Photon Limited Non-Blind Deblurring Using Algorithm Unrolling

Yash Sanghvi, *Student Member, IEEE*, Abhiram Gnanasambandam, *Student Member, IEEE*, and Stanley H. Chan, *Senior Member, IEEE*[*]

***Abstract*—Image deblurring in photon-limited conditions is ubiquitous in a variety of low-light applications such as photography, microscopy and astronomy. However, the presence of photon shot noise due to low-illumination and/or short exposure makes the deblurring task substantially more challenging than the conventional deblurring problems. In this paper we present an algorithm unrolling approach for the photon-limited deblurring problem by unrolling a Plug-and-Play algorithm for a fixed number of iterations. By introducing a three-operator splitting formation of the Plug-and-Play framework, we obtain a series of differentiable steps which allows the fixed iteration unrolled network to be trained end-to-end. The proposed algorithm demonstrates significantly better image recovery compared to existing state-of-the-art deblurring approaches. We also present a new photon-limited deblurring dataset for evaluating the performance of algorithms.**

***Index Terms*—photon limited, Poisson deconvolution, deblurring, Plug-and-Play, algorithm unrolling**

I. Introduction

Image deblurring is a classical restoration problem where the goal is to recover a clean image from an image corrupted by a blur due to motion, camera shake, or defocus. In the simplest setting assuming a spatially invariant blur, the forward image degradation problem is

$$\boldsymbol{y} = \boldsymbol{h} * \boldsymbol{x} + \boldsymbol{\eta}, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^N$ is the clean image to be recovered from the corrupted image $\boldsymbol{y} \in \mathbb{R}^N$, the vector $\boldsymbol{h} \in R^d$ denotes the blur kernel, $\boldsymbol{\eta} \in \mathbb{R}^N$ denotes the additive i.i.d Gaussian noise, and "$*$" denotes the convolution operator. The deblurring problem can be further classified as *non-blind* and *blind*. A non-blind deblurring problem assumes that the blur kernel $\boldsymbol{h}$ is known whereas a blind-deblurring problem do not make such an assumption. In this paper, we focus on the non-blind case.

While non-blind deblurring methods are abundant [1]–[6], the majority are designed for well-illuminated scenes where the noise is i.i.d. Gaussian and the noise level is not too high. However, as one pushes the photon level low enough that the photon shot noise dominates, the deblurring task is no longer as simple. As illustrated in Figure 1 which is a real low-light example we captured using a Canon T6i camera at a photon level approximately 5 lux, the observed image is not only dark but is strongly contaminated by photon shot noise (which is visible in the histogram equalized image). Given the blur kernel, our goal is to recover the image.

The operating regime of the proposed method is illustrated in Figure 2 we use another real low-light image to compare this paper and other mainstream deblurring algorithms. We highlight the raw sensor capture (shown in the bottom left of each sub-figure) and the tone-mapped image (shown in the top right of each sub-figure) at different illumination levels. Our algorithm is specifically designed for an illumination level of 1 lux or lower.

Under such a severe lighting condition, state-of-the-art algorithms have a hard time working. In Figure 3 we use the deep Wiener deblurring network [1] to deblur the image. When the illumination is strong, the method performs well. But when the illumination is weak, the algorithm performs poorly. We remark that this observation is common for many mainstream deblurring algorithms.

The present problem is best described as *photon limited* non-blind deblurring. It is a common problem for a variety of applications such as microscopy [7] and astronomy [8]. One should note that photon-limited imaging is a problem even if we use a perfect sensor with zero read noise and 100% quantum efficiency. The photon shot noise still exists due to the stochasticity of the photon arrival process [9]. Therefore, the solution presented in this paper is pan-sensor, meaning that it can be applied to the standard CCD and CMOS image sensors and the more advanced quanta image sensors (QIS) [10]–[12].

A. Problem formulation

Consider a monochromatic image $\boldsymbol{x} \in \mathbb{R}^N$ normalized to $[0, 1]$. We write the blurred image as $\boldsymbol{H}\boldsymbol{x}$ where $\boldsymbol{H} \in R^{N\times N}$ represents the blur kernel $\boldsymbol{h}$ in the matrix form. In photon-limited conditions, the observed image is given by

$$\boldsymbol{y} = \text{Poisson}(\alpha \cdot \boldsymbol{H}\boldsymbol{x}), \tag{2}$$

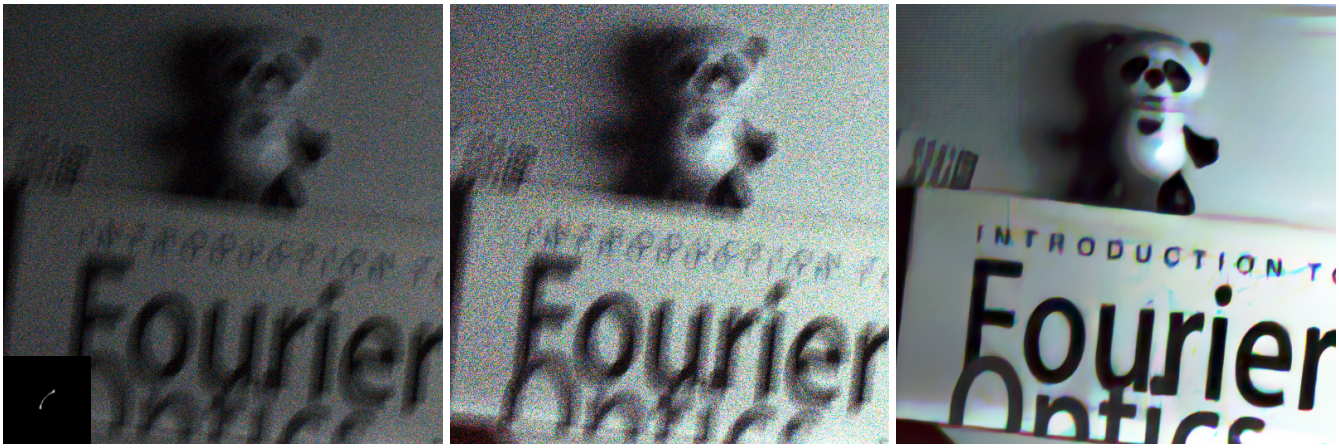
where $\text{Poisson}(\cdot)$ denotes the Poisson process, and α is a scalar to be discussed. The likelihood of the observed image $\boldsymbol{y}$ follows the Poisson probability distribution:

$$p(\boldsymbol{y}|\boldsymbol{x};\alpha) = \prod_{j=1}^{N} \frac{[\alpha \boldsymbol{H}\boldsymbol{x}]_j^{\boldsymbol{y}_j} e^{-[\alpha \boldsymbol{H}\boldsymbol{x}]_j}}{\boldsymbol{y}_j!}, \tag{3}$$

where $[\cdot]_j$ denotes the jth element of a vector. The scalar α represents the photon level. It is a function of the sensor's properties (e.g. quantum efficiency), camera settings (exposure time, aperture), and illumination level of the scene. For a given illumination, the photon level α can be increased by increasing the exposure time or the aperture. To give readers a better idea of the photon level α, we give a rough estimate of the photon flux (measured in terms of lux level) in Table I under a few typical imaging scenarios.[1]

[*]Y. Sanghvi, A. Gnanasambandam and S. Chan are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA. Email: {ysanghvi, agnanasa, stanchan}@purdue.edu.

[1]To estimate the photon level α from the photon flux level, we set the scene illumination to 1 lux (measured using a light meter) and measure the corresponding photons-per-pixel from the image sensor data captured using a Canon EOS Rebel T6i.



(a) Raw camera image (b) Histogram equalized image (c) Our reconstruction

Fig. 1. **Overview.** The goal of this paper is to present a new algorithm that reconstructs images from blur at a photon-limited condition.

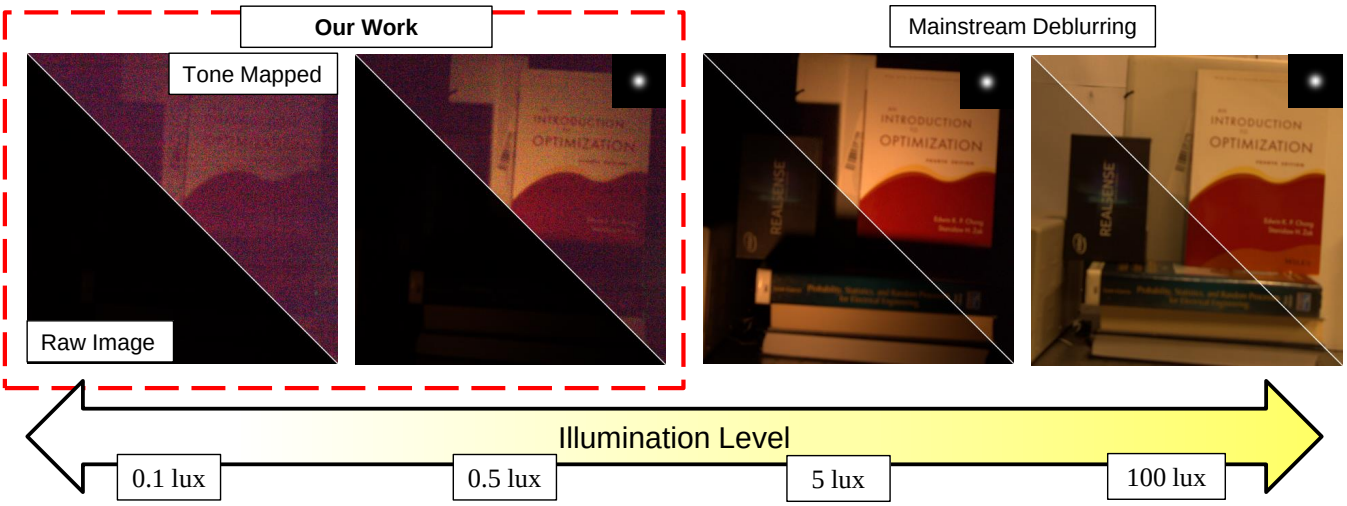


Fig. 2. **Comparison of photon-limited scenes (Left) with relatively well illuminated scenes (Right).** Raw images and their tone mapped versions taken in different illuminations and blurred by defocus are shown in the figure. As illumination of the scene decreases, the photon shot noise becomes more dominant, making the deblurring problem substantially more difficult - as shown in Figure 3. In this paper, we address the the problem of non-blind deblurring in a *photon-limited* setting i.e. when the number of photons captured by the sensor is low leading to corruption of images by the photon shot noise.

TABLE I
LIGHTING CONDITION AND ILLUMINATION LEVEL

Lighting condition	Illumination (lux)
Sunset	400
Dimly-lit Street	20-50
Moonlight	1
$\alpha = 5$ (This paper)	1

B. Contributions and scope

Photon-limited non-blind deblurring is a special case of the Poisson linear inverse problem. We limit the scope to deblurring so that we can demonstrate the algorithm using real low-light data.

Existing photon-limited deblurring methods are mostly deterministic [13]–[15]. To overcome the limitation of these methods, in this paper we present a deep-learning solution. We make two contributions:

1) We propose an unrolled plug-and-play (PnP [16], [17]) algorithm for solving the non-blind deblurring problem in *photon-limited* conditions. Unlike existing work such as [18] which uses an inner optimization to solve the Poisson proximal map, we use a three-operator splitting technique to turn all the sub-routines differentiable. This allows us to train the unrolled network end-to-end (which is previously not possible), and hence makes us the first unrolled network for Poisson deblurring.
2) We overcome the difficulty of collecting *real* photon-limited motion blur kernels and images for algorithm evaluation. A dataset containing 30 low-light images and the corresponding blur kernels are produced. We make this dataset publicly available.

II. Related Work

A. Poisson deconvolution

Poisson deconvolution has been studied for decades because of its important applications [19]. One of the earliest and the most cited works is perhaps the Richardson-Lucy (RL) algorithm [14], [15]. The method assumes a known blur kernel and derives an iterative scheme which converges to

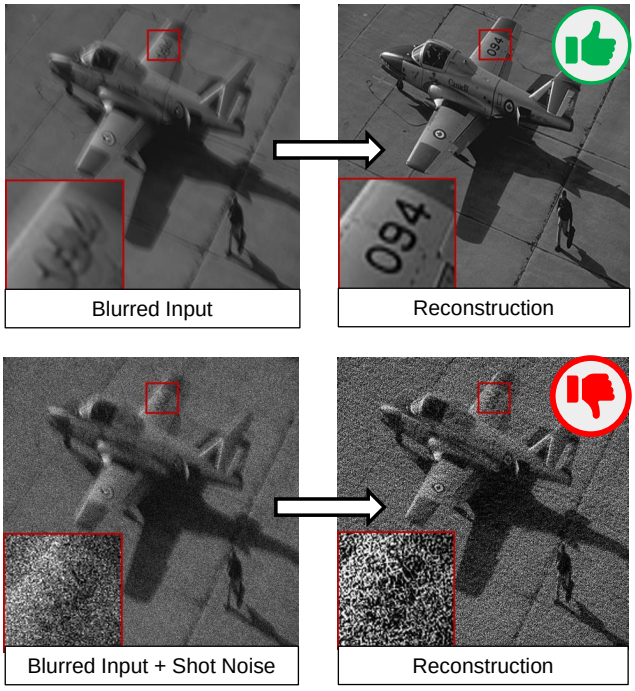


Fig. 3. **Limitation of existing image deblurring algorithms when applied to low-light images.** In this example we use the pre-trained neural network [1] to recover a well-illuminated scene and a poorly-illuminated scene. The method completely fails because of the noise, eve though the deblurring in a well-illuminated scene is satisfactory.

the maximum-likelihood estimate (MLE) of the deconvolution problem. The RL algorithm was applied to problems such as emission tomography [20] and confocal microscopy [21], [22]. However, since the prior is not used, the quality of reconstruction is limited.

Another class of iterative methods is based on maximum-a-posteriori (MAP) estimation by using a signal prior. For example, PIDAL-TV [23] solves a MAP cost function with the total-variation (TV) regularization using an augmented Lagrangian framework. Similarly, the sparse Poisson intensity reconstruction algorithm (SPIRAL) [24] looks for sparse solutions in an orthonormal basis, whereas [25] solves a MAP cost function with multiscale prior using the expectation-maximization algorithm.

Shrinkage based approaches such as PURE-LET [13] assume the deconvolution output to be a linear combination of elementary functions and minimize the expected mean squared error under a joint Poisson-Gaussian noise model. This boils down to solving a linear system of equations and has been also used to solve denoising, deblurring processes under Gaussian noise assumptions [26], [27].

Denoising under Poisson noise conditions can be viewed as a special case of the deblurring problem. One of the widely used techniques for Poisson denoising is the variance stabilizing transforms (VST) which applies the Anscombe transform [28] to stabilize the spatially varying noise variance. A standard denoising method is then used, followed by the inverse Anscombe transform. In [29], it was shown that an optimal inverse transform can outperform other standard Poisson denoising methods such as [30], [31]. The method in [32] provides an iterative version of the denoising via VST scheme by treating last iteration's denoised image as scaled Poisson data.

B. Plug-and-play

The Plug-and-play (PnP) framework was first introduced in [16] as a general purpose method to solve inverse problems by leveraging an off-the-shelf denoiser. Since then, the framework has been applied to different problems like bright field electron tomography [33] and magnetic resonance imaging (MRI) [34]. Using the same principle but with the half-quadratic splitting scheme, [35] demonstrated the use of a single denoiser for different image restoration tasks such as super-resolution, deblurring, and inpainting. Variations of PnP have also been used for Poisson deblurring [18], [36] and non-linear inverse problems [37]. A stochastic version of the scheme (PnP stochastic proximal gradient method) has been proposed for inverse problems with prohibitively large datasets [38]. Using the consensus equilibrium (CE) framework [39], the scheme can be extended to fuse multiple signal and sensor models.

The convergence of the Plug-and-Play scheme has been studied in detail. For example, [17] provided a variation of the scheme which can was provably convergent under the assumptions of a bounded denoiser and its performance was analysed under assumptions of a graph filter denoiser in [40]. [41] showed that if a denoiser satisfies certain Lipshitz conditions, the corresponding Plug-and-Play scheme can be shown to converge. Furthermore, the authors proposed real-spectral normalization as a way to impose the conditions on deep-learning based denoisers.

A closely related method which provides a framework to solve inverse problems using denoisers is REgularization by Denoising (RED) [42], [43]. The framework poses the cost function for an inverse problem as sum of a data term and image-adaptive Laplacian regularization term. This allows the resulting iterative process to be written as a series of denoising steps. In [44], it was mentioned that for RED to be valid the denoiser needs to have a symmetric Hessian.

C. Algorithm unrolling

The difficulty of running PnP and RED is that theyt need to iteratively use a deep network denoiser. An alternative way to implement the algorithm was proposed by Gregor and LeCun in 2010 [45] to unroll an iterative algorithm and train it in a supervised manner. For example, one can unroll the iterative shrinkage threshold algorithm (ISTA) for the purpose of approximating sparse codes of an image. The idea of unrolled networks has been employed in various image restoration tasks such as super-resolution [46], deblurring [47] and compressive sensing [48]. For a more extensive review of the algorithm unrolling, we refer the reader to [49]. More recently, there are new attempts to relax the fixed iteration structure of unrolling by analyzing the equilibrium of the underlying operators [50] .

Unrolling iterative algorithms provide multiple advantages compared to generic deep learning architectures. The unrolled

networks provide greater interpretability and are often parameter efficient compared to their counterparts such as the U-Net [51]. Since the networks are unrolled version of iterative algorithms, they are less susceptible to problem of overfitting.

III. Method

A. Algorithm unrolling

The proposed solution for the Poisson deblurring problem is to unroll the iterative PnP algorithm. The idea is that we start by deriving the PnP iterative update steps. In the "unrolled" version of the iterative algorithm, each iteration is treated as a computing block. Each computing block has its own set of trainable parameters. The blocks are concatenated in series with each other. The output at the end of the last block is used as the target for a supervised loss to fine-tune the trainable parameters.

Before describing the iterative algorithm we aim to unroll, we briefly describe the underlying cost function. Most inverse problem algorithm aim to determine the MAP estimate of the underlying signal $\boldsymbol{x}$ by maximizing the log-posterior

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}} \Big[\log p(\boldsymbol{y}|\boldsymbol{x}) + \log p(\boldsymbol{x}) \Big], \tag{4}$$

where $p(\boldsymbol{x})$ denotes the natural image prior. Plugging (3) in (4) and taking the negative of the cost function, the maximization becomes

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \Big[\alpha \mathbb{1}^T \boldsymbol{H}\boldsymbol{x} - \boldsymbol{y}^T \log(\alpha \boldsymbol{H}\boldsymbol{x}) - \log p(\boldsymbol{x}) \Big], \tag{5}$$

where $\mathbb{1}$ represents the all-one vector. Note that the factorial term $\log \boldsymbol{y}!$ has been dropped since it is independent of $\boldsymbol{x}$. The prior $p(\boldsymbol{x})$ has not been explicitly specified yet and this issue will be addressed through the use of a denoiser in the next subsection.

B. Conventional PnP for Poisson inverse problems

Now we describe how the Plug-and-Play method can be applied to the Poisson deblurring problem. We start with the alternate direction of method of multipliers (ADMM) [52] formulation – where we convert the unconstrained optimization problem to a constrained optimization problem by performing the variable splitting $\boldsymbol{x} = \boldsymbol{z}$

$$\begin{aligned} \{\boldsymbol{x}^*, \boldsymbol{z}^*\} = \underset{\boldsymbol{x},\boldsymbol{z}}{\operatorname{argmin}} \Big[- \log p(\boldsymbol{y}|\boldsymbol{x}) - \log p(\boldsymbol{z}) \Big], \\ \text{subject to } \boldsymbol{x} = \boldsymbol{z}, \end{aligned} \tag{6}$$

At the minimum of the above optimization problem, the constraint $\boldsymbol{x}^* = \boldsymbol{z}^*$ must be satisfied and hence the constrained optimization solution is equivalent to the unconstrained solution in (5).

The augmented Lagrangian associated with the constrained problem in (6) is

$$\begin{aligned} \{\boldsymbol{x}^*, \boldsymbol{z}^*, \boldsymbol{u}^*\} = \underset{\boldsymbol{x},\boldsymbol{z}}{\operatorname{argmin}} \Big[\alpha \mathbb{1}^T \boldsymbol{H}\boldsymbol{x} - \boldsymbol{y}^T \log(\alpha \boldsymbol{H}\boldsymbol{x}) \\ - \log p(\boldsymbol{z}) + \frac{\rho}{2} \|\boldsymbol{x} - \boldsymbol{z} + \boldsymbol{u}\|^2 - \frac{\rho}{2} \|\boldsymbol{u}\|^2 \Big], \end{aligned} \tag{7}$$

where $\boldsymbol{u}$ denotes the scaled Lagrange multiplier corresponding to the constraint $\boldsymbol{x} = \boldsymbol{z}$, and ρ denotes the penalty parameter. The corresponding iterative updates are:

$$\boldsymbol{x}^{k+1} = \underbrace{\underset{\boldsymbol{x}}{\operatorname{argmin}} \Big[\alpha \mathbb{1}^T \boldsymbol{H}\boldsymbol{x} - \boldsymbol{y}^T \log(\alpha \boldsymbol{H}\boldsymbol{x}) + \frac{\rho}{2} \|\boldsymbol{x} - \widetilde{\boldsymbol{x}}^k\|^2 \Big]}_{\text{Proximal operator for the negative log-likelihood}}, \tag{8a}$$

$$\boldsymbol{z}^{k+1} = \underbrace{\underset{\boldsymbol{z}}{\operatorname{argmin}} \Big[- \log p(\boldsymbol{z}) + \frac{\rho}{2} \|\boldsymbol{z} - \widetilde{\boldsymbol{z}}^k\|^2 \Big]}_{\text{Proximal operator for the negative-log-prior}}, \tag{8b}$$

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + (\boldsymbol{x}^{k+1} - \boldsymbol{z}^{k+1}), \tag{8c}$$

with $\widetilde{\boldsymbol{x}}^k \overset{\text{def}}{=} \boldsymbol{z}^k - \boldsymbol{u}^k$ and $\widetilde{\boldsymbol{z}}^k \overset{\text{def}}{=} \boldsymbol{x}^k + \boldsymbol{u}^k$. In the Plug-and-Play framework [16], [17], the $\boldsymbol{z}$ update in (8b) is implemented by an image denoiser.

The difficulty of solving the above problem is that the $\boldsymbol{x}$-update in (8a) does not have a closed form expression for the Poisson likelihood. Thus (8a) needs to be solved using an inner-loop optimization method such as L-BFGS [53]. Unrolling this inner-loop optimization solver can be inefficient as it may not be differentiable. Hence unrolling the PnP scheme for the Poisson inverse problem using the existing framework is infeasible. To be more specific, while the $\boldsymbol{z}$-update in (8b) can be implemented as a neural network and hence is differentiable, the same cannot be said for $\boldsymbol{x}$-update in (8a). As shown in Figure 4, when (8a) is solved using another iterative method such as L-BFGS (for e.g. in [18]), it is not differentiable. As a result, training the unrolled network via backpropagation is not possible unless (8a) can be made differentiable.

C. Three-operator splitting for Poisson PnP

As explained in the previous subsection, the current framework does not allow for algorithm unrolling. To circumvent this issue, we use an alternate three-operator formulation of the PnP-framework. Through this reformulation of Plug-and-Play, we derive a series of iterative updates where each step can be implemented as a single-step that is differentiable. The three-operator splitting strategy we use here has been used in context of Poisson deblurring in [36], [54] using a BM3D and TV denoiser respectively.

In this scheme, instead of a two-operator splitting strategy for conventional PnP in (6), we use three-operator splitting to form the corresponding constrained optimization problem. Specifically, in addition to splitting the variable as $\boldsymbol{x} = \boldsymbol{z}$, we introduce a third variable $\boldsymbol{v}$ corresponding to blurred image $\boldsymbol{H}\boldsymbol{x}$ and hence the constraint $\boldsymbol{H}\boldsymbol{x} = \boldsymbol{v}$.

$$\begin{aligned} \{\boldsymbol{x}^*, \boldsymbol{z}^*, \boldsymbol{v}^*\} = \underset{\boldsymbol{x},\boldsymbol{z},\boldsymbol{v}}{\operatorname{argmin}} \Big[- \boldsymbol{y}^T \log(\alpha \boldsymbol{v}) + \alpha \mathbb{1}^T \boldsymbol{v} + \log p(\boldsymbol{z}) \Big], \\ \text{subject to } \boldsymbol{x} = \boldsymbol{z}, \quad \text{and} \quad \boldsymbol{H}\boldsymbol{x} = \boldsymbol{v}. \end{aligned} \tag{9}$$

After forming the corresponding augmented Lagrangian, we arrive at the following iterative updates:

$$\boldsymbol{x}^{k+1} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \Big[\frac{\rho_1}{2} \|\boldsymbol{x} - \widetilde{\boldsymbol{x}}_0^k\|^2 + \frac{\rho_2}{2} \|\boldsymbol{H}\boldsymbol{x} - \widetilde{\boldsymbol{x}}_1^k\|^2 \Big], \tag{10a}$$

$$\boldsymbol{z}^{k+1} = \underset{\boldsymbol{z}}{\operatorname{argmin}} \Big[- \log p(\boldsymbol{z}) + \frac{\rho_1}{2} \|\boldsymbol{z} - \widetilde{\boldsymbol{z}}_1^k\|^2 \Big], \tag{10b}$$

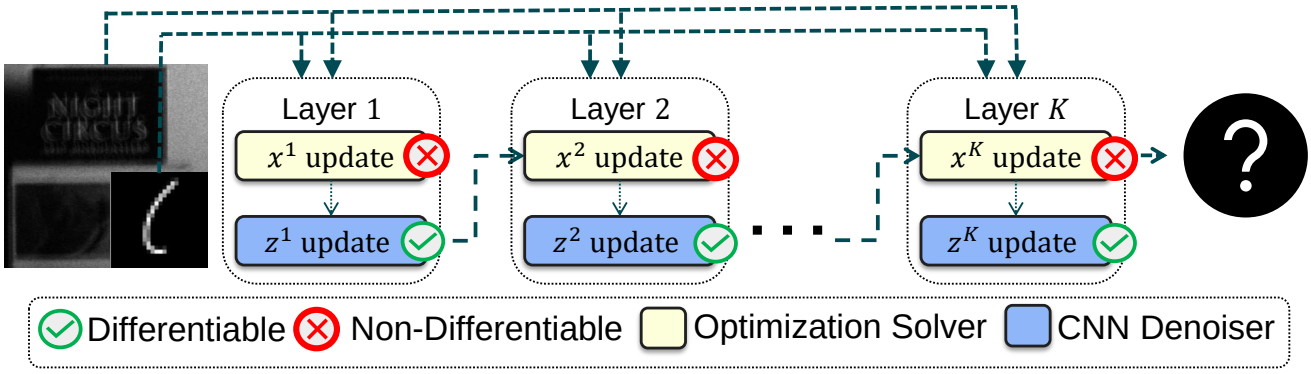


Fig. 4. **Conventional two-operator splitting Plug-and-Play.** Conventional Plug-and-Play applied to the Poisson deblurring problem using equations (8a) and (8b). While (8b) is implemented as an image denoiser and hence differentiable, $\boldsymbol{x}$-update i.e. (8a) is implemented as a convex optimization solver and hence not differentiable. This makes the conventional PnP infeasible for fixed iteration unrolling and hence end-to-end training.

$$\boldsymbol{v}^{k+1} = \underset{\boldsymbol{v}}{\text{argmin}} \Big[-\boldsymbol{y}^T \log(\alpha \boldsymbol{v}) + \alpha \mathbb{1}^T \boldsymbol{v} + \frac{\rho_2}{2} \|\boldsymbol{v} - \widetilde{\boldsymbol{v}}^k\|^2 \Big], \tag{10c}$$

$$\boldsymbol{u}_1^{k+1} = \boldsymbol{u}_1^k + \boldsymbol{x}^{k+1} - \boldsymbol{z}^{k+1}, \tag{10d}$$

$$\boldsymbol{u}_2^{k+1} = \boldsymbol{u}_2^k + \boldsymbol{H}\boldsymbol{x}^{k+1} - \boldsymbol{v}^{k+1}, \tag{10e}$$

where $\widetilde{\boldsymbol{x}}_0^k \overset{\text{def}}{=} \boldsymbol{z}^{k+1} - \boldsymbol{u}_1^k$, $\widetilde{\boldsymbol{x}}_1^k \overset{\text{def}}{=} \boldsymbol{v}^{k+1} - \boldsymbol{u}_2^k$, $\boldsymbol{v}^k \overset{\text{def}}{=} \boldsymbol{H}\boldsymbol{x}^k + \boldsymbol{u}_2^k$, and $\widetilde{\boldsymbol{z}}^k \overset{\text{def}}{=} \boldsymbol{x}^k + \boldsymbol{u}_1^k$. Similar to the PnP formulation described in last subsection, the vectors $\boldsymbol{u}_1, \boldsymbol{u}_2$ denote the scaled Lagrangian multipliers for the constraints $\boldsymbol{x} - \boldsymbol{z} = 0$ and $\boldsymbol{H}\boldsymbol{x} - \boldsymbol{v} = 0$ respectively. The scalars ρ_1, ρ_2 denote the corresponding penalty parameters.

Each of the subproblems defined in (10a, 10b, 10c) have a closed form solution and are described below:

$\boldsymbol{x}$-subproblem: (10a) is a least squares minimization problem, whose solution can be explicitly given as follows:

$$\boldsymbol{x}^{k+1} = (\boldsymbol{I} + (\rho_2/\rho_1)\boldsymbol{H}^T\boldsymbol{H})^{-1}(\widetilde{\boldsymbol{x}}_0^k + (\rho_2/\rho_1)\boldsymbol{H}^T\widetilde{\boldsymbol{x}}_1^k). \tag{11}$$

Since $\boldsymbol{H}$ represents a convolutional operator, the operation can be performed without any matrix inversions using Fourier Transforms.

$$\boldsymbol{x}^{k+1} = \mathcal{F}^{-1}\Big[\frac{\mathcal{F}(\widetilde{\boldsymbol{x}}_0^k) + (\rho_2/\rho_1)\overline{\mathcal{F}(\boldsymbol{h})}\mathcal{F}(\widetilde{\boldsymbol{x}}_1^k)}{1 + (\rho_2/\rho_1)|\mathcal{F}(\boldsymbol{h})|^2}\Big], \tag{12}$$

where $\mathcal{F}(\cdot)$ represents the discrete Fourier transform of the image or blur kernel implemented using the Fast Fourier Transform after appropriate boundary padding. We refer to it as the *deblurring operator*.

$\boldsymbol{z}$-subproblem: (10b) is a proximal operator for the negative log prior term. Using the insight provided in Plug-and-Play scheme, (10b) can be viewed as a denoising operation

$$\boldsymbol{z}^{k+1} = D(\widetilde{\boldsymbol{z}}^k), \tag{13}$$

where $D(\cdot)$ is any image denoiser. For end-to-end training, we require $D(\cdot)$ to be differentiable and trainable – a property satisfied by all convolutional neural network denoisers.

$\boldsymbol{v}$-subproblem: (10c) is a convex optimization problem but can be solved without an iterative procedure. Separating out each component of the vector minimization and setting the gradient equal to zero gives the following equation

$$-\frac{[\boldsymbol{y}]_i}{[\boldsymbol{v}^{k+1}]_i} + \alpha + \rho_2([\boldsymbol{v}^{k+1}]_i - [\widetilde{\boldsymbol{v}}^k]_i) = 0, \tag{14}$$

for $i = 1, 2, \cdots, N$. Solving the resulting quadratic equation and ignoring the negative solution gives the following update step

$$\boldsymbol{v}^{k+1} = \frac{(\rho_2\widetilde{\boldsymbol{v}}^k - \alpha) + \sqrt{(\rho_2\widetilde{\boldsymbol{v}}^k - \alpha)^2 + 4\rho_2\boldsymbol{y}}}{2\rho_2}, \tag{15}$$

Since the optimization problem in (10c) is a sum of the the negative log-likelihood for Poisson noise and a quadratic penalty term, we refer to this update as *Poisson proximal operator*.

Algorithm 1 Three-Operator Splitting for Poisson PnP

Input: Blurred and Noisy Image $\boldsymbol{y}$, kernel $\boldsymbol{h}$, Photon level α
Initialize $\boldsymbol{x}^0$ using (16)
$\boldsymbol{z}^0 \leftarrow \boldsymbol{x}^0$, $\boldsymbol{v}^0 \leftarrow \boldsymbol{y}$ $\boldsymbol{u}_1^0 \leftarrow 0$, $\boldsymbol{u}_2^0 \leftarrow 0$
for $k = 1, 2, \cdots, K$ **do**
    Update $\boldsymbol{x}^k$ using Eq. (12)
    Update $\boldsymbol{z}^k$ using Eq. (13)
    Update $\boldsymbol{v}^k$ using Eq. (15)
    $\boldsymbol{u}_1^k \leftarrow \boldsymbol{u}_1^{k-1} + \boldsymbol{x}^k - \boldsymbol{z}^k$
    $\boldsymbol{u}_2^k \leftarrow \boldsymbol{u}_2^{k-1} + \boldsymbol{H}\boldsymbol{x}^k - \boldsymbol{v}^k$
end for
return $\boldsymbol{x}^K$

The convergence of Algorithm 1 has been derived in [23]. It was shown that as long as $\boldsymbol{G} = [\boldsymbol{H}^T, \boldsymbol{I}]^T$ has a full column rank, the three-operator splitting scheme converges. Furthermore, assuming the denoiser D is continuously differentiable and $\nabla D(\cdot)$ is symmetric with eigenvalues in $[0, 1]$, convergence results in [33] show that the corresponding negative-log prior, i.e., $-\log(p(\cdot))$ is closed, proper and convex. Combined with the result from [23], it can be shown that the three-operator PnP scheme in Algorithm 1 converges.

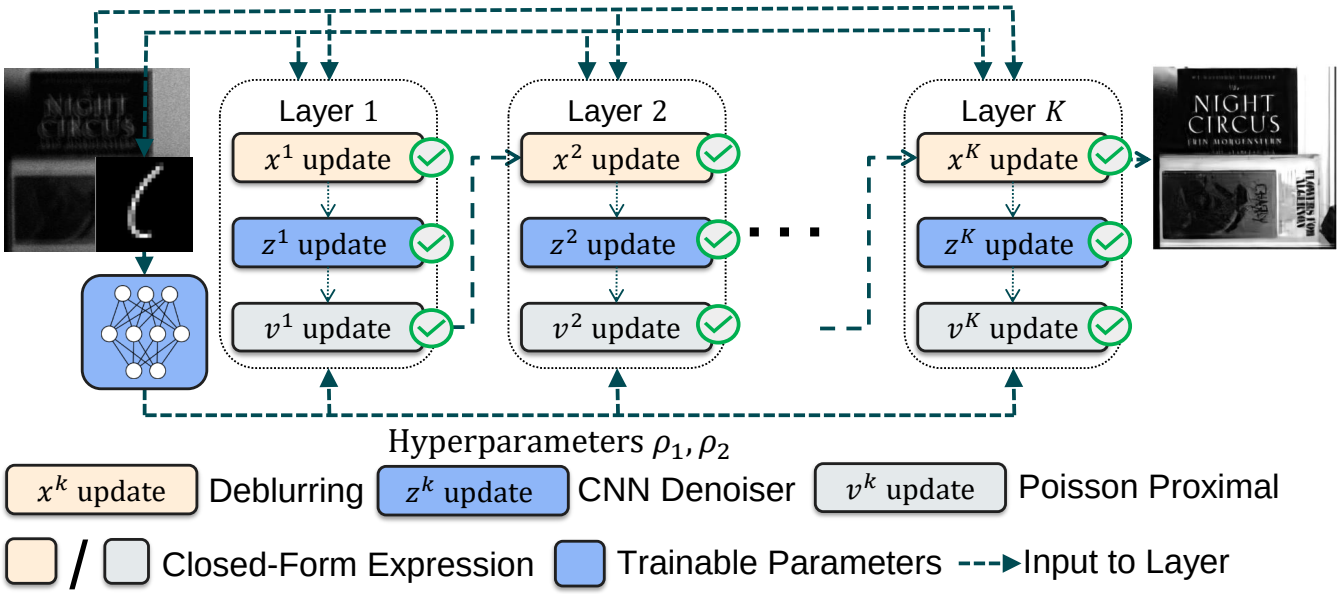


Fig. 5. **Proposed unrolled Plug-and-Play for deblurring.** For conventional PnP, the data sub-problem cannot be solved in a single step and instead requires convex optimization solvers. This stops us from unrolling the iterative procedure and training it end-to-end via back-propagation. Through the three-operator splitting formulation of the problem, each sub-module in an iteration is in closed form and more importantly, differentiable. This allows for end-to-end training which was not possible in conventional PnP.

D. Unfolding the three-operator splitting

With an end-to-end trainable iterative process, we can now describe the unfolded iterative network. The Plug-and-Play updates described in Algorithm 1 are now unfolded for $K = 8$ iterations and the entire differentiable pipeline is trained in a supervised manner, as summarized in Figure 5.

Initialization: To initialize the variable $\boldsymbol{x}^0$, we use the Wiener filtering step (not to be confused with [1]) :

$$\boldsymbol{x}^0 = \frac{1}{\alpha}\mathcal{F}^{-1}\left\{\frac{\overline{\mathcal{F}(\boldsymbol{h})}\mathcal{F}(\boldsymbol{y})}{1/\alpha + |\mathcal{F}(\boldsymbol{h})|^2}\right\}, \tag{16}$$

where the constant factor $1/\alpha$ in the denominator represents the inverse of the signal-to-noise ratio of the blurred measurements. Note that this step can be derived as an ℓ_2 regularized solution of the deconvolution problem as well.

Hyperparameters: The parameters used in updates (10a), (10c) – ρ_1, ρ_2 are changed for each iteration and determined in one-shot by the blurring kernel $\boldsymbol{h}$ and photon level α as they control the degradation of the image. The kernel $\boldsymbol{h}$ is used as input to 4 convolutional layers, flattened to a vector of length 1024. Along the with the photon level α, the flattened vector is used as an input to a 3-layer fully connected network which output two set of vectors i.e. $\{\rho_1^1, \rho_1^2, ..., \rho_1^K\}$ and $\{\rho_2^1, \rho_2^2, ..., \rho_2^K\}$. We refer the readers to the supplementary document for further architectural details.

Denoiser: For the denoiser used in (13), we use the architecutre provided in [46] which introduces skip connections in a U-Net architecture known as ResUNet. Like a standard U-Net, there are four downsampling operations followed by 4 upsampling operations with skip connections between the upsampling and downsampling operators. For further details of the architecture we refer the readers to [46] or the supplmentary document. Note that in our implementation of the architecture, we do not concatenate the denoiser input $\tilde{\boldsymbol{z}}^k$ with a noise level.

IV. Experiments

A. Training

We train the network described in section III using the multi-scale ℓ_1-loss function. It is defined as

$$\mathcal{L}(\hat{\boldsymbol{x}}, \boldsymbol{x}) = \sum_{l=1}^{3} ||\hat{\boldsymbol{x}}^l - \boldsymbol{x}^l||_1, \tag{17}$$

where $\hat{\boldsymbol{x}}^l$, $\boldsymbol{x}^l$ represent the reconstructed output and ground truth downsampled by a factor of $2^{(l-1)}$.

We use images from the Flickr2K [55] dataset to train the network. The dataset contains a total of 2650 images of which we partition using a $80/20$ split for training and validation. All images are converted to gray-scale, scaled to a size of 256×256, and are blurred using motion kernels generated from [56] and Gaussian blur kernels. Due to memory limits of GPU, random patches of size 128×128 were cropped and used as inputs for the network during training.

For training, a combination of 60 motion kernels generated from [56] and 10 isotropic gaussian blur kernels with σ varying from $[0.1, 2.5]$ were used. All the kernels were pre-generated prior to training and were randomly selected during training. Entries of the blur kernel are non-negative and sum to 1. Photon Shot noise is synthetically added to the blurred image according to (2). The photon level α is uniformly sampled from the range $[1, 60]$.

The inputs to the network consist of the blurred and corrupted image $\boldsymbol{y}$, the normalized blur kernel $\boldsymbol{h}$, and the photon noise level α. The output from the network is the reconstructed image $\boldsymbol{x}^K$ where K denotes the number of iterations for which

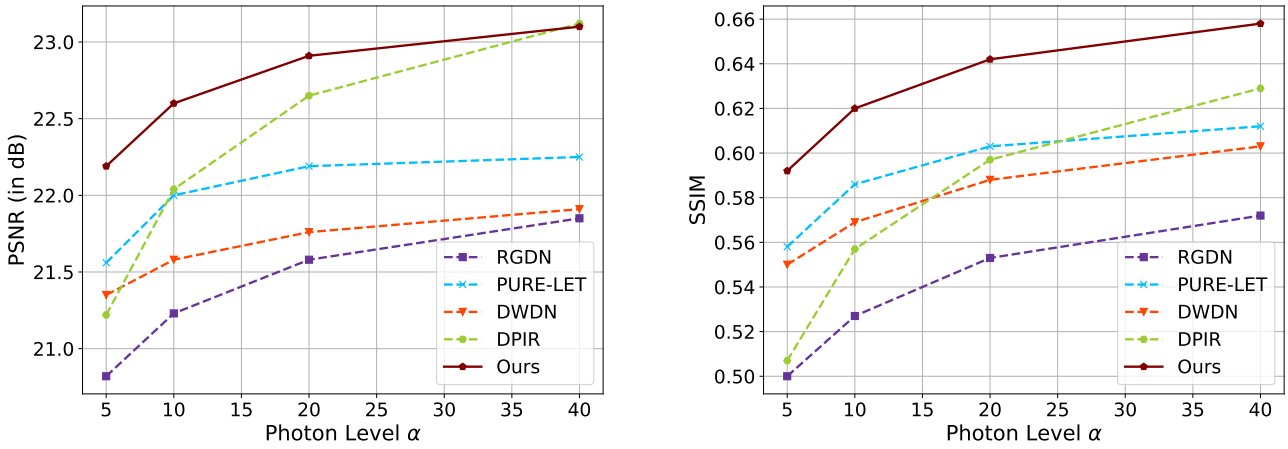


Fig. 6. **Quantitative evaluation.** Comparison of PSNR and SSIM of the different methods on Levin et. al. dataset [57]. The dataset consists of 32 blurred images generated by blurring 4 images by 8 motion kernels. The images were corrupted by Poisson noise at photon levels $\alpha = 5, 10, 20$ and 40.

the scheme is unrolled for. We set the the number of iterations in our implementation to $K = 8$. Using the loss function in (17), we train the network with Adam optimizer [58] using a learning rate 1×10^{-4} and batch size of 5 for 100 epochs. All the parameters of the network are initialized using Xavier initialization [59] and is implemented in Pytorch 1.7.0. For training, we use an NVIDIA Titan Xp GP102 GPU and it takes approximately 20 hours for training to complete.

B. Quantitative Evaluation

The results are summarized in Figure 6. We evaluate our method using synthetically generated noisy blurred images on 100 images from the BSDS300 dataset [60], from now on referred to as *BSD100*. We evaluate the performance on different photon levels ($\alpha = 5, 10, 20, 40$) representing various levels of degradation in terms of signal-to-noise ratio. We test the methods for different blur kernels - specifically 4 isotropic Gaussian kernels, 4 anisotropic Guassian kernels, and 4 motion kernels, as illustrated in Figure 7. Note that the top-left kernel's width is very small - this can be viewed as an identity operator and hence equivalent to evaluating the method's performance on denoising (as opposed to deblurring).

TABLE II
DIFFERENT FEATURES OF METHODS USED IN THIS PAPER FOR POISSON DEBLURRING. WE CLASSIFY THE METHODS BASED ON THREE CRITERIA - ITERATIVE/NON-ITERATIVE, END-TO-END TRAINABILITY AND WHETHER THE MODEL EXPLICITLY INCORPORATES THE FACT THAT THE IMAGES ARE CORRUPTED BY POISSON SHOT NOISE.

Method	**Iterative?**	**End-to-End Trainable?**	**Handles Poisson Noise?**
RGDN [2]	✓	✓	×
PURE-LET [13]	×	×	✓
DWDN [1]	×	✓	×
DPIR [35]	✓	×	×
Ours	✓	✓	✓

We compare our method with the following deblurring methods - **RGDN** [2], **PURE-LET** [13], **DWDN** [1], and **DPIR** [35]. RGDN (Recurring Gradient Descent Network) poses deblurring as a MAP estimation problem with a least squares data term. The corresponding optimization problem is iteratively unrolled and then trained end-to-end. PURE-LET (Poisson Unbiased Risk Estimate - Linear Expansion of Thresholds) proposes the solutions as a linear combination of basis function whose weights are determined by minimizing the unbiased estimate of the mean squared loss under given noise conditions. DWDN (Deep Wiener Deconvolution Network) performs deconvolution of the image in a learned feature space using Wiener filtering and predicts the image from the deconvolved features. DPIR (Deep Plug-and-Play Image Restoration) uses a pre-trained denoiser in a half-quadratic splitting scheme.

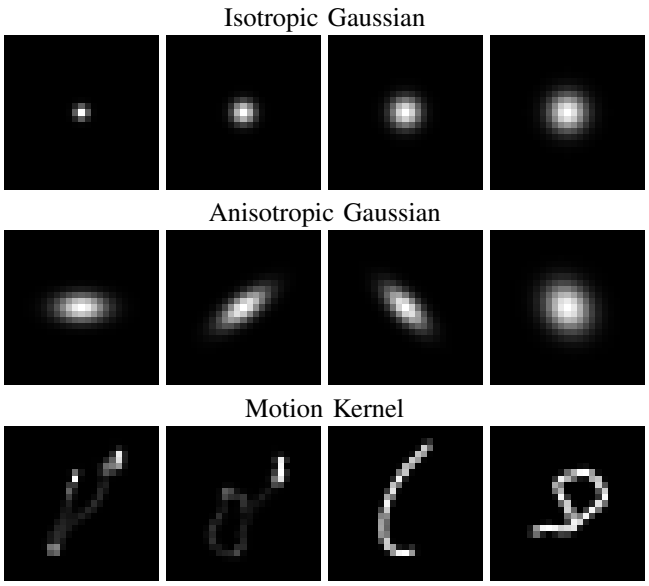


Fig. 7. **Kernels used for evaluation on BSD100 dataset.**

We compare our approach with the above methods because they are relatively recent and have demonstrated an ability to

deblur under heavy noise. For the sake of a fair comparison, the end-to-end trainable methods RGDN and DWDN were retrained using the same procedure as that of our method.

In addition to the BSD100 dataset, we also evaluated these methods on the blurring dataset provided in Levin et. al [57]. This dataset contains a set of 32 blurred images generated by blurring 4 different clean images by 8 different motion kernels. We synthetically corrupt the blurred images with Poisson noise at different illumination levels.

The results for these evaluations are provided in Table (III) and Figure 6. For qualitative comparison on grayscale and colour reconstructions, one can refer to Figure 8. On the BSDS100 dataset, our method outperforms the competing methods on all blurring kernels and illumination levels. For the dataset by Levin et. al, we outperform the other methods except DPIR at photon level $\alpha = 40$. On both datasets, we observe that the gap between conventional deblurring and our method decreases as the illumination levels increase. This is because as the mean of a Poisson random variable starts increasing, the probability distribution function resembles that of a Gaussian. Therefore, the conventional deblurring methods which are designed for Guassian noise show improved performance.

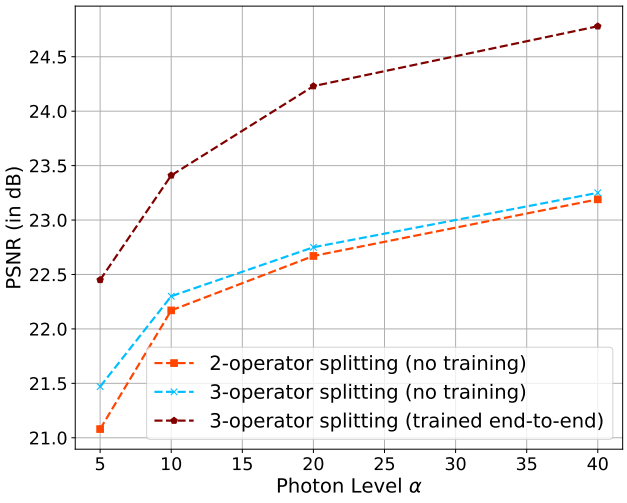


Fig. 9. **Ablation study to quantify significance of algorithm unrolling.** We evaluate the following three schemes on the BSD100 dataset **(a)** conventional PnP (two-operator splitting) with a DnCNN denoiser. **(b)** alternate PnP (three-operator splitting) with a DnCNN denoiser. **(c)** proposed fixed iteration unrolled network using a DnCNN denoiser. The results of this experiment show that the significant improvement is achieved due to the network unrolling.

C. Comparison with conventional Plug-and-Play

As explained in Section III-B, conventional Plug-and-Play using two-operator splitting is not suitable for algorithm unrolling. The proposed three-operator splitting enables algorithm unrolling because every iterative step is differentiable. It is this end-to-end training that allows us to a better performance. In this experiment, we perform an ablation study to quantify the performance gain through different combinations of unrolling and training.

In Figure 9, we show the reconstruction performance of three schemes on the BSD100 dataset: (a) conventional two-operator splitting PnP using FFDNet denoiser as described in Section III-B (b) an alternate three-operator splitting formulation using FFDNet as described in Section III-C and (c) the proposed unrolled version of the scheme described in Section III-C. The results show that the two iterative schemes (a) and (b) perform similarly. However, training the proposed algorithm unrolling achieves a consistent performance gain of more than 1dB across all photon levels.

When implementing the conventional PnP in (a), we use the approach from [18] and solve the $\boldsymbol{x}$-update (8a) using a L-BFGS solver [53]. Like the original implementation, we use a surrogate cost function to approximate the near zero entries with a quadratic approximation to avoid the singularities in the original cost function. A pretrained DnCNN [61] for noise level $\sigma = 15/255$ was used for the $\boldsymbol{z}$-update (8b). For the three-operator splitting scheme in (b), the same denoiser was used. To ensure a fair comparison, in the proposed fixed iteration unrolled network, we replace the ResUNet denoiser with a DnCNN and train it using the method described in Section IV-A. Further details about the experiment are provided in the supplementary document.

D. Color reconstruction

The focus of this paper is image deblurring. We acknowledge that most image sensors today acquire color images using the color filter arrays. However, adding the deblurring task with demosaicking is substantially beyond the scope of this paper. Even for demosaicking without any blur, the shot noise requires customized design, e.g., [62]. Therefore, color images shown in this paper were processed individually for each color channel and then fused using an off-the-shelf demosaicking algorithm. While this approach is sub-optimal, our real image experiments show that the performance is acceptable.

V. Experiments using Sensor Data

Unlike conventional deblurring problems where datasets are widely available, photon-limited deblurring data is not easy to collect. In this section we report our efforts in collecting a new dataset for evaluating low-light deblurring algorithms.

A. Photon-limited deblurring dataset

We collect shot-noise corrupted and blurred images using a digital single lens reflex (DSLR) camera. The DLSR is hand-held to generate motion blur. A Dell 24-inch monitor, pointing towards the region of interest, was used as a programmable illumination source to control the photon level α. A light-meter is placed in the scene to measure the photon flux level.

Image Capture: We use an Canon EOS Rebel T6i camera to capture the images with exposure time of 30 ms and aperture $f/5.0$. The ISO was set to the highest possible value of 12800 to maximize the internal gain of the sensor and hence minimize the quantization effects of the analog-to-digital convertor (ADC). The same scene was captured using different illumination levels and correspondingly different motion blur

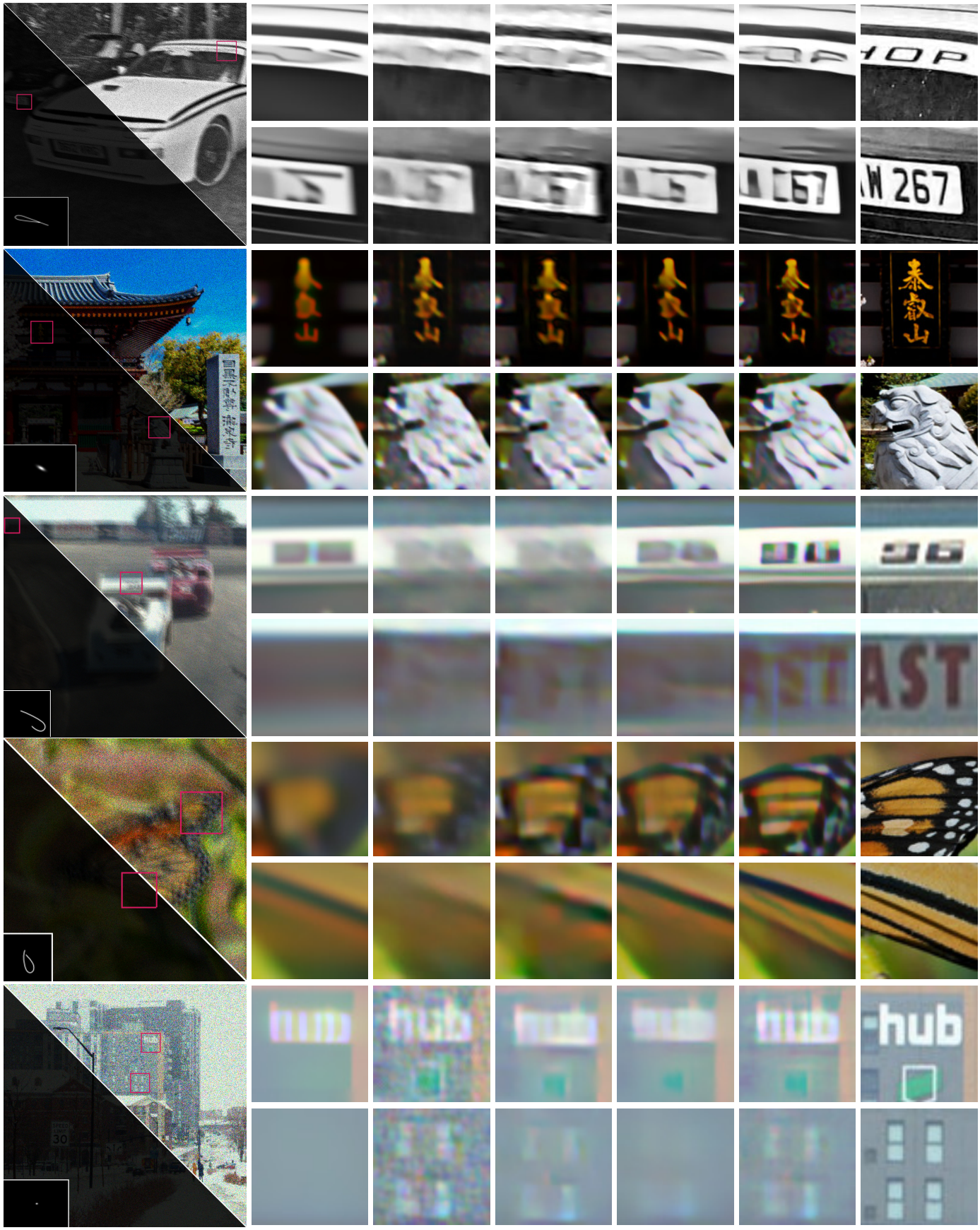


Fig. 8. **Qualitative Evaluation on synthetic images.** We compare the performance of the proposed method with competing methods on synthetic grayscale and color images.

TABLE III
COMPARISON OF PROPOSED METHOD WITH OTHER COMPETING APPROACHES ON BSD100 DATASET

Photon Level	Kernel		RGDN [2]	PURE-LET [13]	DWDN [1]	DPIR [35]	Ours
$\alpha = 5$	Isotropic Gaussian	PSNR (dB)	21.77	22.78	22.50	22.33	**23.46**
		SSIM	0.440	0.502	0.493	0.431	**0.531**
	Anisotropic Gaussian	PSNR (dB)	21.62	22.22	22.19	21.92	**22.70**
		SSIM	0.427	0.463	0.464	0.409	**0.491**
	Motion	PSNR (dB)	21.14	21.49	21.54	21.35	**22.12**
		SSIM	0.377	0.419	0.413	0.377	**0.433**
$\alpha = 10$	Isotropic Gaussian	PSNR (dB)	22.57	23.54	22.86	23.17	**24.24**
		SSIM	0.491	0.549	0.527	0.476	**0.576**
	Anisotropic Gaussian	PSNR (dB)	22.30	22.81	22.56	22.60	**23.28**
		SSIM	0.466	0.501	0.494	0.448	**0.525**
	Motion	PSNR (dB)	21.51	22.07	21.94	21.98	**22.80**
		SSIM	0.399	0.454	0.443	0.411	**0.475**
$\alpha = 20$	Isotropic Gaussian	PSNR (dB)	23.11	24.27	23.16	23.98	**24.96**
		SSIM	0.528	0.594	0.558	0.522	**0.621**
	Anisotropic Gaussian	PSNR (dB)	22.78	23.34	22.86	23.20	**23.83**
		SSIM	0.494	0.536	0.522	0.485	**0.557**
	Motion	PSNR (dB)	21.82	22.70	22.27	22.65	**23.47**
		SSIM	0.418	0.494	0.475	0.448	**0.515**
$\alpha = 40$	Isotropic Gaussian	PSNR (dB)	23.47	25.00	23.35	24.76	**25.68**
		SSIM	0.555	0.638	0.582	0.569	**0.663**
	Anisotropic Gaussian	PSNR (dB)	23.10	23.82	23.10	23.74	**24.36**
		SSIM	0.515	0.569	0.545	0.520	**0.589**
	Motion	PSNR (dB)	22.07	23.38	22.52	23.36	**24.20**
		SSIM	0.436	0.538	0.502	0.488	**0.564**

kernels. The raw image files were used for image processing instead of the compressed JPG files.

Generating Blur: To capture the blur kernel along with the image, we place a point source in each scene (see bottom right of middle image in Figure 10). The point source is created by placing an LED behind a black screen with a $30\mu m$ pinhole. The strength of the point source is maximized to ensure the kernel is not corrupted with shot noise without saturation of pixel values. Some example kernels collected through this process can be visualized in Figure 11.

Photon Level: The illumination of the scenes varies between 1-5 Lux, as measured by the light-meter shown in Figure 10. To maximize the amount of photons captured, the aperture is kept as large as possible. However shot noise is still present due to the relatively short exposure time. The estimated average photons-per-pixel (ppp) varied from 5-60.

Generating Ground Truths: For quantitative evaluation, we also provide the ground truth for each noisy blurred image. For each noisy image corrupted by motion and noise - we place the camera on a tripod and capture 10 frames of the scene under the same illumination and camera settings. The frames, captured without any blur due to camera shake, are averaged to reduce the shot noise as much as possible. These images serve as ground truth when evaluating the performance of reconstruction methods using PSNR/SSIM.

B. Reconstruction from real data

Pre-processing: To reconstruct the images using our network, we first need to convert it into the format representing the number of photons captured from the raw sensor values. The raw digital data ($\boldsymbol{y}_{\text{raw}}$) from the .RAW file is presented using a 14-bit value. To convert the 14-bit format to the number of photons, we use the following linear transform

$$\boldsymbol{y}_i = \frac{\boldsymbol{y}_{i\ \text{raw}} - b}{G}, \tag{18}$$

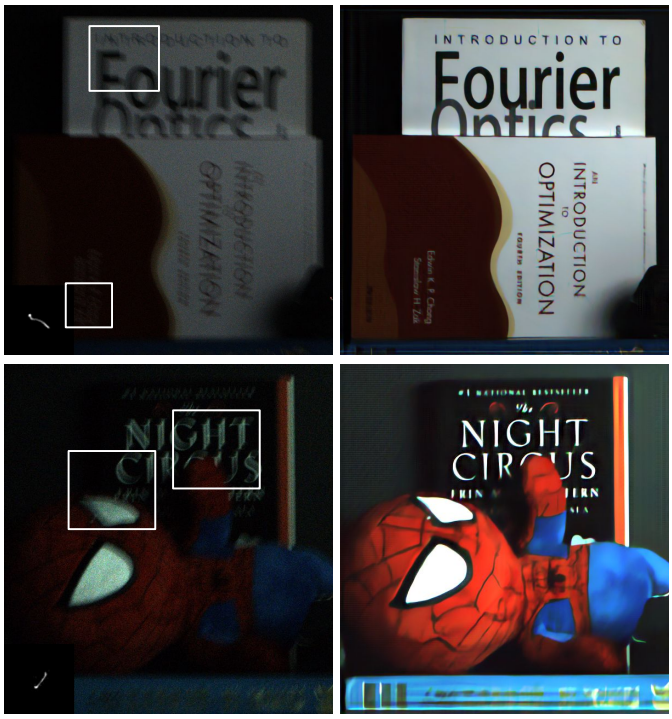
where b represents the zero-level offset of the camera which can be obtained from the metadata of the image .RAW file and is set equal to $b = 2047$. G represents the gain factor between the digital output of the sensor and the actual electrons collected by the sensor. This gain is calculated from the camera data available at [63]. Specifically, we look at the read noise of the camera in terms of digital numbers and electrons. The ratio of these two data will give the gain G. For Canon EOS Rebel T6i, at ISO 12800, the gain is estimated to be $G \approx 71$.

Our reconstruction results are shown in Figure 12. We also the compare reconstruction using proposed method with other contemporary deblurring methods (RGDN, PURE-LET, DPIR and DWDN) in Figure 13. Through a visual inspection, one can conclude that our method is able to reconstruct finer details

(a) Experimental setup, well illuminated scene (b) Real capture

Fig. 10. **Experimental setup**. For evaluation of the proposed method on real images, we collect noisy and blurred images using a DSLR as shown in the setup shown above. To capture a single degraded image, we reduce the illumination to a level that shot noise becomes visible. We blur image using camera shake. For the blur kernel, each scene contains a point source and the corresponding motion kernels can be visualized in Figure 11.

Fig. 11. **Real kernels** generated by our optical experiment setup.



(a) Real input (b) Processed

Fig. 12. **Proposed method on real data.** For a qualitative comparison of other deblurring approaches on these images, refer to Figure 13.

from the noisy and blurred image while leaving behind fewer artifacts.

Quantitative Evaluation: For evaluation of metrics such as PSNR and SSIM, we register the ground truth to the reconstruction using homography transformation to account for the differences in camera positions. The average PSNR and SSIM on the real datset for the proposed and competing methods are reported in Table IV. We outperform the second-best competing methods, i.e. [1], by 0.6dB in terms of PSNR and by 0.005 in terms of SSIM. As shown in Figure 13, when evaluating SSIM on a few patches containing text, the gap between our method and [1] becomes much wider.

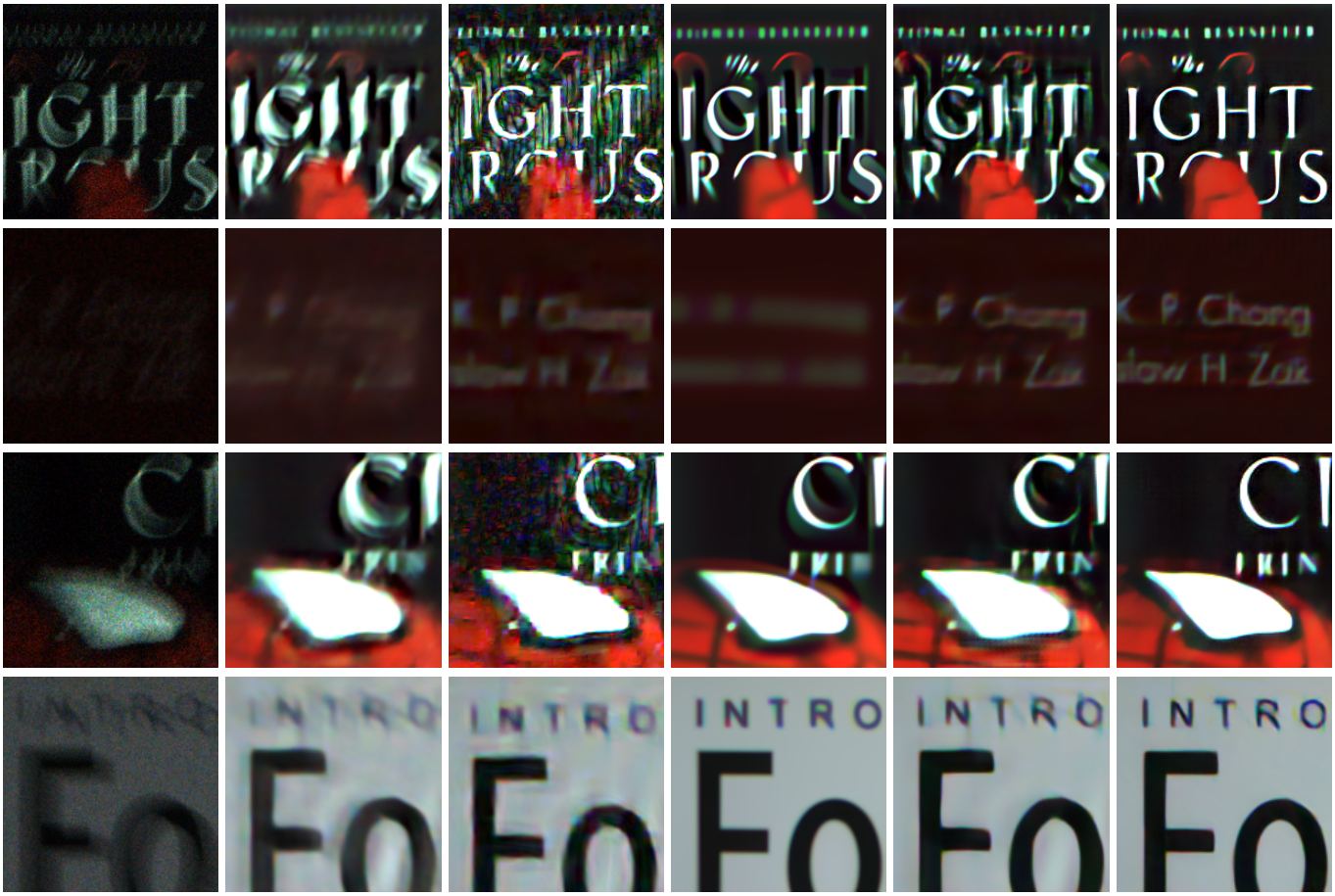
TABLE IV
PSNR (IN DB) AND SSIM EVALUATED ON REAL DATASET OF 30 IMAGES.

Method	RGDN [2]	PURE-LET [13]	DPIR [35]	DWDN [1]	Ours
PSNR	19.80	20.88	22.09	22.85	**23.48**
SSIM	0.476	0.501	0.548	0.561	**0.566**

VI. Conclusion and Future Work

In this paper, we formulated the photon-limited deblurring problem as a Poisson inverse problem. We presented an end-to-end trainable solution using a algorithm unrolling technique. We performed extensive numerical experiments to compare our approach with other existing state-of-the-art non-blind deblurring approaches and demonstrated how our method can be applied to real sensor data. Even though the present solution is focused on image deblurring, it can be easily extended to other photon-limited inverse problems such as compressive sensing, lensless imaging, and super-resolution.

The algorithm presented in this paper can be used to reconstruct a single clean image from multiple blurred images. This would allow us to take advantage of the temporal redundancy which would be necessary to obtain a meaningful clean signal in much challenging scenarios (e.g. photon level $\alpha \leq 5$).



Raw Image	RGDN [2]	PURE-LET [13]	DPIR [35]	DWDN [1]	**Ours**
PSNR (dB) / SSIM	17.43 / 0.423	20.83 / 0.478	21.50 / 0.546	21.94 / 0.575	**22.69 / 0.613**

Fig. 13. **Qualitative Comparison** We look at zoomed in regions of the reconstructed images from Figure 12 using competing methods. The average PSNR and SSIM evaluated on the given patches is provided at the bottom. From visual inspection one can see that our method is able to recover finer details compared to other methods. Note that in the first row, the DPIR output may look qualitatively similar to our result. This is because the former often tends to blur out images for a "cleaner" looking image as observed in the second row of zoomed in reconstructions.

Another interesting but challenging problem which can be attempted using the framework is *low-light blind deconvolution* i.e. recovering the clean image and blur kernel simultaneously from blurred images corrupted with photon shot noise.

Photon Limited Non-Blind Deblurring Using Algorithm Unrolling - Supplementary Material

VII. ResUNet Architectural Details

In this section, we describe the architectural details of *ResUNet* - the denoiser used in the proposed scheme. The ResUNet has an encoder-decoder structure with 4 downsampling, 1 residual convolutional and 4 upsampling blocks. The upsampling and downsampling blocks increases/decreases the size of the input by a factor of $\times 2$ and the residual blocks keeps the size of the image constant. The downsampling and upsampling blocks as shown in Figure 14 are preceded and succeded by a convolutional layer each converting the image space into feature space and vice-versa. Note that except for the strided and transposed convolution filters, appropriate padding is added to each image to keep the output images size same as the input.

Downsampling Block: Each downsampling block consists of 2 residual sub-blocks followed by strided convolutional filter which downsizes the image size by factor of 2 but also doubles the number of output features. Each residual convolutional block has a convolutional layer, followed by ReLU activation followed by another convolutional layer. The output of these three layers is added to the original input - hence the term residual sub-block.

Upsampling Block: The upsampling block is similar to the downsampling block as described above. It consists of, in the sequence given, a transposed convolutional filter followed by 2 residuals sub-blocks. The strided convolutional filter upsamples the input by a factor $\times 2$ and halves the number of output features. Similar to those in the downsampling block, the residual sub-blocks keep the output size same as that of the output.

Residual Convolutional Block: The residual convolutional blocks consist of 3 resiudal sub-blocks in series with each other. In the ResUNet, it is found after the 4 downsampling operators and before the 4 upsampling operators.

VIII. Initialization of Hyperparameters

Given the number of fixed iterations K, for which the scheme is unrolled, the hyperparameters ρ_1^k, ρ_2^k for $k = 1, 2, 3, ..., K$ are initialized using the photon level α and the blur kernel $\boldsymbol{k}$ using the network *InitNet*. In this section, we describe its architecture in detail.

The first stage consists of converting the blur kernel to a feature vector. First we ensure the kernel is of size 128×128. If not, then it is padded around the edges to ensure that. Then, we take the magnitude square of the spectrum of the kernel $\boldsymbol{k}$, which is of the same size as the kernel i.e. 128×128. Then the spectrum-squared is used as the actual input to the convolutional block of the InitNet.

The convolutional block consists of 4 consecutive downsampling blocks (different from that of ResUNet). The details of the downsampling blocks are provided in Figure 16 and the output is half the size of the input image. After the downsampling blocks the output, of size $8 \times 16 \times 16$ is then flattened to a vector, concatenated with a scalar α to form a vector of size 2049×1. This is used as input the the fully connected part of the InitialNet (also described in Figure 16)

IX. Comparison with Conventional Plug-and-Play

In this section, we discuss the details of the experiment conducted in Section IV E where we compare the performance of conventional and alternate formulation of Plug-and-Play scheme. We evaluate the performance of the 3 schemes as described in the subsection on the *BSD100* dataset. Each image in the dataset is randomly blurred by a kernel from Figure 6 from the main document a fixed photon level α. The schemes are then separately evaluated at photon levels 5, 10, 20, and 40.

For the conventional PnP, we use the pretrained FFD-Net. Since the denoiser used is blind (i.e. agnostic to noise level), the output of the scheme is independent of σ and dependent only on ρ. We use the adaptive-update rule in [17] to update ρ for each iteration. Specifically the following update rule is applied:

1) if $\Delta^{(k)} > 0.99\Delta^{(k-1)}$ then $\rho^{(k+1)} = 1.01\rho^{(k)}$
2) if $\Delta^{(k)} \leq 0.99\Delta^{(k-1)}$ then $\rho^{(k+1)} = \rho^{(k)}$

where $\Delta^{(k)}$ is defined as follows

$$\Delta^{(k)} \stackrel{\text{def}}{=} \frac{1}{3}\Big(||\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}|| + ||\boldsymbol{z}^{(k)} - \boldsymbol{z}^{(k-1)}|| + ||\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}||\Big) \tag{19}$$

We terminate the PnP scheme when Δ_k fall below 10^{-2} or after 150 iterations, whichever happens first.

For the $\boldsymbol{x}$-update step which needs to be solved as convex optimization solver, we use the memory-limited BFGS method. We use the implementation provided in `scipy.optimize` which requires the function to be optimized and the corresponding gradient value as its arguments. We use the default setting of the optimizer where it converges when the gradient norm is below 10^{-5}.

The alternate formulation using 3-operator splitting is implemented in a similar manner as that of conventional PnP. The parameters $\rho_1^{(k)}, \rho_2^{(k)}$ determine the output of the scheme

*Y. Sanghvi, A. Gnanasambandam and S. Chan are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA. Email: {ysanghvi, agnanasa, stanchan}@purdue.edu.

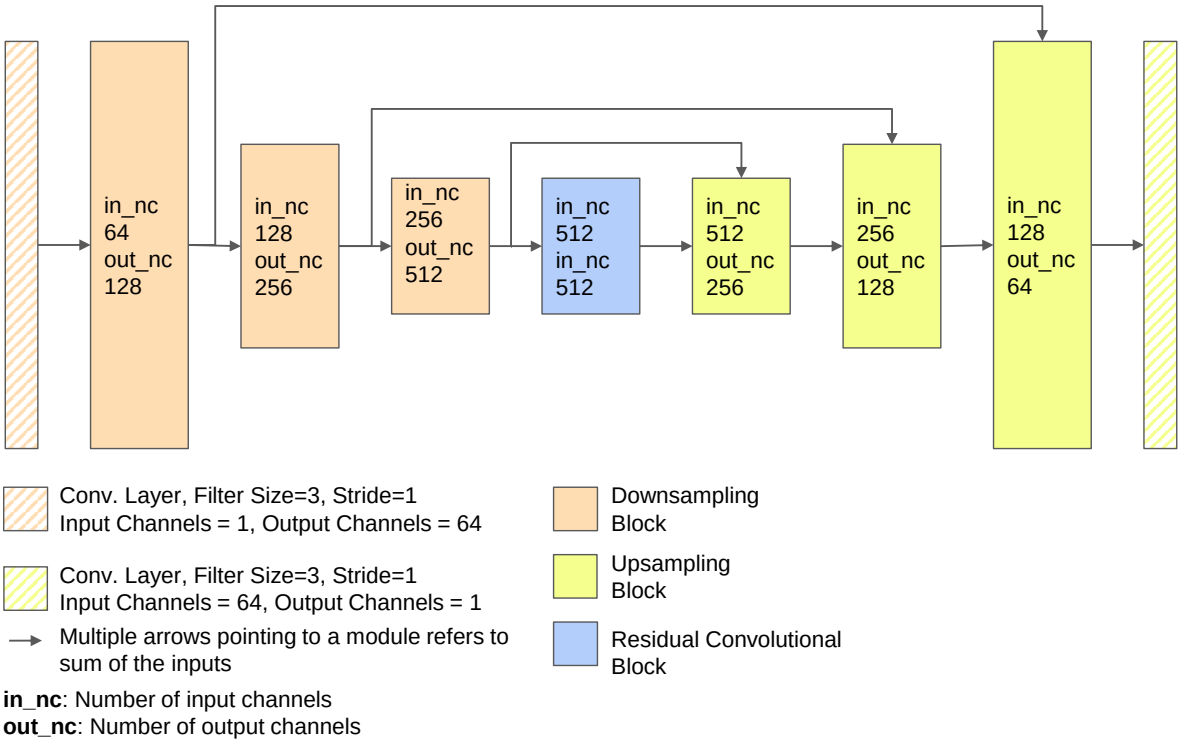


Fig. 14. Architecture details of *ResUNet* - a version of the U-Net with skip connections as described in [35]. Each downsampling block reduces the size of features by a factor of 2 and each upsampling block increases the size of features by the same factor. Note that the skip connections in ResU-Net are not concatenated to the other input going into a module but are added to them instead.

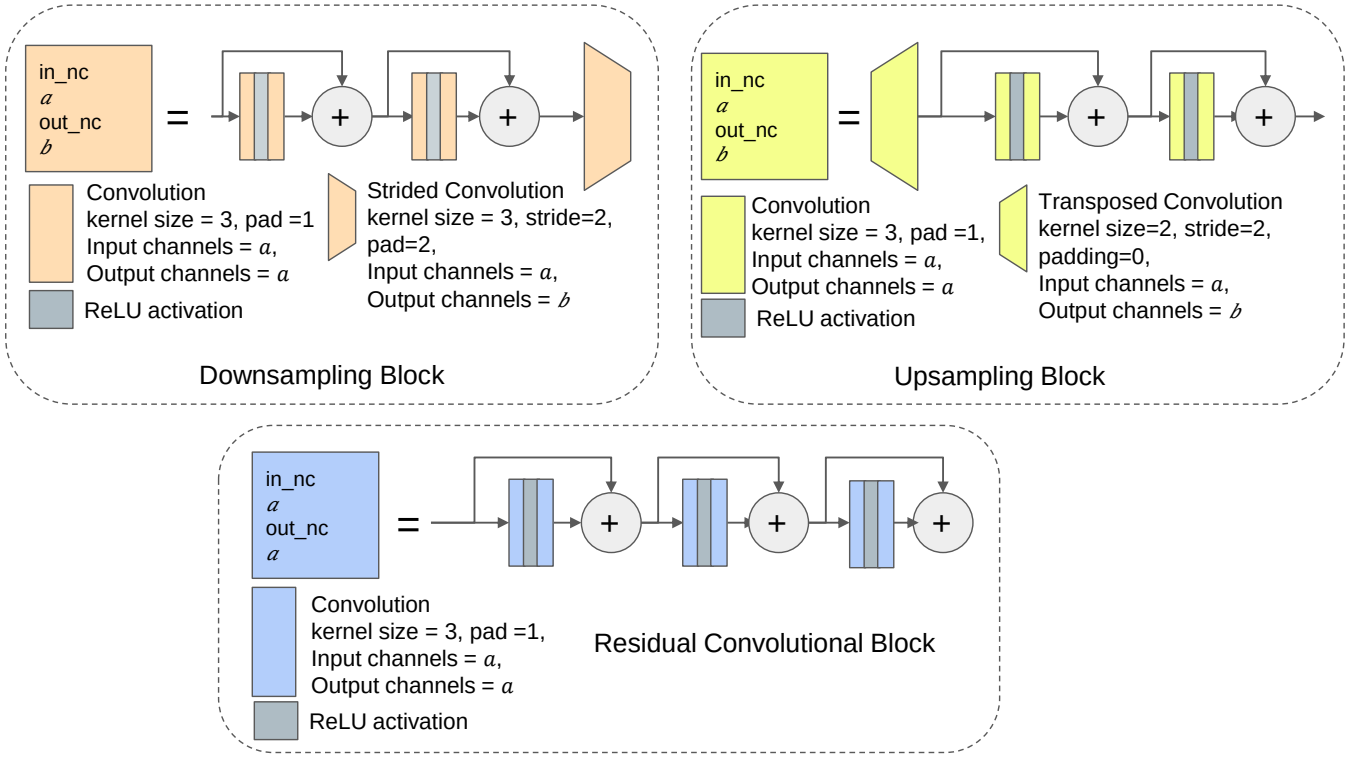


Fig. 15. Description of various blocks shown in Figure 14. Note that all convolutional layers don't have a bias term in them.

and are set to be same as each other across iterations i.e. $\rho_1^{(k)} = \rho_2^{(k)} = \rho_0^{(k)}$. $\rho_0^{(k)}$ is also adaptively updated using the following scheme

1) if $\Delta^{(k)} > 0.99\Delta^{(k-1)}$ then $\rho^{(k+1)} = 1.01\rho^{(k)}$

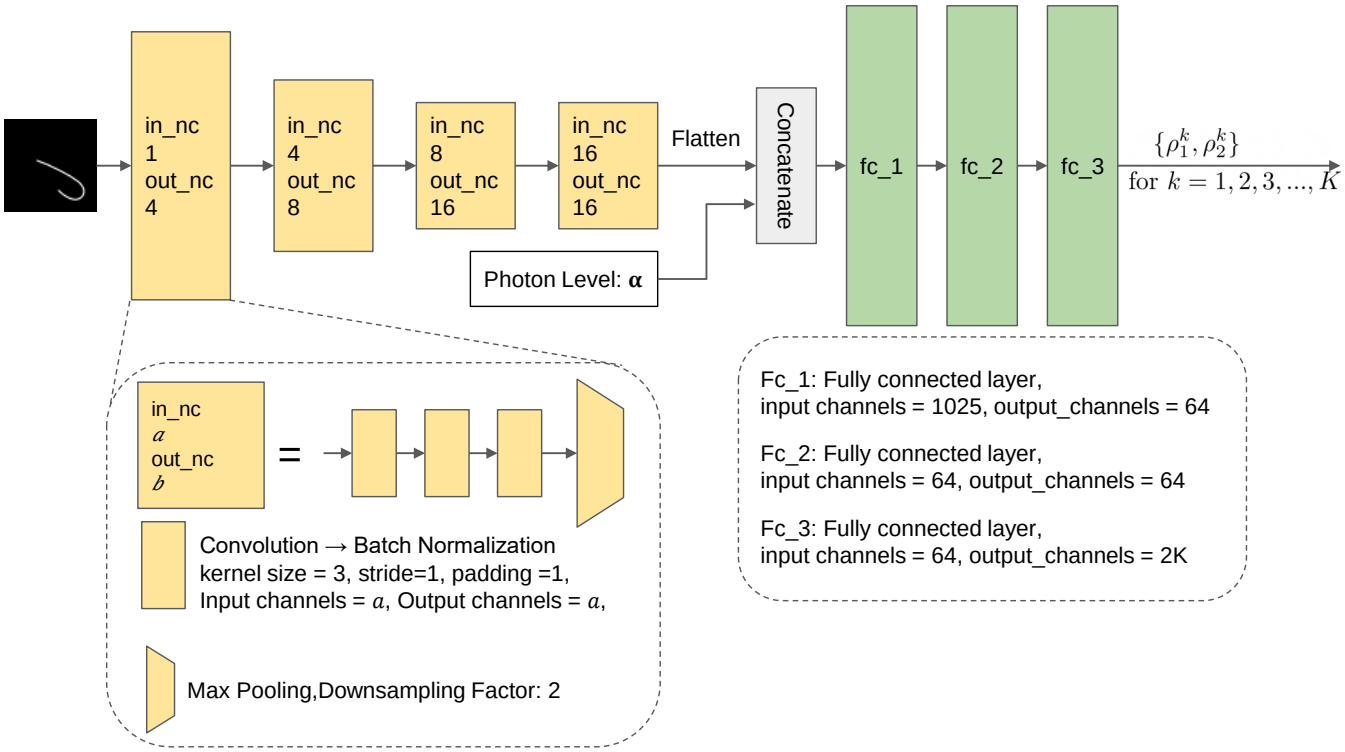


Fig. 16. Description of network architecture used to determine the hyperparameters ρ_1^k, ρ_2^k for $k = 1, 2, ..., K$

2) if $\Delta^{(k)} \leq 0.99\Delta^{(k-1)}$ then $\rho^{(k+1)} = \rho^{(k)}$
where $\Delta^{(k)}$ is defined as follows

$$\Delta^{(k)} \stackrel{\text{def}}{=} \frac{1}{5}\Big(||\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}|| + ||\boldsymbol{z}^{(k)} - \boldsymbol{z}^{(k-1)}|| + ||\boldsymbol{v}_1^{(k)} - \boldsymbol{v}_1^{(k-1)}|| + ||\boldsymbol{u}_1^{(k)} - \boldsymbol{u}_1^{(k-1)}|| + ||\boldsymbol{u}_2^{(k)} - \boldsymbol{u}_2^{(k-1)}||\Big) \quad (20)$$

For a fixed photon level α, we use the same $\rho^{(0)}$ and $\rho_0^{(k)}$ for 2-operator and 3-operator scheme respectively. Through trial-and-error, the initial ρ values are varied as shown in Table I.

Photon Level α	$\rho^{(0)}$/ $\rho_0^{(0)}$
5	200
10	400
20	800
40	1000

TABLE V

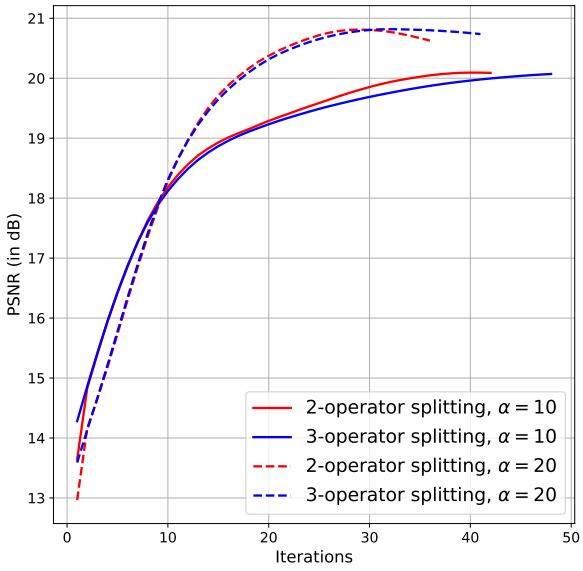


Fig. 17. Convergence of 2-operator splitting vs 3-operator splitting

X. Reconstruction Implementation Details

To reconstruct blurry and noisy real images, there are certain implementation details which need to be accounted for some assumptions for simulating Poisson blur are not valid for the real-world data.

One such example is the boundary conditions for convolution. We assume circular boundary conditions while blurring the image with a kernel since it allows us to implement convolution faster using FFT and provides a diagonalizable convolutional matrix $\boldsymbol{H}$. However, we observed artifacts in our reconstructions as the circular boundary conditions may not apply when the the opposite ends of the image are not similar. To fix this issue, we pad the input image to double its original size by reflecting along the edges and then pushing it through the network. The relevant center portion is then cropped out from the network output.

Another issue we encounter when reconstructing real-world

data is that the scalar quantity α indicative of the level of photon shot noise is not provided. We estimate it from the raw data (in terms of electrons generated) using the following heuristic.

$$\hat{\alpha} = \frac{\sum_{i=1}^{N} \boldsymbol{y}_i}{\beta N} \tag{21}$$

i.e. the photon level α is estimated to be equal to the average photon-per-pixels divided by a constant factor $\beta < 1$. Using trial-and-error, we set the constant factor $\beta = 0.33$.

To convert the individual grayscale image reconstructions i.e. *R, G1, G2, B* into a color reconstruction, we need to account for color balancing. For this purpose, we use the gray-world assumption [64]. We normalize each channel reconstruction so that mean of each channel is equal to 1. This is followed by a combining grayscale images to form an RGB image using a demosaicing process.

REFERENCES

[1] J. Dong, S. Roth, and B. Schiele, "Deep Wiener deconvolution: Wiener meets deep learning for image deblurring," in *34th Conference on Neural Information Processing Systems*, Curran Associates, Inc., 2020.

[2] D. Gong, Z. Zhang, Q. Shi, A. van den Hengel, C. Shen, and Y. Zhang, "Learning deep gradient descent optimization for image deconvolution," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 12, pp. 5468–5482, 2020.

[3] J. Kruse, C. Rother, and U. Schmidt, "Learning to push the limits of efficient FFT-based image deconvolution," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4586–4594, 2017.

[4] T. Eboli, J. Sun, and J. Ponce, "End-to-end interpretable learning of non-blind image deblurring," in *Proceedings of 16th European Conference on Computer Vision, Part XVII 16*, pp. 314–331, Springer, 2020.

[5] Y. Nan, Y. Quan, and H. Ji, "Variational-EM-based deep learning for noise-blind image deblurring," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3626–3635, 2020.

[6] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu, "Denoising prior driven deep neural network for image restoration," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 10, pp. 2305–2318, 2018.

[7] H. Wang and P. C. Miller, "Scaled heavy-ball acceleration of the richardson-lucy algorithm for 3d microscopy image restoration," *IEEE Transactions on Image Processing*, vol. 23, no. 2, pp. 848–854, 2013.

[8] J.-L. Starck and F. Murtagh, *Astronomical image and data analysis*. Springer Science & Business Media, 2007.

[9] J. W. Goodman, *Statistical optics*. John Wiley & Sons, 2015.

[10] Y. Chi, A. Gnanasambandam, V. Koltun, and S. H. Chan, "Dynamic low-light imaging with quanta image sensors," in *Proceedings of 16th European Conference on Computer Vision, Part XVII 16*, pp. 122–138, Springer, 2020.

[11] A. Gnanasambandam, O. Elgendy, J. Ma, and S. H. Chan, "Megapixel photon-counting color imaging using quanta image sensor," *Optics express*, vol. 27, no. 12, pp. 17298–17310, 2019.

[12] C. Li, X. Qu, A. Gnanasambandam, O. A. Elgendy, J. Ma, and S. H. Chan, "Photon-limited object detection using non-local feature matching and knowledge distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3976–3987, 2021.

[13] J. Li, F. Luisier, and T. Blu, "Pure-let image deconvolution," *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 92–105, 2017.

[14] W. H. Richardson, "Bayesian-based iterative method of image restoration," *J. Opt. Soc. Am.*, vol. 62, no. 1, pp. 55–59, 1972.

[15] L. B. Lucy, "An iterative technique for the rectification of observed distributions," *The Astronomical Journal*, vol. 79, p. 745, 1974.

[16] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *2013 IEEE Global Conference on Signal and Information Processing*, pp. 945–948, IEEE, 2013.

[17] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play ADMM for image restoration: Fixed-point convergence and applications," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 84–98, 2016.

[18] A. Rond, R. Giryes, and M. Elad, "Poisson inverse problems by the plug-and-play scheme," *Journal of Visual Communication and Image Representation*, vol. 41, pp. 96–108, 2016.

[19] M. Bertero, P. Boccacci, G. Desiderà, and G. Vicidomini, "Image deblurring with Poisson data: from cells to galaxies," *Inverse Problems*, vol. 25, no. 12, p. 123006, 2009.

[20] L. A. Shepp and Y. Vardi, "Maximum likelihood reconstruction for emission tomography," *IEEE Transactions on Medical Imaging*, vol. 1, no. 2, pp. 113–122, 1982.

[21] N. Dey, L. Blanc-Feraud, C. Zimmer, P. Roux, Z. Kam, J.-C. Olivo-Marin, and J. Zerubia, "Richardson–lucy algorithm with total variation regularization for 3d confocal microscope deconvolution," *Microscopy Research and Technique*, vol. 69, no. 4, pp. 260–266, 2006.

[22] M. Laasmaa, M. Vendelin, and P. Peterson, "Application of regularized richardson–lucy algorithm for deconvolution of confocal microscopy images," *Journal of Microscopy*, vol. 243, no. 2, pp. 124–140, 2011.

[23] M. A. Figueiredo and J. M. Bioucas-Dias, "Restoration of Poissonian images using alternating direction optimization," *IEEE transactions on Image Processing*, vol. 19, no. 12, pp. 3133–3145, 2010.

[24] Z. T. Harmany, R. F. Marcia, and R. M. Willett, "This is SPIRAL-TAP: Sparse Poisson intensity reconstruction algorithms—theory and practice," *IEEE Transactions on Image Processing*, vol. 21, no. 3, pp. 1084–1096, 2011.

[25] R. D. Nowak and E. D. Kolaczyk, "A statistical multiscale framework for Poisson inverse problems," *IEEE Transactions on Information Theory*, vol. 46, no. 5, pp. 1811–1825, 2000.

[26] T. Blu and F. Luisier, "The SURE-LET approach to image denoising," *IEEE Transactions on Image Processing*, vol. 16, no. 11, pp. 2778–2786, 2007.

[27] F. Xue, F. Luisier, and T. Blu, "Multi-Wiener SURE-LET deconvolution," *IEEE Transactions on Image Processing*, vol. 22, no. 5, pp. 1954–1968, 2013.

[28] F. J. Anscombe, "The transformation of Poisson, binomial and negative-binomial data," *Biometrika*, vol. 35, no. 3/4, pp. 246–254, 1948.

[29] M. Makitalo and A. Foi, "Optimal inversion of the Anscombe transformation in low-count Poisson image denoising," *IEEE transactions on Image Processing*, vol. 20, no. 1, pp. 99–109, 2010.

[30] F. Luisier, C. Vonesch, T. Blu, and M. Unser, "Fast interscale wavelet denoising of Poisson-corrupted images," *Signal processing*, vol. 90, no. 2, pp. 415–427, 2010.

[31] B. Zhang, J. M. Fadili, and J.-L. Starck, "Wavelets, ridgelets, and curvelets for Poisson noise removal," *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1093–1108, 2008.

[32] L. Azzari and A. Foi, "Variance stabilization for noisy+ estimate combination in iterative Poisson denoising," *IEEE signal processing letters*, vol. 23, no. 8, pp. 1086–1090, 2016.

[33] S. Sreehari, S. V. Venkatakrishnan, B. Wohlberg, G. T. Buzzard, L. F. Drummy, J. P. Simmons, and C. A. Bouman, "Plug-and-play priors for bright field electron tomography and sparse interpolation," *IEEE Transactions on Computational Imaging*, vol. 2, no. 4, pp. 408–423, 2016.

[34] R. Ahmad, C. A. Bouman, G. T. Buzzard, S. H. Chan, E. T. Reehorst, and P. Schniter, "Plug and play methods for magnetic resonance imaging.," 2019.

[35] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3929–3938, 2017.

[36] T. He, Y. Sun, B. Chen, J. Qi, W. Liu, and J. Hu, "Plug-and-play inertial forward–backward algorithm for Poisson image deconvolution," *Journal of Electronic Imaging*, vol. 28, no. 4, p. 043020, 2019.

[37] U. S. Kamilov, H. Mansour, and B. Wohlberg, "A plug-and-play priors approach for solving nonlinear imaging inverse problems," *IEEE Signal Processing Letters*, vol. 24, no. 12, pp. 1872–1876, 2017.

[38] Y. Sun, B. Wohlberg, and U. S. Kamilov, "An online plug-and-play algorithm for regularized image reconstruction," *IEEE Transactions on Computational Imaging*, vol. 5, no. 3, pp. 395–408, 2019.

[39] G. T. Buzzard, S. H. Chan, S. Sreehari, and C. A. Bouman, "Plug-and-play unplugged: Optimization-free reconstruction using consensus equilibrium," *SIAM Journal on Imaging Sciences*, vol. 11, no. 3, pp. 2001–2020, 2018.

[40] S. H. Chan, "Performance analysis of plug-and-play admm: A graph signal processing perspective," *IEEE Transactions on Computational Imaging*, vol. 5, no. 2, pp. 274–286, 2019.

[41] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, "Plug-and-play methods provably converge with properly trained denoisers," in *International Conference on Machine Learning*, pp. 5546–5557, PMLR, 2019.

[42] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (RED)," *SIAM Journal on Imaging Sciences*, vol. 10, no. 4, pp. 1804–1844, 2017.

[43] R. Cohen, M. Elad, and P. Milanfar, "Regularization by denoising via fixed-point projection (RED-PRO)," *arXiv preprint arXiv:2008.00226*, 2020.

[44] E. T. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations," *IEEE transactions on computational imaging*, vol. 5, no. 1, pp. 52–67, 2018.

[45] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th international conference on international conference on machine learning*, pp. 399–406, 2010.

[46] K. Zhang, L. V. Gool, and R. Timofte, "Deep unfolding network for image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3217–3226, 2020.

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

[48] Y. Yang, J. Sun, H. Li, and Z. Xu, "Deep ADMM-net for compressive sensing mri," in *Proceedings of the 30th international conference on neural information processing systems*, pp. 10–18, 2016.

[49] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *arXiv preprint arXiv:1912.10557*, 2019.

[50] D. Gilton, G. Ongie, and R. Willett, "Deep equilibrium architectures for inverse problems in imaging," *arXiv preprint arXiv:2102.07944*, 2021.

[51] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, Springer, 2015.

[52] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[53] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on scientific computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[54] M. A. Figueiredo and J. M. Bioucas-Dias, "Deconvolution of Poissonian images using variable splitting and augmented lagrangian optimization," in *2009 IEEE/SP 15th Workshop on Statistical Signal Processing*, pp. 733–736, IEEE, 2009.

[55] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[56] G. Boracchi and A. Foi, "Modeling the performance of image restoration from motion blur," *IEEE Transactions on Image Processing*, vol. 21, no. 8, pp. 3502–3517, 2012.

[57] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding and evaluating blind deconvolution algorithms," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1964–1971, IEEE, 2009.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[59] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[60] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, pp. 416–423, IEEE, 2001.

[61] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[62] O. A. Elgendy, A. Gnanasambandam, S. H. Chan, and J. Ma, "Low-light demosaicking and denoising for small pixels using learned frequency selection," *IEEE Transactions on Computational Imaging*, vol. 7, pp. 137–150, 2021.

[63] "Photons to Photos." https://www.photonstophotos.net/. Accessed: Sep-9-2021.

[64] G. Buchsbaum, "A spatial processor model for object colour perception," *Journal of the Franklin institute*, vol. 310, no. 1, pp. 1–26, 1980.