can solve conjointly the deblurring problem and lead to state-of-the-art results. Moreover, we adopt a novel network training strategy, which is not specific to our proposed architecture but it applies to any network that involves the solution of a linear system. Our strategy eliminates completely the need to unroll the iterations of the linear solver and as a result it allow us to reduce significantly the memory requirements during the training stage. This makes it possible to use a larger number of network iterations, which lead to improved reconstruction quality.

2. Problem Formulation

To deal with the problem of non-blind image deconvolution we first need to consider the observation (forward) model, which relates the observed blurry and noisy image with the latent image that we aim to restore. In this work we adopt the following linear observation model

\[ y = Hx + n, \]  

(1)

which is the most widely used in the literature and usually can serve as an adequate approximation of the image acquisition process. In the above formula \( y \in \mathbb{R}^M \) and \( x \in \mathbb{R}^N \) represent the vectorized forms of the observed and underlying images, respectively, assuming that they have been raster-scanned using a lexicographical order. Under this notation, \( H \in \mathbb{R}^{M \times N} \), with \( M < N \), is the degradation matrix that models the spatial response of the imaging device, which is responsible for the presence of blur in the observed image. Note that according to the model of Eq. (1), the observed image \( y \) has smaller spatial dimensions than the underlying image \( x \), which translates to \( H \) being a Toeplitz convolution matrix. This is a more realistic assumption than the alternative and frequently used one of considering \( H \) to be a square circulant convolution matrix. Apart from the blur degradation, the image measurements are also perturbed by noise, which hereafter we will assume it to be zero mean i.i.d Gaussian noise of variance \( \sigma^2 \), i.e. \( n \sim \mathcal{N}(0, \sigma^2) \).

The recovery of \( x \) from the distorted measurements \( y \) belongs to the broad class of linear inverse problems [5]. Despite the linear nature of the acquisition process, the image restoration is far from a trivial task. This is due to the presence of noise, whose exact realization is unknown, and the fact that the blurring operator \( H \) in practice is singular. These two factors turn image deblurring to a highly ill-posed problem [21]. This has the implication that a unique solution to the problem does not exist and therefore we cannot solely rely on the image evidence but we further need to take into account a priori information about the solution.

One way to move forward is to adopt a Bayesian approach and seek for the Maximum A Posteriori (MAP) estimate [23]

\[ x^* = \arg \max_{x} \log (p(y|x)) + \log (p(x)), \]

(2)

where \( \log (p(y|x)) \) corresponds to the log-likelihood of the observation \( y \) and \( \log (p(x)) \) is the log-prior of \( x \). Given our initial assumption that the noise perturbing the measurements is i.i.d Gaussian, the problem in (2) can be equivalently reformulated as the minimization problem

\[ x^* = \arg \min_{x} \left( \mathcal{E}(x; y, H) \equiv \frac{1}{2\sigma^2} ||y - Hx||_2^2 + r(x) \right), \]

(3)

where the first term of the objective function \( \mathcal{E}(\cdot) \) corresponds to the negative log-likelihood and the second term corresponds to the negative log-prior. This problem formulation has direct links to variational methods where the first term of the objective can be interpreted as the data-fidelity that quantifies the proximity of the solution to the observation, while the second term, \( r(x) \), amounts to the regularizer, whose role is to promote solutions that exhibit certain favorable image properties.
Under this framework, it becomes apparent that the selection of a proper regularizer (image prior) is of utmost importance and it relates directly to the quality of the reconstruction. This has led to a wide research interest for developing novel ways to effectively model key image properties that can subsequently lead to improved reconstruction results. The majority of the existing regularizers in the literature can be expressed in the following generic form

\[ r(x) = \phi(|Gx|), \quad (4) \]

where \( G : \mathbb{R}^N \rightarrow \mathbb{R}^{F \times D} \) is a linear operator that acts on the latent image \( x \) and maps it to a linear space of \( F \) features of \( D \) dimensions each, which is also referred to as the regularization operator, \( \phi : \mathbb{R}^{F \times D} \rightarrow \mathbb{R}_+ \) is a non-decreasing potential function which penalizes the response of the operator \( G \) on \( x \), while the operation \( | \cdot | \) is meant to act element-wise.

In the recent past, very popular choices for the regularization operator have been first and second order differential operators such as the gradient [45], the structure tensor [33], the Laplacian and the Hessian [32, 34], wavelet-like operators such as wavelets, curvelets and ridgelets (see [18] and references therein), and learned convolution operators [44], while for the potential function the predominant choice had been the squared \( \ell_2 \) norm, which leads to the well known Tikhonov regularization strategy [21]. The reason behind the strong preference in using the squared \( \ell_2 \) norm has been that in this case the entire objective function is quadratic and thus computationally efficient linear solvers can be employed to obtain the solution. Indeed, the minimizer of a quadratic objective function can be derived as the solution of the corresponding normal equations.

However, a significant drawback of using this potential function in the regularizer is the extensive over-smoothing that the resulting reconstructed images typically exhibit. Nowadays, it is widely acknowledged that employing different and more expressive potential functions, such as \( \ell_p \) norms or pseudo-norms with \( 0 \leq p < 1 \) or the logarithm, can lead to sharper and higher-quality results [2, 17, 28, 53]. Nevertheless, one great challenge that arises with the use of alternative potential functions is that the solution cannot anymore be obtained by simply solving a system of linear equations and more advanced optimization techniques are needed. Indeed, there is a variety of existing strategies to deal with the resulting objective functions, such as FISTA [4], Split Bregman [19], Alternating Method of Multipliers [7], HQS [38] just to name a few. The common underlying idea behind all these optimization strategies is that in order to find a minimizer which corresponds to the reconstructed image, instead of directly dealing with the original minimization problem of Eq. (3), we consider several easier to solve problems.

From the previous discussion it becomes clear that in order to be in position of obtaining a satisfactory solution to the image deblurring problem, first we have to address two important issues. The first one is the selection of an appropriate regularizer, by wisely choosing the regularization operator and the potential function. This will allow us to promote meaningful solutions that exhibit key properties adequately describing the set of natural images. The second challenge is to come up with an optimization strategy that can find such solutions in a computationally efficient way.

In Sec. 3 we focus on the design of an optimization strategy that can efficiently deal with objective functions of the form provided in Eq. (3), while in Sec. 4 we describe how we avoid to specify the exact form of the image regularizer and instead implicitly model it using a novel network architecture.

### 3. Image Restoration via Fixed Point Iteration

There are two key difficulties in the minimization of the objective function in Eq. (3). The first one is the coupling that exists between the singular convolution degradation operator \( H \) and the latent image \( x \). The second one is that the regularizer \( r(x) \), as defined in Eq. (4), has typically a non-quadric form. These two factors prevent us from aiming for a direct solution. Thus, we can only opt for an iterative-based minimization strategy. Now, if we assume that the potential function \( \phi \) is smooth and \( x \) doesn’t belong to the null space of the regularization operator \( G \), then we can compute the gradient of the regularizer as:

\[
\nabla r(x) = G^T \text{diag}(\text{sgn}(Gx)) \nabla \phi(|Gx|) \\
= G^T \text{diag}(\nabla \phi(|Gx|)) \text{diag}(|Gx|)^{-1} Gx \\
= G^T W(Gx) Gx,
\]

(5)

where we use the notation \( W(Gx) \) to denote the diagonal and positive semi-definite matrix that has a direct dependency on \( Gx \).

Next, it is straightforward to show that \( x^* \) is a minimizer (stationary point) of the objective function \( E(x; y, H) \) if it holds:

\[
\frac{1}{\sigma^2} H^T (Hx^* - y) + G^T W(Gx^*) Gx^* = 0.
\]

(6)

Therefore, in order to find the solution to our problem, it is sufficient to solve a system of non-linear equations as shown in Eq. (6). By carefully inspecting the above system of equations, we observe that its non-linear nature stems exclusively from the dependency of the matrix \( W \) on \( x^* \). This suggests that we can use the following fixed-point iteration strategy:

\[
\frac{1}{\sigma^2} H^T (Hx^{k+1} - y) + G^T W Gx^{k+1} = 0,
\]

(7)

\[\text{Note that since the potential functional } \phi \text{ is non-decreasing, it’s gradient will consist of non-negative values, which in turn implies that the diagonal matrix } W(Gx) \text{ will be positive semi-definite.}\]
with \( W^k \equiv W (Gx^k) \), which in turn implies that the solution can be obtained through a sequence of updates of the form:

\[
x^{k+1} = \arg \min_x \frac{1}{\sigma^2} \|y - Hx\|_2^2 + \|Gx\|_W^2 = \arg \min_x \frac{1}{\sigma^2} \|y - Hx\|_2^2 + \|W^{1/2} Gx\|_2^2,
\]

where \( \|x\|_A^2 \equiv x^T A x \), with \( A \succeq 0 \). The update rule provided in Eq. (8) is reminiscent of the classical Tikhonov-regularized solution [21], with the regularization operator selected to be of the form \( W^{1/2} G \). The only difference is that in our case the regularization operator is not fixed but it changes in every iteration according to the solution of the previous iteration. An additional improvement to the previous update rule, which leads to a more stable and robust iterative strategy, is to include an extra term that enforces the solution of the current iteration to be not too far from the previous one. According to this reasoning, we modify Eq. (8) to be of the form:

\[
x^{k+1} = \arg \min_x \frac{1}{\sigma^2} \|y - Hx\|_2^2 + \|Gx\|_W^2 + \alpha \|x - x^k\|_2^2,
\]

where \( \alpha \) is a positive constant. We note that the addition of the last term doesn’t affect the final solution, since upon convergence of the algorithm to a fixed point, it will hold that \( x^{k+1} = x^k \) and thus the extra term will become zero.

Finally, based on all the above we end up with an iterative optimization strategy that allows us to solve the original minimization problem of interest by solving a sequence of non-negative least squares (NNLS) problems, whose solutions can be computed as

\[
x^{k+1} = (S^k \equiv \frac{1}{\sigma^2} H^T H + G^T W^k G + \alpha I)^{-1} \tilde{y}^k,
\]

with \( \tilde{y}^k = \frac{1}{\sigma^2} H^T y + \alpha x^k \). Compared to other alternative minimization strategies, which also attack the problem by splitting it in a series of simpler sub-problems, our proposed approach has an important advantage, since it relies solely on existing efficient and fast matrix-free linear solvers designed for large scale problems. More specifically, given that the corresponding system matrix is symmetric and positive definite we can readily use the Conjugate Gradient (CG) method [46] and its variants, which are specifically designed for this task and have been shown to be robust and very efficient. It is also worth noting that our strategy has close ties to the Iterative Reweighted Least Squares (IRLS) method [11], which has been the minimization strategy of choice in the field of compressive sensing [8, 15] and matrix completion [24]. In fact, our strategy can be considered as an extension of IRLS to the case of regularizers with more generic form than \( \ell_p \) norms.

![Figure 2. Architecture of the proposed Recurrent Least Squares Deconvolution Network. The proposed network consists of three main components: (a) The linear feature extraction network, (b) the weight prediction network, and (c) the non-negative least squares implicit layer. Since the network is recurrent, the output of the current iteration along with the blurry input \( y \) serve as the inputs of the network in the next iteration.](image-url)

**4. Deep Recurrent Least Squares Network**

Until now we have refrained entirely from discussing specific choices for the linear operator \( G \) and the potential function \( \phi \), which constitute the image regularizer and as mentioned earlier play a crucial role in the reconstruction quality. In fact, the main idea that we explore in this work is that instead of manually selecting these parameters, we consider them as parameters of a network and learn them during training.

To design such a network, we rely on the update rule of Eq. (10) and we end up with the recurrent architecture depicted in Fig. 2. Our proposed network consists of three main components: (a) The linear feature extraction module, which initially accepts as input either some estimate of the reconstructed image or the blurry image itself, while in the subsequent iterations it accepts as input the output of the network from the previous iteration. This linear sub-network essentially constitutes a parametrization of the regularization operator \( G \). (b) The weight prediction network (WPN) which acts on the output of the feature extraction module. The role of this sub-network is to predict the diagonal positive semi-definite matrix \( W^k \), which as described in Eq. (7) has a direct dependency on \( x^k \), and according to Eq. (8) is used to weight the regularization operator \( G \). WPN is implemented as an image-to-image network with the additional constraint that its output is non-negative, so as to ensure the positive semi-definite nature of \( W^k \). We note that WPN models the potential function \( \phi \) in an implicit way. Indeed from Eq. (5) we see that \( W^k \) is defined as \( W^k = \text{diag} \left( \nabla \phi (Gx^k) \right) \), with the division being applied element-wise, and involves the gradient of the potential function. (c) The NNLS layer, whose role is to refine the current estimate of the reconstructed image by solving a NNLS problem according to Eq. (10).

One important point regarding the NNLS layer is that in real applications the inversion of the system matrix \( S^k \) in Eq. (10) is practically infeasible. Therefore we need to rely on a matrix-free large-scale linear solver such as CG.
Algorithm 1: Back-propagation for an implicit layer whose output is the solution of a linear system.

Layer’s parameters: $w$

Forward Pass
1. Compute $x^*$ as the solution of the linear system:
$$A(w)x = b(w).$$

Backward Pass
1. Use $x^*$ as the input to an auxiliary residual layer with parameters $w$ and compute its output as:
$$r = b(w) - A(w)x^*.$$
2. Compute $g$ by solving the linear system
$$A^T(w)g = \rho,$$
where $\rho = \nabla_w \mathcal{L}$ and $\mathcal{L}$ is the training loss function.
3. Obtain the gradient $\nabla_w \mathcal{L}$ by computing the product $\nabla_w A^T g$ using any of the existing autograd libraries (pytorch, tensorflow, etc.).
4. Use $\nabla_w \mathcal{L}$ to update the layer’s parameters $w$.

However, since there is a variety of existing linear solvers that we could use, we implement our layer as an implicit one [3]. The key difference of implicit layers over conventional explicit layers, which are usually employed in deep learning, is that instead of having to explicitly specify a set of operations that the layer needs to perform to its input in order to produce the output, we only need to specify a set of constraints that the input and the output should satisfy. Then we are free to use any algorithm among the available ones that can enforce the desired constraints imposed by the layer. This is a rather different approach and leads to more flexible layers that offer one additional level of abstraction. In our case, the constraint imposed by the NNLS layer can be expressed as:
$$g(x^{k+1}, x^k, y) = S^k x^{k+1} - \frac{1}{\sigma^2} H^T y - \alpha x^k = 0,$$ (11)
where $y$ and $x^k$ are the inputs and $x^{k+1}$ is the output of the layer, while $S^k$ is a short-hand notation for the system matrix that appears in Eq. (10).

5. Network Training

As we have discussed earlier, the output of our network is obtained by finding the minimizers of a sequence of NNLS problems, which boils down to computing the solutions of a sequence of linear problems. A strategy that is commonly adopted in cases of recurrent networks like ours, is to unroll the network using a fixed number of iterations and update the network parameters either by means of back-propagation through time (BPTT) or by its truncated version (TBPTT) [27, 43]. Unfortunately, our proposed architecture is not entirely compatible to such a training strategy. The reason is that apart from the external iterations, our network also involves the internal iterations required by the linear solver in order to compute the solutions of each one of the related linear problems. This means that if we had to unroll both the external and the internal iterations of the network, then we would naturally end up with a very deep architecture. However, unrolling such a deep network would be prohibitive since the overall depth of the network is bounded by the available GPU memory or RAM that is required during its training.

5.1. Implicit Back-Propagation

Fortunately, it is possible to come around this problem and completely avoid the need of unrolling the iterations of each linear solver. Indeed, if we denote by $w$ the set of the network parameters, and consider the generic linear system:
$$A(w)x = b(w),$$ (12)
where we specifically indicate the dependency of the system matrix $A$ and the rhs vector $b$ on $w$, then we can compute the gradient of some loss function $\mathcal{L}$ w.r.t the network parameters $w$, following the steps described in Algorithm 1. In the supplementary material we provide a formal proof that motivates the proposed algorithm. The important implication of using Algorithm 1 as part of our network training strategy, is that now we can use any linear solver running for any required number of iterations without having to worry about saving any intermediate results that would result in high memory utilization during training. Indeed, the only intermediate results we are required to save during the forward pass of the network are the final solutions of each one of the linear systems that we encounter. The cost we pay for this reduction of the memory footprint is that during the backward pass, for each one of the external network iterations we have to compute the forward pass of an auxiliary residual network and solve another linear system. While this can increase the training time, it lifts the memory constraints and allow us to use a larger number of external and internal iterations during training.

6. Experiments and Results

In this section, we discuss the implementation details of our network and report the performed comparisons against recent state-of-the-art deblurring methods on several publicly available datasets. Additional results are provided in the supplementary material.

Table 1. Comparison of RLSDN with state-of-the-art methods on the Levin et al. [35] real grayscale benchmark.

| Method       | PSNR (dB) | SSIM   |
|--------------|----------|--------|
| IRCNN [56]   | 27.05    | 0.8200 |
| RGDN [20]    | 33.57    | 0.9648 |
| FDN [29]     | 36.20    | 0.9811 |
| RLSDN (ours) | **36.92**| **0.9842** |
6.1. Train and test data

For training purposes, we have combined DIV2K [1], and Flickr2K [50] datasets, which allowed us to use 3450 high-resolution source images in total. We used blur kernels of support sizes lying in the range between $13 \times 13$ to $35 \times 35$ pixels, randomly sampled from a Brownian motion model using the procedure proposed in [6]. Ground truth crops of size $128 \times 128$ pixels were selected based on responses of the Laplacian filter in order for each sample to contain high-frequency details. For all target samples, we follow the degradation model in Eq. (1) to obtain the corresponding blurred inputs. We have trained two separate models considering small and high noise level scenarios with standard deviations lying within the range of $[1.0, 3.0]$ for the former and $[11.75, 13.75]$ for the latter.

To show the general applicability of our network, we have conducted a separate experiment considering deblurring of saturated images. For this purpose, we have used a similar procedure as the one proposed in [14] to obtain train samples. Although the degradation model for such cases departs from the adopted model of Eq. (1), we have used the same RLSDN architecture and retrained the network to handle saturated cases with noise levels of standard deviation lying in the range of $[1.0, 3.0]$. By construction, none of our training data intersects with any data we have used for validation and testing purposes.

To validate the performance of our network and compare it against other methods we consider color and grayscale deblurring benchmarks consisting of both synthetic and real images. For synthetically blurred images, a standard dataset has been proposed by Sun et al. [47]. This dataset consists only of grayscale images. In order to be able to compare both for grayscale and color cases, we have used the original color images from [48] and the same blur kernels as
images with saturated pixels.

Table 3. Comparison of RLSDN with state-of-the-art methods on images with saturated pixels.

| Noise | Metrics | IRCNN [56] | RGDN [20] | FDN [29] | DWDN [13] | SVMAP [14] | RLSDN (ours) |
|-------|---------|------------|------------|----------|-----------|------------|--------------|
| 1%    | PSNR    | 29.31      | 26.90      | 32.63    | –         | –          | **33.07**    |
|       | SSIM    | 0.8047     | 0.6216     | 0.8894   | –         | –          | **0.9006**   |
| 5%    | PSNR    | 27.60      | 14.06      | 27.72    | –         | –          | **28.13**    |
|       | SSIM    | 0.7444     | 0.1267     | 0.7348   | –         | –          | **0.7573**   |
| Color | 1% PSNR | 29.68      | 30.98      | 32.51    | 34.09     | 34.36      | **34.64**    |
|       | SSIM    | 0.8311     | 0.8840     | 0.8857   | 0.9197    | 0.9249     | **0.9287**   |
| 5%    | PSNR    | 28.85      | 26.93      | 27.66    | 29.16     | 29.15      | **29.44**    |
|       | SSIM    | 0.7961     | 0.7121     | 0.7292   | 0.7905    | 0.7925     | **0.8077**   |

Table 2. Comparison of RLSDN with state-of-the-art methods on the Sun et al. [47] synthetic benchmark. With – we indicate the methods whose inference code is publicly available only for color images and underperform when applied on grayscale images.

those considered in [47]. Then, we created the test dataset by applying a valid convolution as the blurring operation and two different noise levels corresponding to standard deviations of 1% and 5% of the peak image intensities. To validate all the methods, we follow the protocol proposed by [47] and compute all the metrics by considering the central part of the original image (we discard 50 pixel from each border), so as to avoid the influence of boundary artifacts in the SSIM and PSNR scores.

Several synthetic benchmarks were previously proposed to evaluate deblurring methods on large set of images with saturated pixels [14]. Unfortunately, none of them contain neither publicly released data, nor concrete instructions on how to reproduce them. For this reason we collected a set of 21 images with saturated areas and used those to produce synthetically blurred results using the same kernels as those used above. All synthetic data used for evaluation can be downloaded from [this link](https://example.com).

To compare our approach on real data with uniform blur, we have used a benchmark dataset provided by Levin et al. [35], which consists of optically blurred images and accurate blur kernels. Since the degraded images and corresponding ground truths are not well aligned, we compute the scores using the original Matlab code provided by the authors in [36], which internally performs image alignment.

### 6.2. Model Specification and Training

The concrete implementation of the proposed RLSDN model requires us to parameterize several “free” variables and in particular the regularization operator \( G \), the weight prediction network and the regularization constant \( \alpha \), as well as other hyper-parameters. In all our experiments, we parameterize \( G \) with a valid convolution layer that consists of 128 output channels and filters of size \( 13 \times 13 \). As it was proposed in [31] we further overparameterize this operation by a deep linear network (i.e. a composition of convolution layers with smaller filter sizes) to accelerate the training. We apply spectral normalization to each layer of the deep linear network to stabilize the training and resolve the ambiguity that can be introduced due to the interplay between WPN and \( G \). To parameterize the WPN, we use the RRDB backbone [50] with 128 input and output channels, and we add a final ReLU activation layer to ensure non-negativity of its output. Finally, the regularization constant \( \alpha \) is modeled as \( \alpha = e^\beta \), where \( \beta \) is learned without any restrictions on its values.

We have trained our network by performing 5 steps, with the first one involving a separately learned Wiener filter. Its output \( \mathbf{x}^1 = (\mathbf{H}^\top \mathbf{H} + \sigma^2 \mathbf{G}_w^\top \mathbf{G}_w)^{-1} \mathbf{H}^\top \mathbf{y} \), provides a good initial result that we further refine in the following four adaptive steps. At each iteration, we limit the amount of CG steps to 250 during forward and 500 during backward passes. We also employ an early exit strategy if the relative tolerance of the residual lies below \( 10^{-3} \) for all elements of the batch. During the forward pass, the output from the previous RLSDN recurrent step is used as the initial solution of the linear solver to further reduce the amount of iterations required for convergence.

All our models were learned using back-propagation by minimizing the sum of mean squared errors between ground truth and outputs of each recurrent step. We used Adam optimizer [25] with a learning rate \( 2 \cdot 10^{-4} \) decreasing by a factor of 0.98 after each epoch and a warm-up strategy for the first two epochs. We set the batch size to 32 and trained our models for 100 epochs, where each epoch consists of 2000 batch passes.

### 6.3. Results

#### Synthetic blur

In Tab. 2 we provide quantitative comparisons with state-of-the-art methods for two different noise levels. From the reported results it is clear, that our proposed network outperforms all competing methods by a good margin in both color and grayscale deblurring on small and large noise levels. In particular, for the case of color...
deblurring our method leads to improved results of almost 0.3dB for both 1% and 5% Gaussian noise compared to the recent best performing network [14]. The improved performance is even more pronounced in the case of grayscale image deblurring, where our method outperforms the second best FDN network [29] by nearly 0.4dB in all tested cases. Representative grayscale and color deblurring results that demonstrate visually the restoration quality of the proposed models are shown in Fig. 3.

Comparisons among methods specifically designed for image deblurring of images with saturated pixels are presented in Tab. 3 and follow a similar trend as the previous results. In particular, our network outperforms the second best method [9] by 0.56dB, SVMAP [14] by 2.46dB and the method of [52] by more than 7dB. It is noteworthy to mention that while SVMAP’s network architecture is also inspired by IRLS, it performs significantly worse than our network. Our improved performance can be attributed both to our specific network architecture and our training strategy, which allow us to employ more recurrent steps and linear solver iterations. Indeed, SVMAP consists of two external iterations with five CG iterations each, which typically are not enough for large-scale restoration problems. A visual comparison of the different restored results obtained by three of the methods under consideration is possible by referring to Fig. 5. According to these results we observe that our network leads to a visually better restored image, without the presence of strong artifacts or noise residual. This is not the case both for SVMAP [14] and the method of [9], where the restored image by the former method suffers from strong noise while the image recovered by the latter method is significantly over-smoothed and as a result fine details of the image are failed to be recovered entirely.

Real blur. In Tab. 1 we report results for real grayscale image deblurring. Our comparisons involve only methods that work well on grayscale images and we refrain from reporting results by other methods appearing in Tab. 2 which are designed for color image deblurring and are not producing competitive enough results. Tab. 1 clearly demonstrates the superiority of our approach and its ability to handle real motion blur. Similar to the synthetic case, our network outperforms FDN [29], which is the current state-of-the-art grayscale deblurring method, by 0.72dB. An important observation is that our network, while it is exclusively trained on synthetic examples, performs very well on real images when accurate blur kernels are provided. For a visual inspection of the restoration performance of our grayscale model we refer to Fig. 4. For the most practical cases where an accurate motion trajectory is not available, a third party kernel estimation method such as [40] can be used to complement our network. As it can be seen from Fig. 1, even in cases where only an estimate of the blur kernel is available, our network is still able to restore the image adequately and produce results of good visual quality.

7. Conclusions and Future Work

In this work we introduced a novel recurrent network architecture that we deploy to deal with the task of non-blind image deconvolution. The design of the proposed network is inspired by a large-scale fixed-point iteration method that we developed and it amounts to solving a series of adaptive non-negative least squares problems. This approach coupled with a new training strategy that eliminates the need to unroll the involved iterations of the linear solver leads to a very effective deconvolution network that produces state-of-the-art results under different scenarios, including noise of low and high levels and images with saturated pixels.

While fixed-point iteration methods are frequently met in optimization, unless they satisfy certain conditions they might diverge. Since our proposed network architecture implements a particular fixed-point iteration strategy, not all of its possible configurations are guaranteed to always converge, irrespectively of the input data. Based on this, an interesting future research direction that we plan to pursue is to investigate possible ways to ensure that the learned network parameters will lead to a configuration with provable convergence guarantees. This is a very important property for a network to possess and can be extremely useful in practical applications. Another possible direction is to investigate the applicability of our network architecture to other inverse imaging problems such as demosaicking and image super-resolution.
Appendix

A. Implicit Back-Propagation for Linear Systems

Let \( A \in \mathbb{R}^{n \times m} \) be a non-singular matrix and \( b \in \mathbb{R}^n \) a vector. Both \( A \) and \( b \) depend on a parameter vector \( w \in \mathbb{R}^m \) and form the following linear system:

\[
A(w) x = b(w).
\]

Let us also consider an implicit network layer whose parameters correspond to the latent vector \( w \) and its output is given as:

\[
\hat{x} = A^{-1}(w) b(w).
\]

In order to learn the layer’s parameters using back-propagation, we need to be able to compute the gradient of its output w.r.t \( w \). To do so, first we compute the differential of \( \hat{x} \) as:

\[
d\hat{x} = (dA^{-1}(w)) b(w) + A^{-1}(w) db(w)
= -A^{-1}(w) (dA(w)) A^{-1}(w) b(w) + A^{-1}(w) db(w)
= -A^{-1}(w) (dA(w)) \hat{x} + A^{-1}(w) db(w)
= - (\hat{x}^T \otimes A^{-1}(w)) \text{vec} (dA(w)) + A^{-1}(w) db(w)
= - (\hat{x}^T \otimes A^{-1}(w)) \frac{\partial \text{vec}(A(w))}{\partial w} dw.
\]

where we have used the property \( dA^{-1} = -A^{-1} (dA) A^{-1} \) \(^{[37]}\) and the vec identity \( \text{vec}(ABC) = (C^T \otimes A) \text{vec}(B) \) with \( A, B, C \) being proper-sized matrices and \( \otimes \) denoting the Kronecker product.

From Eq. (15) and using the identification theorem \[^{[37]}\] we can compute the gradient of \( \hat{x} \) w.r.t \( w \) as:

\[
\nabla_w \hat{x} = -\nabla_w \text{vec}(A(w)) (\hat{x} \otimes A^{-T}(w))
+ \nabla_w b(w) A^{-T}(w).
\]

Next, let us compute the gradient of a scalar loss function \( L : \mathbb{R}^n \rightarrow \mathbb{R} \) w.r.t the parameter vector \( w \) given that \( \nabla_{\hat{x}} L = \rho \in \mathbb{R}^n \). Using the chain rule and Eq. (16), we get:

\[
\nabla_w L = \nabla_{\hat{x}} \hat{x} \nabla_{\hat{x}} L
= \left[ -\nabla_w \text{vec}(A(w)) (\hat{x} \otimes A^{-T}(w))
+ \nabla_w b(w) A^{-T}(w) \right] \rho
= -\nabla_w \text{vec}(A(w)) (A^{-T}(w) \rho \hat{x}^T)
+ \nabla_w b(w) A^{-T}(w) \rho
= -\nabla_w \text{vec}(A(w)) (g \hat{x}^T) + \nabla_w b(w) g,
\]

where \( g = A^{-T}(w) \rho \) and can be efficiently computed by using a “matrix-free” linear solver.

According to Eq. (17), in order to compute the exact expression of \( \nabla_w L \) we first need to be able to compute the gradients of \( \text{vec}(A(w)) \) and \( b(w) \) w.r.t \( w \). In certain cases, these gradients can be challenging to be derived in analytical form. This mostly depends on the specific structure of \( A \) and \( b \) and their dependency on \( w \). Moreover, in most of the cases we don’t have access to the individual elements of the matrix \( A \), but rather to the product of \( A \) with a vector. Fortunately, as we describe next, there is a workaround which allows us to avoid the explicit computation of these gradients and instead rely on their computation using an auto-grad library. Specifically, let us consider the residual vector

\[
r = b(w) - A(w) \hat{x}
\]

and let us compute its differential as:

\[
dr = db(w) - d(A(w)) \hat{x}
= db(w) - (\hat{x}^T \otimes I_n) \text{vec} (d(A(w)))
= \frac{\partial b(w)}{\partial w} dw - (\hat{x}^T \otimes I_n) \frac{\partial \text{vec}(A(w))}{\partial w} dw.
\]

Now, from Eq. (19) and the identification theorem we can compute the gradient of \( r \) w.r.t \( w \) as:

\[
\nabla_w r = \nabla_w b(w) - \nabla_w \text{vec}(A(w)) (\hat{x} \otimes I_n).
\]

Using Eq. (20) we can show that:

\[
(\nabla_w r) g = \nabla_w b(w) g - \nabla_w \text{vec}(A(w)) (\hat{x} \otimes I_n) g
= \nabla_w b(w) g - \nabla_w \text{vec}(A(w)) \text{vec}(g \hat{x}^T).
\]

Comparing Eqs. (17) and (21) we finally have that:

\[
\nabla_w L = (\nabla_w r) g = (\nabla_w r) A^{-T}(w) \nabla_{\hat{x}} L.
\]

The above equality suggests the following algorithmic steps for computing \( \nabla_w L \), given \( \rho = \nabla_{\hat{x}} L \):

1. **Forward Pass**
   Compute the solution \( \hat{x} \) by solving the linear system \( A(w) x = b(w) \).

2. **Backward Pass**
   (a) Use \( \hat{x} \) as the input of an auxiliary network that produces the output
   \[
r = b(w) - A(w) \hat{x}.
   \]
   (b) Compute \( g \) by solving the linear system \( A^T(w) g = \rho \).
   (c) Obtain the gradient \( \nabla_w L \) by computing the product \( (\nabla_w r) g \). This last product can be computed automatically by any of the existing auto-grad libraries with \( g \) being the incoming gradient in the auxiliary network that produces the output \( r \).
B. Ablation Study

B.1. Convergence to a fixed point

In this section we provide some empirical evidence that our trained models have learned a mapping that converges to a fixed point. In order a fixed point iteration method to enjoy theoretical convergence guarantees, it is sufficient to show that the mapping of the input to the output is contractive, for any input. We note that in our case we do not impose any constraints on the learned parameters that can enforce our network to have this property. Nevertheless, from the results we report below we observe that in practice our trained networks indeed reach a fixed point in all cases and do not diverge.

To investigate, how the network’s performance evolves with the amount of recurrent steps performed, we have run our model for 20 steps and evaluated the PSNR of the output from each step on the whole Sun et al. color dataset with noise level of 1%. Moreover, we performed several runs where we used different settings for the linear solver. In particular, we investigated the achieved performance when the maximum allowed CG iterations vary between 25 to 500. The convergence behavior of all the different network configurations is depicted in Fig. 6. From this figure it is clearly observed that in all the different cases the proposed RLSDN network manages to reach an equilibrium (the final output is close to a fixed point of the network). The exact amount of recurrent steps needed for convergence depends on the maximum allowed number of CG iterations. The higher the CG iterations the smaller the recurrent steps for which the network converges. It is also important to note that according to the reported results, by limiting the amount of CG iterations to 75 the performance of our network at the 5th step and beyond does not degrade comparing to the case where 500 CG iterations are used. This is an important empirical evidence which suggests that we can speed-up the inference without suffering any significant loss in reconstruction quality. Note that all the results we reported in the paper correspond to those obtained by using 5 recurrent steps with maximum 250 CG iterations per step. We also provide a visual example in Fig. 8 that further supports our claims regarding convergence to a fixed point. While we found the presented results valid for all our trained networks, in this figure we depict the specific example from our saturated images dataset and per step PSNR evolution of the RLSDN network trained for scenario of color saturated image deblurring with 1% noise. We also provide the per step relative error, which is computed as \( tol = \frac{||x_i - x_{i-1}||}{||x_i||} \).

From Fig. 8 we notice the same behaviour as in Fig. 6, where the output of our network stabilizes at around the 4th step, and then approximately reaches an equilibrium. Finally, we observe that the relative error decreases monotonically up to the 20th step.

B.2. Adaptive complexity of RLSDN

A natural advantage of iterative algorithms is that they adapt their complexity according to the type and level of distortions of the input. Specifically, iterative methods tend to converge in less iterations when the degradation of the input is small, while they need more iterations when the degradation is excessive. To investigate whether this is also the case for the RLSDN network, we have used an image from the Sun et al. dataset, distorted it with 1% noise and 5% noise and studied the amount of CG iterations needed
for convergence. We have used an early exit strategy when the relative error during CG iterations falls below $1 \cdot 10^{-4}$ and limit the maximum amount of iterations to 250. We report our findings regarding convergence in Fig. 7. From this figure it can be seen that for the case of 1% noise RLSDN indeed converges faster, performing less amount of internal CG iterations than for the 5% noise case. The faster convergence is clearly visible from the first plot of Fig. 7, as the RLSDN relative error for the 5% case lies above the 1% case and catches up only at the 17th step. The same can be seen from the bottom plot, which depicts the CG relative error. For the 5% noise case the relative error is an order of magnitude larger than for the 1% case. Indeed, as it can be seen from the central plot, for the case of 1% noise CG performs an early exit at almost every RLSDN step, while for the 5% noise case 250 iterations seems not to be enough to converge at any RLSDN step. From this study we can conclude that for the images presented in Fig. 7 in order for RLDN to converge to a relative tolerance of $1 \cdot 10^{-3}$, it requires to perform 8 steps with 761 CG iterations in total for the 1% noise case, while for the 5% noise case it converges in 16 steps and 4000 CG iterations in total.

It is important to note, that the adaptive behaviour of RLSDN stems directly from our proposed architecture and our adopted training strategy which employs implicit back-propagation. In turn, this leads to a significant boost in restoration quality, which is impossible to achieve with traditional feed-forward convolution neural networks (CNN). Indeed, under the classical deep learning training strategy, when unrolled, our model consists of a sequence of CG iterations roughly represented by the following blocks: convolution layer, point-wise multiplication, transpose convolution layer, skip-connection. As discussed in the previous paragraph, in order to converge for a hard case scenario, our network requires 4000 CG iterations, which corresponds to 8000 convolution layers in total. Compared to a large ResNet-150 architecture [22] with 150 convolution layers in total, where also a block of two convolution layers is followed by a skip connection, our approach represents 50 times deeper neural network for a hard case scenario, and 5 times deeper network for an easy case. Clearly, due to memory constraints, it is extremely difficult to train a ResNet-8000, while our network can be learned in an end-to-end fashion without any problems.

C. Real Color Image Deblurring Comparisons

In this section we present more visual comparisons of our proposed models with current state-of-the-art deblurring methods for real examples. We present the most interesting practical case of real image deblurring, when the blur kernel is estimated by a third-party method. To evaluate the performance of RLSDN we use a collection of three images and kernels provided by different methods and we present the results in Fig. 9. For a fair comparison, we do not tune the noise standard deviation for all methods requiring it as an additional input (including ours), but instead use the value predicted by the WMAD estimation method [16]. As it can be seen from the presented examples, compared to the rest of the methods our network restores finer details in a more accurate manner without amplifying the noise or exhibiting an over-smoothing effect.

D. Impact of the blur kernel

All the above results suggest that our proposed network leads to very competitive performance for different scenarios. However, a limiting factor to our approach, similarly to the rest of non-blind deconvolution methods, is that our network relies on a third-party blur kernel estimation method. Consequently, when the blur kernel is poorly estimated this has immediate effect in the reconstruction result. This is depicted in Fig. 10, where we observe that when the estimated kernel is relatively accurate our network is able to produce a high quality image reconstruction, while when the used blur kernel is inaccurate the quality of the deblurred image can drop significantly.
Figure 9. Visual comparison of RLSDN with state-of-the art methods on real color blurred images with blur kernels estimated by a third-party estimation method. First row: image and kernel from [40], middle row: image and kernel from [39], bottom row: image from [30], kernel estimated by [40].

Figure 10. Performance of our method on a real image (“Roma” from [30]) suffering from extensive blur. Two existing kernel estimation methods have been used in order to estimate the blur kernel. From the result depicted in (b) it is clear, that our network is capable of producing deblurring results of high quality when an accurate enough blur kernel is provided. However, when the kernel estimation fails, then the quality of the reconstruction result can be very poor as shown in (c).
References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, July 2017. 6
[2] D. Babacan, M. N. Molina, R. Do, and A. K. Katsaggelos. Bayesian blind deconvolution with general sparse image priors. In European Conference on Computer Vision, pages 341–355, 2012. 3
[3] Shaogjei Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In Advances in Neural Information Processing Systems, volume 32, 2019. 5
[4] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci., 2:183–202, 2009. 3
[5] Mario Bertero and Patrizia Boccacci. Introduction to Inverse Problems in Imaging. IOP Publishing, 1998. 1, 2
[6] Giacomo Boracchi and Alessandro Foi. Modeling the performance of image restoration from motion blur. IEEE Transactions on Image Processing, 21(8):3502–3517, 2012. 6
[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. Now Publishers, 2011. 3
[8] Emmanuel J. Candés and Michael B. Wakin. An introduction to compressive sensing. IEEE Signal Processing Magazine, 25(2):21–30, 2008. 4
[9] Sunghyun Cho and Seungyong Lee. Fast motion deblurring. In ACM SIGGRAPH Asia 2009 papers, pages 1–8. 2009. 1, 8
[10] Sunghyun Cho, Jue Wang, and Seungyong Lee. Handling outliers in non-blind image deconvolution. In 2011 International Conference on Computer Vision, pages 495–502. IEEE, 2011. 7, 8
[11] Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C Sinan Güntürk. Iteratively reweighted least squares minimization for sparse recovery. Communications on Pure and Applied Mathematics, 63(1):1–38, 2010. 4
[12] Jiangxin Dong, Jinshan Pan, Zhixun Su, and Ming-Hsuan Yang. Blind image deblurring with outlier handling. In Proceedings of the IEEE International Conference on Computer Vision, pages 2478–2486, 2017. 1, 2
[13] Jiangxin Dong, Stefan Roth, and Bernt Schiele. Deep wiener deconvolution: Wiener meets deep learning for image deblurring. arXiv preprint arXiv:2103.09962, 2021. 1, 2, 6, 7, 12
[14] Jiangxin Dong, Stefan Roth, and Bernt Schiele. Learning spatially-variant map models for non-blind image deblurring. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4886–4895, June 2021. 1, 2, 6, 7, 8, 12
[15] David L. Donoho. Compressed sensing. IEEE Transactions on information theory, 52(4):1289–1306, 2006. 4
[16] David L. Donoho and Iain M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. Biometrika, 81(3):425–455, 1994. 11
[17] Rob Fergus, Barun Singh, Aaron Hertzmann, Sam T Roweis, and William T Freeman. Removing camera shake from a single photograph. In ACM SIGGRAPH, pages 787–794. 2006. 3
[18] M.A.T. Figueiredo, J.M. Bioucas-Dias, and R.D. Nowak. Majorization–minimization algorithms for wavelet-based image restoration. IEEE Trans. Image Process., 16:2980–2991, 2007. 3
[19] T. Goldstein and S. Osher. The split Bregman method for L1-regularized problems. SIAM J. Imaging Sci., 2:323–343, 2009. 3
[20] Dong Gong, Zhen Zhang, Qinfeng Shi, Anton van den Hengel, Chunhua Shen, and Yanning Zhang. Learning deep gradient descent optimization for image deconvolution. IEEE transactions on neural networks and learning systems, 31(12):5468–5482, 2020. 1, 5, 6, 7, 12
[21] P. C. Hansen, J. G. Nagy, and D. P. O’Leary. Deblurring Images: Matrices, Spectra, and Filtering. SIAM, 2006. 1, 2, 3, 4
[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition, 2016. 11
[23] Steven M Kay. Fundamentals of Statistical Signal Processing: Estimation Theory. Prentice-Hall, Inc., 1993. 2
[24] Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. IEEE transactions on information theory, 56(6):2980–2998, 2010. 4
[25] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014. 7
[26] Rolf Köhler, Michael Hirsch, Betty Mohler, Bernhard Schölkopf, and Stefan Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In Computer Vision – ECCV 2012, pages 27–40, 2012. 1
[27] Filippos Kokkinos and Stamatios Lefkimmiatis. Iterative joint image demosaicking and denoising using a residual denoising network. IEEE Transactions on Image Processing, 28(8):4177–4188, 2019. 5
[28] Dilip Krishnan, Terence Tay, and Rob Fergus. Blind deconvolution using a normalized sparsity measure. In Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition, pages 233–240. IEEE, 2011. 3
[29] Jakob Kruse, Carsten Rother, and Uwe Schmidt. Learning to push the limits of efficient fft-based image deconvolution. In Proceedings of the IEEE International Conference on Computer Vision, pages 4586–4594, 2017. 1, 2, 5, 6, 7, 8, 12
[30] Wei-Sheng Lai, Jia-Bin Huang, Zhe Hu, Narendra Ahuja, Jakob Kruse, Carsten Rother, and Uwe Schmidt. Learning to deconvolve: Wiener meets deep learning for image deblurring. In IEEE Conf. Comput. Vision and Patt. Recogn. (CVPR), pages 1701–1709. IEEE, 2016. 2, 12
[31] Thomas Laurent and James Brecht. Deep linear networks with arbitrary loss: All local minima are global. In International Conference on Machine Learning, pages 2902–2907. PMLR, 2018. 7
[32] S. Lefkimmiatis, A. Bourquard, and M. Unser. Hessian-based norm regularization for image restoration with biomedical applications. *IEEE Trans. Image Process.*, 21(3):983–995, 2012. 2, 3

[33] S. Lefkimmiatis, A. Roussos, P. Maragos, and M. Unser. Structure tensor total variation. *SIAM J. Imaging Sci.*, 8:1090–1122, 2015. 3

[34] S. Lefkimmiatis, J. Ward, and M. Unser. Hessian Schatten-norm regularization for linear inverse problems. *IEEE Trans. Image Process.*, 22(5):1873–1888, 2013. 3

[35] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *IEEE Conf. Comput. Vision and Patt. Recogn. (CVPR)*, pages 1964–1971, 2009. 5, 6, 7

[36] Anat Levin, Yair Weiss, Fredo Durand, and William T Freeman. Efficient marginal likelihood optimization in blind deconvolution. In *CVPR 2011*, pages 2657–2664. IEEE, 2011. 7

[37] Jan R Magnus and Heinz Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 1999. 9

[38] Mila Nikolova and Michael K Ng. Analysis of half-quadratic minimization methods for signal and image recovery. *SIAM Journal on Scientific computing*, 27(3):937–966, 2005. 3

[39] Jinshan Pan, Zhouchen Lin, Zhixun Su, and Ming-Hsuan Yang. Robust kernel estimation with outliers handling for image deblurring. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2800–2808, 2016. 12

[40] Jinshan Pan, Deqing Sun, Hanspeter Pfister, and Ming-Hsuan Yang. Blind image deblurring using dark channel prior. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1628–1636, 2016. 1, 8, 12

[41] Daniele Perrone and Paolo Favaro. Total variation blind deconvolution: The devil is in the details. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2909–2916, 2014. 12

[42] Dongwei Ren, Kai Zhang, Qilong Wang, Qinghua Hu, and Wangmeng Zuo. Neural blind deconvolution using deep priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3341–3350, 2020. 1, 2

[43] A. J. Robinson and Frank Fallside. The Utility Driven Dynamic Error Propagation Network. Technical Report CUED/F-INFENG/TR.1, Engineering Department, Cambridge University, Cambridge, UK, 1987. 5

[44] Stefan Roth and Michael J Black. Fields of experts. *International Journal of Computer Vision*, 82(2):205–229, 2009. 3

[45] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992. 3

[46] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994. 4

[47] Libin Sun, SungHyun Cho, Jue Wang, and James Hays. Edge-based blur kernel estimation using patch priors. In *Proc. IEEE International Conference on Computational Photography*, 2013. 6, 7

[48] Libin Sun and James Hays. Super-resolution from internet-scale scene matching. In *Proceedings of the IEEE Conf. on International Conference on Computational Photography (ICCP)*, 2012. 6

[49] Phong Tran, Anh Tran, Quyhn Phung, and Minh Hoi. Explore image deblurring via encoded blur kernel space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[50] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018. 6, 7

[51] Fei Wen, Rendong Ying, Yipeng Liu, Peilin Liu, and Trieu-Kien Truong. A simple local minimal intensity prior and an improved algorithm for blind image deblurring. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(8):2923–2937, 2021. 12

[52] Oliver Whyte, Josef Sivic, and Andrew Zisserman. Deblurring shaken and partially saturated images. *International journal of computer vision*, 110(2):185–201, 2014. 2, 7, 8

[53] Li Xu, Shicheng Zheng, and Jiaya Jia. Unnatural l0 sparse representation for natural image deblurring. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 1107–1114, 2013. 3

[54] Xiangyu Xu, Jinshan Pan, Yu-Jin Zhang, and Ming-Hsuan Yang. Motion blur kernel estimation via deep learning. *IEEE Transactions on Image Processing*, 27(1):194–205, 2017. 1

[55] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Trans. Image Process.*, 2017. 1

[56] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep CNN denoiser prior for image restoration. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, July 2017. 5, 6, 7, 12