The Secrets of Non-Blind Poisson Deconvolution

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

***Abstract*—Non-blind image deconvolution has been studied for several decades but most of the existing work focuses on blur instead of noise. In photon-limited conditions, however, the excessive amount of shot noise makes traditional deconvolution algorithms fail. In searching for reasons why these methods fail, we present a systematic analysis of the Poisson non-blind deconvolution algorithms reported in the literature, covering both classical and deep learning methods. We compile a list of five "secrets" highlighting the do's and don'ts when designing algorithms. Based on this analysis, we build a proof-of-concept method by combining the five secrets. We find that the new method performs on par with some of the latest methods while outperforming some older ones.**

***Index Terms*—Photon-limited, deconvolution, inverse problems, deblurring, shot noise.**

I. Introduction

A. From Gaussian to Poisson Deconvolution

IMAGE deconvolution is one of the most fundamental problems in image restoration. When the blur kernel is fixed and given, the problem is known as *non-blind deconvolution*. For spatially invariant blur and additive i.i.d. Gaussian noise, the goal of deconvolution is to recover $\mathbf{x} \in \mathbb{R}^N$ from the equation

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

where $\mathbf{n} \in \mathbb{R}^N$ is the i.i.d. Gaussian noise, and $\mathbf{H} \in \mathbb{R}^{N \times N}$ is the blur kernel represented as a convolution matrix [1], [2]. The inverse problem associated with (1) has been studied for a few decades, with an extensive list of methods, both classical [3], [4], [5], [6], [7], [8], [9], [10], [11] and deep-learning based [12], [13], [14], [15], [16], [17], [18].

With such a large volume of prior work, it would appear that the problem is solved. However, as we push the limit of image deconvolution to *low-light* conditions, the problem remains wide open. Moreover, the growth of advanced photon counting image sensors and the need for extreme low light imaging applications [19], [20], [21], [22], [23], [24] makes the problem even more interesting than before. As people have shown in [25], even an ideal image sensor with zero read noise cannot escape from the photon shot noise. Thus, signal processing at this limit remains critical.

The change from a well-illuminated condition to a low-light condition is not just about switching the Gaussian model to a Poisson model [1]

$$\mathbf{y} = \text{Poisson}\{\alpha\mathbf{H}\mathbf{x}\}, \tag{2}$$

where α is the average number of photons in the scene [26] and the clean image $\mathbf{x}$ is assumed to be normalized to $[0, 1]$. The increased difficulty is not associated with the unbounded-below and the non-differentiable-at-origin property of the Poisson negative-log-likelihood, but the magnitude of the noise exhibited in the data. In a typical low light condition, the mean photon count can be as low as ten photons per pixel. At this photon level, the random fluctuation of the signal would cause many algorithms to fail.

The impact of noise in Poisson deconvolution is noticeable in every step of a deconvolution algorithm. Since there is noise, it becomes much harder for an algorithm to invert the blur (usually in the Fourier space) and remove the noise. Deep learning algorithms also suffer from heavy noise because extracting features from the image becomes more difficult. In fact, Poisson deconvolution has only been discussed in a few deep-learning papers [28], [29], [30], [31].

B. Scope and Contributions

Given the success of Gaussian-noise based image deconvolution algorithms, we believe that the lessons learned in the past can shed light on understanding the Poisson problem. To this end, we analyze a large collection of non-blind deconvolution algorithms reported in the literature. We look into the design details of each method and compile a list of do's and don'ts we learned from these methods.

As a preview of our results, we show in Fig. 1 the image reconstruction results of three methods published in the literature: PURE-LET [27] (T-IP, 2017), DWDN [14] (T-PAMI, 2022), and USRNet [18] (CVPR, 2020). All three methods are fine-tuned using Poisson data. In the same figure, we also report a proof-of-concept method by combining the "secrets" we learned in this paper. We stress that this proof-of-concept method is not meant to become a state-of-the-art

Manuscript received 4 September 2023; revised 9 December 2023 and 8 February 2024; accepted 12 February 2024. Date of publication 27 February 2024; date of current version 1 March 2024. This work was supported by the National Science Foundation under Grant IIS-2133032, Grant ECCS-2030570, and Grant CCF-1763896. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Michael T McCann. *(Corresponding author: Yash Sanghvi.)*

Abhiram Gnanasambandam is with Samsung Research America, Plano, TX 75024 USA (e-mail: abhiram.g94@gmail.com).

Yash Sanghvi and Stanley H. Chan are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: ysanghvi@purdue.edu; stanchan@purdue.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TCI.2024.3369414, provided by the authors.

Digital Object Identifier 10.1109/TCI.2024.3369414

[1]The Poisson model we study in this paper is a simplification of the actual image formation process which should involve dark current, read noise, etc.. However, given that the Poisson problem is already difficult enough, we decided to focus on it in this paper.

Fig. 1. Overview. The goal of this paper is to identify factors that will benefit Poisson image deblurring. Shown in this example are a simulated blurry and noisy image (where the noise is Poisson), and the corresponding image reconstruction results. The proposed method (to be discussed in Section IV) is just a combination of the five factors we identified, *without* introducing any new architectures.

but rather a confirmation of ideas described in the paper. Interestingly, the performance of this proof-of-concept is quite satisfactory.

So, what are our observations? We found the following five "secrets" of non-blind Poisson deconvolution:

i) *Wiener filter is recommended:* While some networks perform deconvolution and denoising simultaneously, we find that it is better to decouple the deconvolution part using a Wiener filter so that we can leverage the fact that the blur kernel is known. Of course, we assume that the blur is spatially invariant.

ii) *Iteration is recommended:* Many networks estimate the image in a single shot. We find that iterative algorithms are more effective. For deep neural networks, the iterative algorithms can be implemented via algorithm unrolling.

iii) *Feature space is recommended:* It is better to perform deconvolution in the feature space than in the spatial domain.

iv) *Poisson likelihood is not needed:* When handling Poisson noise, there is no need to use customized tools such as variance stabilizing transform or the Poisson likelihood. Any architecture for Gaussian noise also works for Poisson.

v) *Learning the hyper-parameters is recommended:* Some algorithms estimate the hyperparameters using an off-the-shelf method or a heuristic rule. We find that end-to-end learning of the hyperparameters helps the performance.

This paper focuses on non-generative methods. Our analysis does not cover generative models (e.g., generative adversarial networks or denoising diffusion probabilistic models) because they belong to a different category of approaches. We do not consider *blind* deconvolution algorithms because we do not estimate the blur kernel.

II. Analysis of Prior Methods

Given the large number of papers published for non-blind image deconvolution, it would be unrealistic to comment on every single method. The approach we take here is to focus on a *representative* subset of existing methods. However, the selection of the representative methods would require some work. In what follows, we first list a number of Poisson deconvolution methods. We group them, and discuss their attributes. Afterward, we select the representative methods and discuss their design philosophies.

A. Prior Methods

To help readers visualize the methods being studied in this paper, we summarize them in Table I. These methods can be categorized into two main classes:

Classical Methods: By classical methods, we mean methods that do not require learning. These methods are typically developed before the deep-learning era. In this paper, we select three representative methods with code publicly available:

- PURE-LET, by Li and Blu [27], is a non-iterative deblurring algorithm that uses the Poisson unbiased risk estimator (PURE) as a metric to guide the steps in linear expansion thresholding (LET). The thresholding idea used here is similar to several other paper [36], [37], [38], [39].
- VSTP, by Azzari and Foi [32], uses the variance stabilization transform (VST) to equalize the variance of the Poisson random variable. Then, a deblurring algorithm is applied to handle the blur.
- Deconvtv, by Chan et al. [33], uses total variation for Gaussian noise removal. Its performance is not necessarily the best compared to other total variation solvers such as [40], [41], [42], [43], [44], [45], [46], but its code is readily available for experiments.

We acknowledge that there are plenty of other classical methods, such as [11], [47], [48], [49], [50], [51], [52], [53], [54]. These papers made great contributions in improving the prior models of the images so that deblurring and denoising can be more effective. Some of these methods perform very well whereas some are similar to the three abovementioned methods. For the concreteness of this paper and considering the availability of their codes, we decided to focus on the ones we mentioned above.

Deep-Learning Methods: While deep learning based deconvolution algorithms are abundant, many of them are *blind* algorithms. For non-blind methods, we consider nine of them.

- Deep Wiener Deconvolution Network (DWDN) [14] is a deep neural network that performs Wiener deconvolution in the feature space followed by a decoder. Similarly, [55] performs Wiener deconvolution followed by an artifact removoval network and INFWIDE [56] adds a cross-residual fusion module. In this paper, we focus on DWDN for clarity and simplicity.
- KerUnc [16], CPCR [34], USRNet [18], PhDNet [29], and [15], [57], [58] perform fixed iteration unrolling of alternating direction method of multipliers (ADMM), half

TABLE I
WE STUDY A COMPREHENSIVE LIST OF METHODS AS SHOWN IN THE TABLE BELOW

	Classical Methods			Deep Learning Methods								
	PURE-LET [27]	VSTP [32]	Deconvtv [33]	DWDN [14]	SVMAP [12]	KerUnc [16]	CPCR [34]	RGDN [15]	PhDNet [29]	USRNet [18]	DPIR [35]	DWKF [17]
Neural Network?	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Decoupling?	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Poisson Likelihood?	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Iterative?	✗	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓
Learned parameter?	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗
Feature space?	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗

The gray-colored methods are selected for further analysis to identify the "secrets" of poisson deconvolution.

quadratic splitting or gradient descent methods followed by end-to-end training.

- DPIR [35] uses the plug-and-play (PnP) based ADMM optimization to solve the problem.
- DWKF [17] is an iterative method that uses kernel prediction networks for imposing the image priors.

B. Attributes of the Methods

With more than ten methods listed in Table I, it would be helpful if we could further categorize them according to their attributes. The attributes we highlight here will be used to inform the do's and don'ts of designing an algorithm.

- **Neural network?** This attribute asks if the method uses a neural network - either trained end-to-end [13] or as a pretrained block [18]. By definition, all classical methods are treated as non-neural network methods in this paper.
- **Decoupling?** Decoupling means that a method handles the deblurring step and the denoising step separately. The decoupling can be realized via variable splitting (e.g., in ADMM), or via a two-stage operation (e.g., in PURE-LET). For neural networks, we say that it employs a decoupling strategy if there are modules explicitly performing deblurring and are separated from denoising.
- **Poisson likelihood?** If a method explicitly uses the Poisson likelihood in an algorithm, then this attribute is satisfied. Some methods, usually deep neural networks, do not incorporate the Poisson likelihood in its algorithm design, for example [14], [34]
- **Iterative?** Both classical and deep learning methods can be iterative. The iteration can occur in the form of an actual iteration (as in optimization steps) or algorithm unrolling in deep learning methods.
- **Learned parameters?** All restoration methods have a set of hyperparameters. If these hyperparameters are picked manually, we say that the parameters are not learned. In contrast, if the hyper-parameters are simultaneously selected by the learning algorithm, then we say that the parameters are learned.
- **Feature space?** For some deep learning methods, the deconvolution does not take place in the spatial domain [27] but in the feature space [13], [56]. We check this box to reflect the property.

C. Design Principles

We now discuss the design principles of the methods shown in Table I. To narrow down the discussion to a smaller set of methods, we compared the methods' performance on a testing dataset. The execution of the experiment is described in Section III when we discuss the five secrets of Poisson deconvolution. For the sake of brevity, the detailed numbers are reported in the Supplementary Material. Based on the performance of the methods, we select five leading methods that cover four categories. They are:

1) Traditional, non-iterative: PURE-LET [27]
2) Traditional, iterative: VSTP [32]
3) Neural-network, non-iterative: DWDN [14]
4) Neural-network, iterative: USRNet [18], PhDNet [28]. Two methods were chosen because of their similar performance.

1) PURE-LET [27]: The core idea of PURE-LET is to construct multiple initial estimates using the Wiener filter, which is essentially a Fast-Fourier transform (FFT) based deconvolution. Given the blur matrix $\mathbf{H}$, PURE-LET estimates a set of K initial guesses via

$$\widehat{\mathbf{x}}_k^{\text{Wiener}} = \texttt{Wavelet}\left[\left(\mathbf{H}^T\mathbf{H} + \lambda_k \mathbb{I}\right)^{-1}\mathbf{H}^T\mathbf{y}\right], \tag{3}$$

where $k = \{1, 2, \ldots K\}$ denotes the kth Wiener estimate, and λ_k is the kth hyperparameter. The operator `Wavelet` denotes the wavelet thresholding, which is the method PURE-LET used to clean up the estimates. The estimates are then linearly combined in such a way that they minimize the mean square error, i.e.,

$$\widehat{\mathbf{x}} = \sum_{k=1}^{K} a_k \cdot \widehat{\mathbf{x}}_k^{\text{Wiener}}, \tag{4}$$

where $\{a_k \mid k = 1, \ldots, K\}$ are the optimal combination weights determined by minimizing the Poisson unbiased risk estimate (PURE).

A conceptual diagram of PURELET is shown in Fig. 2. Referring to Table I, PURELET employs a decoupling strategy by separating the deconvolution step and the denoising step. The Poisson likelihood is used to compute the risk estimate, but it was not used for the deconvolution step which is a filter bank of Wiener filters.

2) DWDN [14]: DWDN has many similarities to PURE-LET. Instead of applying the Wiener filter on the images, DWDN applies it to the features:

$$\widehat{\mathbf{x}}_k^{\text{feature}} = \left(\mathbf{H}^T\mathbf{H} + \lambda_k \mathbb{I}\right)^{-1}\mathbf{H}^T\mathcal{F}_k^{\text{feature}}(\mathbf{y}), \tag{5}$$

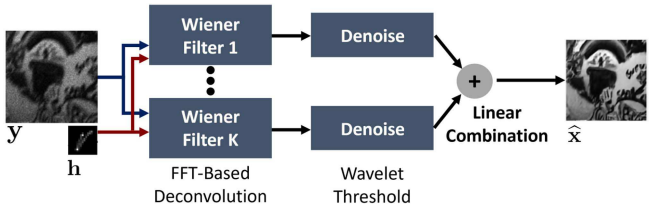


Fig. 2. PURE-LET [27] constructs a bank of Wiener filters to deblur the image, followed by image denoisers.

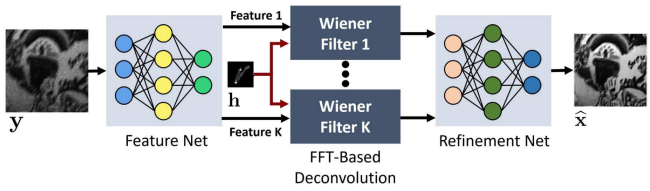


Fig. 3. DWDN [14]. While it shares similarities with PURELET, it performs Wiener deconvolution in the feature space instead of the image space.

where $\mathcal{F}_k^{\text{feature}}(\cdot)$ is a neural network trained to produce features. The estimated deblurred features $\{\widehat{\mathbf{x}}_1^{\text{feature}}, \widehat{\mathbf{x}}_2^{\text{feature}}, \ldots, \widehat{\mathbf{x}}_K^{\text{feature}}\}$ are then fed to another neural network for refinement $\mathcal{F}_{\text{refine}}$ to obtain the final output $\widehat{\mathbf{x}}$:

$$\widehat{\mathbf{x}} = \mathcal{F}_{\text{refine}}\left\{\widehat{\mathbf{x}}_1^{\text{feature}}, \widehat{\mathbf{x}}_2^{\text{feature}}, \ldots, \widehat{\mathbf{x}}_K^{\text{feature}}\right\}. \tag{6}$$

The feature networks $\{\mathcal{F}_k^{\text{feature}} \mid k = 1, \ldots, K\}$ and the refinement network $\mathcal{F}_{\text{refine}}$ are trained end-to-end. When the mean squared error (MSE) loss is used, DWDN and PURE-LET both aim to find the MMSE estimate.

A schematic diagram of DWDN is shown in Fig. 3. If we compare DWDN with PURE-LET, we recognize that the overall multi-channel filter bank idea is the same. The only difference is that DWDN performs the deconvolution operations in the feature space. The denoisers are also replaced by neural networks. Moreover, since DWDN does not need to estimate the risk (as in PURE-LET), the Poisson likelihood is not considered.

3) VSTP [32]: VSTP extends the idea of PURE-LET to make it iterative. VSTP starts with a single estimate of the deblurred image $\widehat{\mathbf{x}}^{\text{Wiener}}$ instead of the multiple estimates used in PURE-LET. However, the overall concept of decoupling the deconvolution and the denoising steps remain the same.

An interesting idea of VSTP is to iteratively update the denoising step so that each denoising step can be "mild". To do so, a linear combination of $\widehat{\mathbf{x}}^{\text{Wiener}}$ and the denoised estimate from the previous iteration $\widehat{\mathbf{x}}_{t-1}$ is obtained via

$$\widehat{\mathbf{x}}_t^{\text{data}} = \lambda_t \widehat{\mathbf{x}}_t + (1 - \lambda_t)\widehat{\mathbf{x}}^{\text{Wiener}} \tag{7}$$

A variance stabilizing transform (VST) is then used to stabilize the spatially varying noise strength of $\widehat{\mathbf{x}}_t^{\text{data}}$, which is then denoised with `Denoiser`,

$$\widehat{\mathbf{x}}_t = \texttt{Denoiser}\left[\text{VST}\left(\widehat{\mathbf{x}}_t^{\text{data}}\right)\right]. \tag{8}$$

The iteration continues until the stopping criteria are met.

In VSTP, the variance stabilizing transform is more of a technical need because the noise is spatially varying. The rationale of using VST is that when the photon level is not too low, VST is able to stabilize the variance so that the spatially varying variance will become invarying.

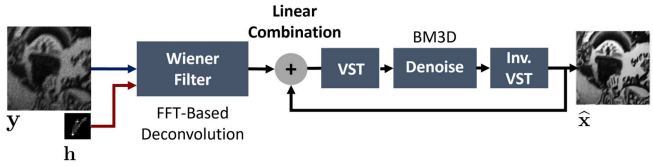


Fig. 4. VSTP [32] applies variance stabilizing transform and a denoiser for the denoising step. The denoising step is also repeated in an iterative manner to improve the performance.

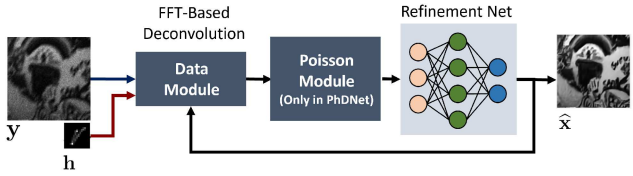


Fig. 5. USRNet [18], PhDNet [28] is an optimization-based algorithm where the problem is decoupled into deconvolution, Poisson data, and image denoising. The method is iterative; in deep neural networks, the iterations are realized via algorithm unrolling.

A schematic diagram of VSTP is shown in Fig. 4. In the literature, people sometimes refer the denoising module as transform-denoise [19].

4) PhDNet [28] and USRNet [18]: Both methods are based on maximizing the posterior probability (hence they are a maximum-a-posteriori (MAP) estimator). More specifically, the estimate is obtained by solving the optimization:

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\text{argmax}}\left[\log \mathbb{P}(\mathbf{y}|\mathbf{x}) + \log \mathbb{P}(\mathbf{x})\right], \tag{9}$$

where $\mathbb{P}(\mathbf{y}|\mathbf{x})$ is the likelihood term and $\mathbb{P}(\mathbf{x})$ is the natural image prior. USRNet models the problem by assuming that the noise is Gaussian (without considering the fact that the true noise distribution is Poisson). Thus, in USRNet, the likelihood term is

$$\log \mathbb{P}(\mathbf{y}|\mathbf{x}) = -\|\mathbf{y} - \alpha\mathbf{H}\mathbf{x}\|^2. \tag{10}$$

PhDNet explicitly takes into consideration of the Poisson noise, which leads to the following likelihood term

$$\log \mathbb{P}(\mathbf{y}|\mathbf{x}) = -\alpha\mathbf{1}^T\mathbf{H}\mathbf{x} + \mathbf{y}^T \log(\alpha\mathbf{H}\mathbf{x}), \tag{11}$$

where $\mathbf{1}$ is a vector with all ones.

Both methods solve the optimization using an unrolled neural network. Two steps are common for both methods:

- The inversion module is similar to a Wiener filter. For iteration t, it is given by

$$\widehat{\mathbf{x}}_t^{\text{data}} = \left(\mathbf{H}^T\mathbf{H} + \alpha\mathbb{I}\right)^{-1}\left(\mathbf{H}^T\mathbf{y} + \alpha\widehat{\mathbf{x}}_{t-1}\right). \tag{12}$$

- The Gaussian denoising module, which can be considered as a refinement step:

$$\widehat{\mathbf{x}}_t = \mathcal{F}^{\text{refine}}\left(\widehat{\mathbf{x}}_t^{\text{data}}\right) \tag{13}$$

PhDNet has an additional step in each iteration to deal with the Poisson noise.

A schematic diagram of the methods is shown in Fig. 5. On neural networks, the iterations are implemented via algorithm

TABLE II
COMPARISON OF DECONVOLUTION METHODS IN THE SPATIAL DOMAIN OR THE FEATURE SPACE

	Config. I	Config II	Config III	Config IV
Features	×	×	✓	✓
3 Wieners	×	✓	×	✓
10 ppp	22.08 / 0.66	22.13 / 0.66	22.43 / 0.68	**22.47 / 0.68**
30 ppp	23.09 / 0.70	23.11 / 0.70	23.41 / 0.71	**23.47 / 0.72**
50 ppp	23.59 / 0.71	23.64 / 0.71	23.91 / **0.73**	**23.97 / 0.73**

The numbers represent PSNR(dB)/SSIM.

TABLE III
COMPARISON OF TWO UNROLLED ITERATIVE ALGORITHMS

	1 Itr.	2 Itr.	4 Itr.	8 Itr.
USRNet [18]	24.40 / 0.70	24.79 / 0.74	24.88 / 0.76	24.89 / 0.76
PhDNet [28]	24.40 / 0.69	24.77 / 0.73	24.87 / 0.76	24.87 / 0.76

USRNET uses a gaussian likelihood whereas phDNet uses a poisson likelihood. The medium photon level of the images in this experiment is set to 30 PPP. The numbers represent PSNR(dB)/SSIM.

TABLE IV
EFFECT OF VST TO A LOW-LIGHT DENOISING TASK

ppp	w/ VST	w/o VST
10 ppp	28.28 / 0.81	28.34 / 0.81
30 ppp	30.61 / 0.85	30.67 / 0.86
50 ppp	31.77 / 0.88	31.82 / 0.89

In this experiment, the denoiser is resunet [18].The numbers represent PSNR(dB)/SSIM.

TABLE V
IMPACT OF LEARNING HYPERPARAMETERS

ppp	DWDN	w/ DWDN + learned para.
10	22.43 / 0.68	**22.50 / 0.69**
30	23.41 / 0.71	**23.49 / 0.72**
50	23.92 / 0.73	**23.99 / 0.74**

The numbers represent PSNR(dB)/SSIM.

unrolling. That is, we unfold the optimization algorithm into a fixed number of blocks where each block is implemented via a neural network. When looping through this fixed number of blocks, effectively we perform an iterative algorithm. For additional details about algorithm unrolling, we refer the readers to [28], [29], [59].

III. THE SECRETS

In Section II we analyzed the structures of the prior methods, but this alone does not tell us much about the secrets of Poisson deconvolution. In this section, our goal is to dive into the details by conducting a series of experiments. From the experimental results, we then draw conclusions about the influencing factors for Poisson deconvolution. Some of the discussions are based on the main experimental result Table VI, which are presented in Section V.

A. Experimental Setting

Our approach to analyzing the performance of the prior methods is based on a series of carefully designed experiments. Since this is an empirical approach, we first state the background experimental settings.

First of all, we consider classical methods and deep learning methods separately, because deep learning methods require training. To make sure that the comparisons are fair, we retrain all the deep learning methods with the exact same training dataset, same training loss, and fine-tune the hyper-parameters to maximize their performances.

For training, we use images from the Flickr2K [60] dataset. We generate 500 random kernels based on [61]. These 500 kernels consist of five groups of sizes and each group has 100. The sizes are 9×9, 18×18, 27×27, 36×36, and 45×45. In addition, we generate 64 Gaussian kernels of varying anisotropy with the blur parameter σ between 0.1 and 5. Images of size 128×128 are cropped randomly from the dataset and then each image is blurred using a random kernel among these 500+64 = 564 kernels assuming circular boundary conditions. For noise, we assume that the photons per pixel (ppp) ranged between 5 and 80, which can be adjusted by varying α in (2). The input images are then downscaled by alpha to make sure that the range of the input images remains roughly the same.

$$\mathbf{y} = \text{Poisson}\{\alpha \mathbf{H}\mathbf{x}\}/\alpha. \tag{14}$$

During training, we use the ℓ_1 loss between the reconstructed image $\widehat{\mathbf{x}}$ and the ground truth image $\mathbf{x}$ to train the networks. The loss function is defined by

$$\mathcal{L}(\widehat{\mathbf{x}}, \mathbf{x}) = \|\widehat{\mathbf{x}} - \mathbf{x}\|_1, \tag{15}$$

where $\|\cdot\|_1$ denotes the ℓ_1 norm. We train all the networks for 500 epochs, with the Adam optimizer. The learning rate is initialized as 10^{-4} which gets halved every 100 epochs. The batch size was set to 2 for all the methods. We do so to ensure a fair comparison because some methods consume more GPU power. We used a NVIDIA GeForce RTX 2080 Ti graphics card for both training and inference of all the methods. The inputs to the networks include the degraded image $\mathbf{y}$ and the blur kernel $\mathbf{h}$. Some methods like [18], [28] take the noise level as inputs. In such cases, the photon level α corresponding to each image was sent as the input.

For testing, we evaluate the methods using synthetically degraded images obtained by blurring 100 images from the BSD300 dataset [62]. We use 3 different sets of 5 motion kernels of size 9×9 (Small), 27×27 (Medium), 45×45 (Large) using [61]. Each combination of the image and motion is evaluated at three different photon levels (10, 30, and 50).

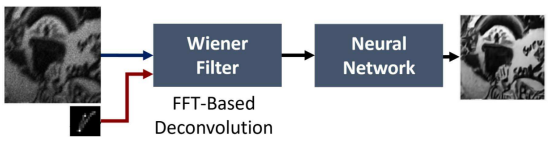
B. Secret 1: Using Wiener Filters is Recommended

We observe that the five methods discussed in Section II-C all have a separate Fourier-based deconvolution module - irrespective of whether they are traditional methods or deep learning-based methods. The presence of the Fourier-based deconvolution module hints that a black-box neural network might have some limitations.

TABLE VI
PERFORMANCE OF THE 5 METHODS OF INTEREST ON THE TEST DATASET

Kernel Size	ppp		PURE-LET [27]	VSTP [32]	DWDN [14]	PhDNet [28]	USRNet [18]	Proposed
Small	10	PSNR (dB)	22.64	24.93	25.13	25.20	25.26	**25.27**
		SSIM	0.672	0.733	0.771	0.775	0.775	**0.778**
		LPIPS	0.509	0.413	0.403	0.398	0.400	**0.397**
	30	PSNR (dB)	23.21	25.71	26.11	26.24	26.30	**26.32**
		SSIM	0.694	0.770	0.805	0.809	0.810	**0.813**
		LPIPS	0.463	0.370	0.358	0.356	0.356	**0.355**
	50	PSNR (dB)	23.57	26.58	26.17	26.74	26.79	**26.84**
		SSIM	0.702	0.779	0.820	0.824	0.826	**0.828**
		LPIPS	0.432	0.351	0.338	0.334	0.333	**0.331**
Medium	10	PSNR (dB)	20.91	22.93	23.06	23.24	23.25	**23.28**
		SSIM	0.621	0.649	0.700	0.704	0.705	**0.707**
		LPIPS	0.549	0.496	0.485	0.480	**0.479**	**0.479**
	30	PSNR (dB)	21.71	23.69	24.07	24.30	24.32	**24.36**
		SSIM	0.644	0.691	0.737	0.744	0.744	**0.747**
		LPIPS	0.511	0.450	0.440	0.435	0.437	**0.432**
	50	PSNR (dB)	22.14	24.08	24.57	24.86	24.88	**24.92**
		SSIM	0.661	0.708	0.755	0.763	0.764	**0.767**
		LPIPS	0.500	0.429	0.418	0.414	**0.413**	**0.413**
Large	10	PSNR (dB)	20.53	22.40	22.43	22.60	22.63	**22.65**
		SSIM	0.588	0.642	0.679	0.683	0.684	**0.685**
		LPIPS	0.598	0.533	0.520	0.515	0.515	**0.514**
	30	PSNR (dB)	21.22	23.18	23.41	23.65	23.69	**23.71**
		SSIM	0.613	0.649	0.714	0.721	0.722	**0.723**
		LPIPS	0.537	0.470	0.459	**0.453**	0.455	**0.453**
	50	PSNR (dB)	21.60	23.56	23.91	24.19	24.22	**24.25**
		SSIM	0.625	0.676	0.732	0.740	0.741	**0.742**
		LPIPS	0.512	0.452	0.441	0.439	**0.437**	**0.437**

PURE-LET [27], and VSTP [32] are traditional methods and do not use any neural networks. DWDN [14] is neural network-based but non-iterative. PhDNet [28] and USRNET [18] are unrolled neural network-based solutions. The best-performing method is shown in bold and the second best method is underlined.



(a) Neural network w/ Wiener Filter

(b) Neural network w/o Wiener Filter

Fig. 6. How Wiener filters are used. We consider two neural networks, where in (a) we decouple the inversion step by a Fourier-based deconvolution module which is the Wiener filter, and in (b) we use only a neural network. The added computational complexity of the Wiener filter is minimal because it is a simple inversion in the Fourier space.

The decoupling approach makes sense in classical methods. In these methods, Poisson deconvolution is often posed as MAP-based optimization. Since it is very difficult for a simple optimization step to simultaneously handle blur and noise, it makes sense to decouple them.

How about deep neural networks? One would expect that since they have a large capacity, they wouldn't need to adopt a decoupling strategy. To examine the need for decoupling, we compare the two configurations as shown in Fig. 6 - neural networks with and without a Wiener filter.

In this experiment, we use the ResUNet from [18] for the task shown in Fig. 6(b). We train the network at a particular light level of 10 photons per pixel (ppp). To ensure that there is no domain gap, we train the network for *one* single blur kernel and test it for the exact same blur kernel. For the configuration shown in Fig. 6(a), we use a single-iteration USRNet. A single-iteration USRNet is nothing but a deconvolution module followed by a refinement network. We train the network with a large range of photon levels and blur kernels, as described in Section III-A. Our argument is that if a specialized network in Fig. 6(b) cannot beat a generic network in Fig. 6(a), then there must be some fundamental limits in the network itself.

The results are shown in Fig. 7. We observe that the black-box neural network cannot handle blur and noise simultaneously. In contrast, a network with an explicit deconvolution step performs much better. Our conjecture is that since we know the blur kernel, it is better to incorporate this forward model in the solution when deblurring the image.

C. *Secret 2: Iterations are Recommended*

Iterative methods, regardless if they are traditional or neural-network-based, tend to perform better according to Table VI. Let us explain why this is the case.

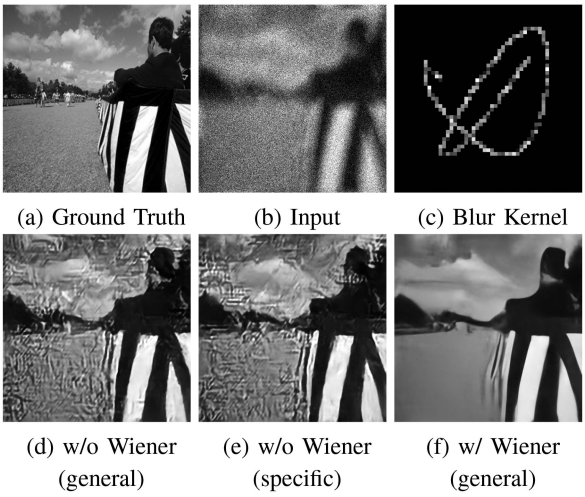


Fig. 7. Secret 1: Wiener filter is recommended (a) Clean. (b) Degraded image. (c) Blur kernel. (d) A deconvolution U-Net trained on a variety of kernels. (e) A deconvolution U-Net trained on the specific blur kernel defined in (b). Note that even if we train the network *specifically for the blur kernel*, the result is still not satisfactory. (f) Deconvolution when the Wiener step is included in the U-Net.

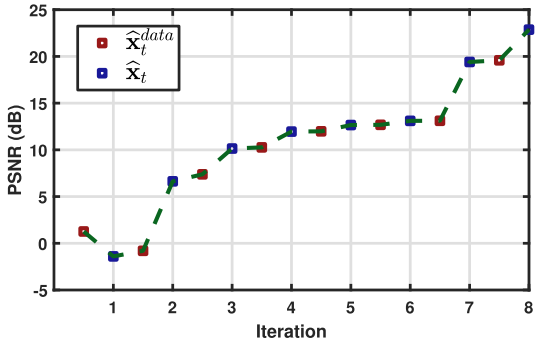


Fig. 8. Secret 2: Iteration is recommended. We plot the PSNR of the estimates at each iteration of USRNet. We can notice that ignoring the first iteration, the plot aligns with our claim - A better deconvolution leads to a better refinement, which turn again leads to a better deconvolution.

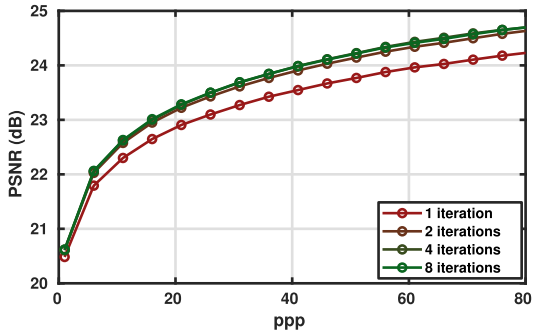


Fig. 9. Number of iterations. We retrain USRNet with different numbers of iterations. We can see that we obtain a performance boost by increasing the number of iterations.

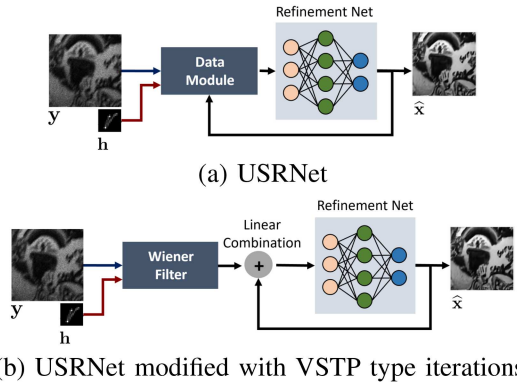


Fig. 10. What kind of iterations help? By comparing USRNet and VSTP, we observe two types of iterations. (a) USRNet sends $\widehat{\mathbf{x}}$ back to the data module iteratively to improve the estimate. (b) VSTP uses $\widehat{\mathbf{x}}$ to form a linear combination with the Wiener filter outputs. We find that iteration in (a) is more effective.

Consider USRNet as an example: It is an iterative algorithm where the iterations are given by (12) and (13). The first step (12) is the deconvolution module which produces an estimate $\widehat{\mathbf{x}}_t^{\text{data}}$, and the second step (13) is a neural network denoiser that refines the estimate to generate $\widehat{\mathbf{x}}_t$. The performance of a denoiser is directly related to the input quality. The noisier the input is, the worse the reconstruction performance will be [63]. In a single-shot low-light deconvolution, we need to have a very good estimate from (12), and this needs to be computed directly from the noisy image itself. In iterative schemes, even though the initial estimate $\widehat{\mathbf{x}}_0^{\text{data}}$ is not good, the mild refinement steps will gradually improve the image quality because they use both the previous estimate $\widehat{\mathbf{x}}_{t-1}^{\text{data}}$ and the current estimate.

Fig. 8 shows a typical per-iteration PSNR of an iterative scheme USRNet. Putting aside the initial estimate (which shows a downward PSNR trend), the performance generally goes up as the number of iterations increases. Specifically, we see that after each pair of $\widehat{\mathbf{x}}_t$ and $\widehat{\mathbf{x}}_t^{\text{data}}$, the performance improves. The exact dynamics of the PSNR is difficult to track because it is image-dependent. However, the trend confirms our hypothesis that iterations are helpful.

For unrolled algorithms, the number of iterations is realized by the number of blocks. A natural question is the number of such iterative blocks — will more blocks improves the overall deconvolution result? Fig. 9 shows the results of four USRNets trained at different number of iterative blocks. It is clear from the result that more iterations leads to a better final performance, although there is a diminishing return after several blocks.

Another question we ask is the *type* of iterations. Among the methods reported in Table VI, there are two different kinds of iterative schemes as shown in Fig. 10. The first one is the USRNet where the estimate $\widehat{\mathbf{x}}$ is fed back to the data module (i.e., the inversion module), and is combined with the raw input $\mathbf{y}$ and kernel $\mathbf{h}$ to construct a new intermediate estimate. The second iterative scheme is the one used in VSTP. In this scheme, the estimate $\widehat{\mathbf{x}}$ is used to form a linear combination with the output of the Wiener filter.

To evaluate the performance of the two schemes, we modify USRNet to incorporate the VSTP mechanism. We argue that this is a fairer comparison than directly using VSTP because VSTP uses a traditional denoiser BM3D. In our modification, we ensure that the two networks are trained with the same type and same amount of data. The results are shown in Fig. 11,

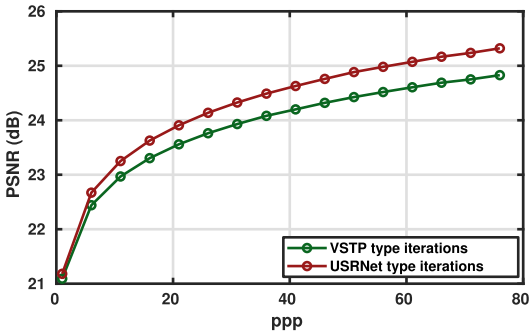


Fig. 11. What kind of iterations help? This figure is a follow-up of Fig. 10 where here we plot the PSNR as a function of the photon level.

where we see that the iterative scheme by USRNet has a clear advantage over the VSTP scheme.

Based on the above experiments, our recommendation regarding the iterative scheme is that iterative schemes offer better performance than one-shoot methods. Among the different iterative schemes, we recommend feeding back the estimate $\widehat{\mathbf{x}}$ directly to the inversion module so that the features of $\widehat{\mathbf{x}}$ will be utilized better.

D. Secret 3: Feature Space is Recommended

The next secret about Poisson deconvolution is that it is better to deconvolve the image in the *feature space* instead of the image space. This observation is based on the difference between PURE-LET and DWDN in Table VI. Both PURE-LET and DWDN use multiple Wiener filters. PURE-LET uses different deblurring strengths (as specified by the hyper-parameter λ), whereas DWDN uses the same Wiener filter for different feature maps. But the biggest difference is that PURE-LET performs the deconvolution in the image space whereas DWDN performs the deconvolution in the feature space. We show in this subsection that the superior performance of DWDN is partially driven by feature space deconvolution.

To prove the usefulness of feature space deconvolution instead of image space, we consider the following four modifications of DWDN by placing the Wiener filters in different ways.

1) **Configuration I** in Fig. 12(a) uses a single Wiener filter followed by a refinement network. This is the vanilla network for baseline analysis.
2) **Configuration II** in Fig. 12(b) uses three Wiener filters as in PURE-LET. Each Wiener filter uses a different regularization parameter λ. We use a deep neural network as the refinement step so that it is a fair comparison with DWDN.
3) **Configuration III** in Fig. 12(c) uses a feature extraction unit to pull the features before sending them to Wiener filters. This is the same as DWDN. In our experiment, there are 16 feature maps. The regularization parameter λ is the same across the 16 Wiener filters.
4) **Configuration IV** in Fig. 12(d) uses 16 Wiener filters where each has three sub-configurations. Each sub-configuration uses a different λ. We regard Configuration IV as the ultimate modification we can make within the context of our analysis.

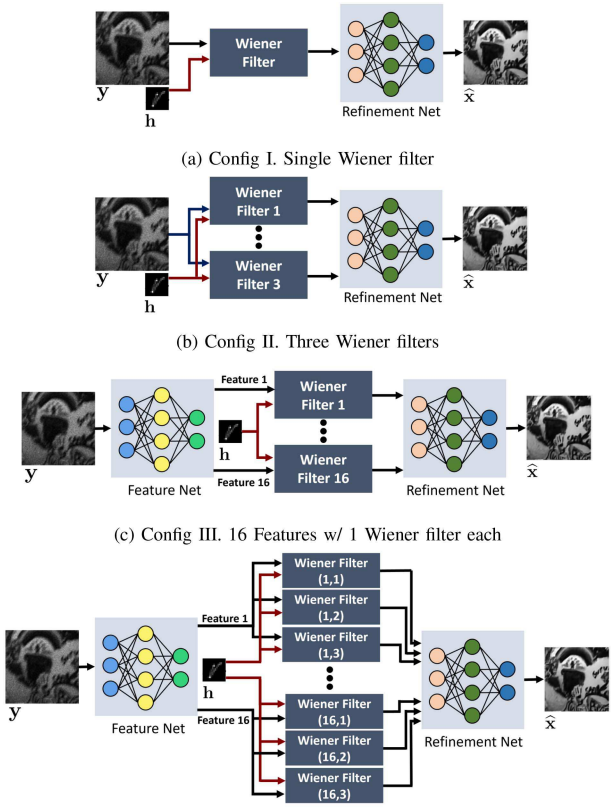


Fig. 12. Secret 3: Feature space is recommended. The two types of deconvolutions: image space and feature space. (a)–(b) perform deconvolution in the spatial domain, whereas (c)–(d) perform deconvolution in the feature space.

The comparisons between these configurations are shown in Table II. We see that across the different photon levels, the ones that perform deconvolution in the feature space are *significantly* better. Our intuitive argument is that in the feature space, the signals are already *decomposed*. If the feature extraction unit is powerful, signals will be captured in a few leading feature dimensions whereas noise will be concentrated in the other dimensions. Therefore, the strong signal features will be deconvolved well by the Wiener filter with a smaller λ, whereas the noise features will be attenuated by a large λ. As a result, the overall deconvolution will be better. As for how much regularization λ is needed, Configuration III and IV tell us that the benefit is marginal.

Based on these findings, our recommendation here is that whenever possible, deconvolution should be performed in the feature space. Using different regularization parameter λ does not seem to have a significant difference.

E. Secret 4: Poisson Likelihood is Not Needed

By virtue of Poisson deconvolution, the likelihood function should be Poisson. However, several observations make us believe that the Poisson likelihood is not needed in a neural network based solution.

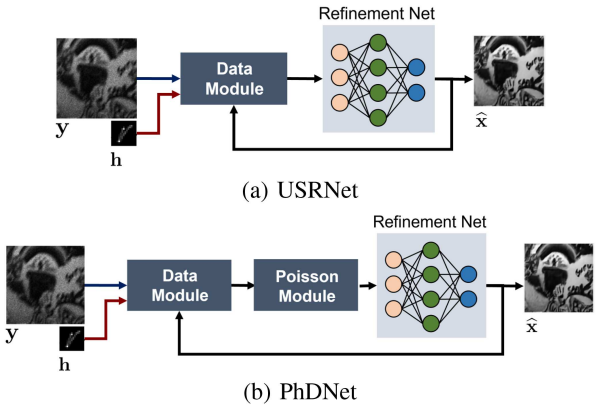


Fig. 13. Secret 4: Poisson likelihood is not needed. USRNet and PhDNet are both iterative unrolled networks. The difference is that in PhDNet, an explicit Poisson module is used to handle the Poisson noise.

Our first observation is the comparison between USRNet and PhDNet in Table VI. USRNet uses a Gaussian likelihood whereas PhDNet uses a Poisson likelihood. Because of the Poisson likelihood, PhDNet needs to introduce a variable splitting technique to specifically handle the Poisson part, see the added Poisson module illustrated in Fig. 13. However, from Table III we observe that the difference in performance between the two methods is negligible.

Readers may argue that the vanishing performance gap is due to the iterations, i.e., as the number of iterations increases, the network capacity increases and hence they are more capable of handling the Poisson statistics. To prove that this is not the case, we train four versions of USRNet and PhDNet with a fixed number of 1, 2, 4, and 8 iterative blocks in their unrolled networks. We can see from Table III that irrespective of the number of iterations used by the method, USRNet performs as well as PhDNet. Therefore, whether or not we use an explicit Poisson module does not matter.

Another "indirect" observation is about the design of PURE-LET. In PURE-LET, the Poisson statistics is used to estimate the PURE score which is an unbiased risk estimator of the mean squared error. However, the actual deconvolution step is performed by a bank of Wiener filters - which is derived from Gaussian statistics.

If Poisson modules are not needed, we expect that techniques associated with the Poisson likelihood would not have any significance to the restoration problem. This observation is supported by inspecting methods using the variance stabilizing transform (VST). Fig. 14 shows a typical VST-based image denoising algorithm. In the VST case, we first apply VST to stabilize the Poisson variance. We then denoise the image, and transform back via the inverse VST. In our experiment, we use the ResUNet from [18] as the denoiser.

Table IV shows the performance between using VST or not. We oberve that using VST does not offer the denoiser any advantage. The network without VST even marginally outperforms the denoiser with VST. This finding is consistent with what was reported in [20] for binomial noise.

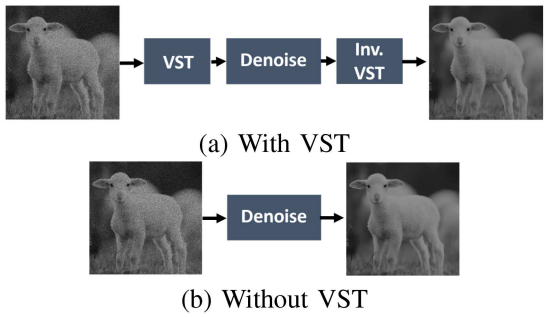


Fig. 14. Variance Stabilizing Transform. VSTs are often used in Poisson noise. (a) A denoising method using VST. (b) A denoising method without VST.

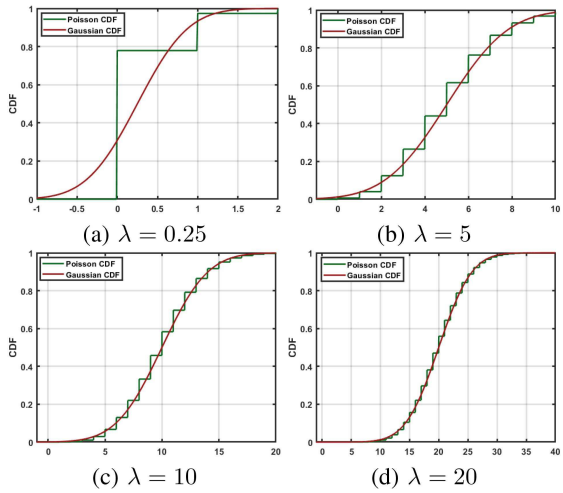


Fig. 15. Secret 4: As long as λ is not too small, the Poisson can be well approximated by the Gaussian. This combined with the expressivity of neural networks make Poisson likelihood unnecessary for network design.

An intuitive explanation of why Poisson likelihood might not be needed is as follows. In the range of photon levels we are considering ($\alpha \in [5, 80]$), the Poisson distribution can be approximated by a Gaussian distribution [64, Lemma 3]

$$\frac{\lambda^k e^{-\lambda}}{k!} \approx \frac{1}{\sqrt{2\pi\lambda}} e^{-\frac{(k-\lambda)^2}{2\lambda}}, \lambda \gg 1. \tag{16}$$

In Fig. 15, we plot the cumulative distribution function (CDF) of Gaussian and Poisson. We see that when λ is not too small, the Poisson can be well approximated by the Gaussian although its variance is signal-dependent. This Gaussian approximation works sufficiently well for Poisson noise with $\lambda \gg 1$. For the signal-dependent noise variance, our results in the variance stabilizing transform show that the neural networks have enough expressiveness to learn the spatially varying noise variance. If the photon level is extremely low, e.g., in some very low-light microscope applications, then we might need to use the full Poisson likelihood.

Based on the above analysis, our conclusion is that when handling the Poisson noise in low-light, network architectures designed for Gaussian likelihood will work just as well. There is no clear advantage of using the more complicated Poisson likelihood and/or variance stabilizing transforms. As long as the

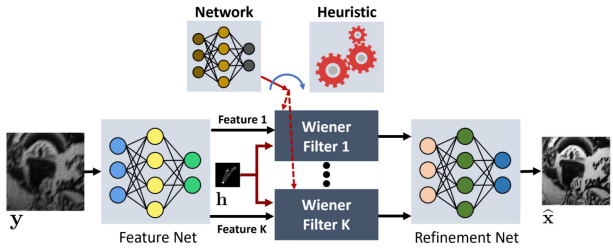


Fig. 16. Secret 5: Hyper-parameter learning is recommended. We can use heuristics or train a network to select the hyper-parameters.

photon level is not extreme, the explicit Poisson modules are not required.

F. Secret 5: Learning Hyperparameters is Recommended

Among the learning-based methods, USRNet and PhDNet use networks to learn the hyperparameters that get used in the data module, Poisson module and the refinement net. However, DWDN uses a heuristic method for estimating this parameter. To understand if it is important to learn the hyperparameters, we modify DWDN and learn the hyperparameter that is being input to the Wiener filters using the same network structure used by PhDNet to learn its parameters. Fig. 16 illustrates the conceptual difference between the two.

The result of this experiment can be found in Table V. We notice that when DWDN is augmented with a small network for learning the hyperparameters, it performs slightly better than using a heuristic for finding the parameters. The improvement is less than 0.1 dB which is not very noticeable. However, since the computational cost of adding a hyper-parameter learning module is so small compared to the whole network, it does not hurt to include it.

Based on the above experiments, we conclude that hyper-parameter learning is helpful but it is not necessary. We still recommend it because it saves us from hand-tuning the hyper-parameters.

IV. Combining the Secrets

After presenting the five secrets, a natural question is: "what if we combine these ideas?" To this end, we create a method called the Five-in-One Network (FIO-Net) as shown in Fig. 17. We make two remarks before we discuss this network: 1. We do not regard FIO-Net as a novel invention or claim it to be a state-of-the-art. We view FIO-Net a check point of the five secrets. We are more interested in checking whether its performance is consistent with the five secrets, rather than expecting it to beat other methods by a big margin. 2. Although FIO-Net is a combination of the five secrets, it would still require some design because otherwise there is no guarantee it should work. We will present a way to integrate these five ideas.

To elaborate on the design principle of the FIO-Net, we first use Secret 4 to replace Poisson likelihood with the Gaussian likelihood. This implies that as far as the network structure is concerned, we can focus on the Gaussian forward model:

$$\mathbf{y} = \alpha\mathbf{H}\mathbf{x} + \mathbf{n}, \tag{17}$$

where α defines the photon level. We remark that this is *not* the original Poisson deconvolution problem that we want to solve. However, since Secret 4 tells us that utilizing the Poisson likelihood is not needed, we consider the Gaussian model when designing the neural network. When training the model, we take blurred images and add synthetic Poisson noise instead of Gaussian noise.

Remark: The concept using a sub-optimal forward model in exchange of better reconstruction performance is perhaps counter-intuitive. The general line of argument is known as computational image formation which we refer readers to [65] for detailed elaborations.

Our next step is to use Secret 3, which suggests us to perform the deconvolution in the feature space. To this end, we consider a set of linear filters $\{\mathbf{F}_i \mid i = 1, 2, \ldots, M\}$ and apply them to

$$\mathbf{F}_i\mathbf{y} = \alpha\mathbf{F}_i\mathbf{H}\mathbf{x} + \mathbf{F}_i\mathbf{n}. \tag{18}$$

Since $\mathbf{F}_i$ and $\mathbf{H}$ represent convolutional operations in matrix form, we can switch the order using the commutative property of convolution to obtain

$$\mathbf{F}_i\mathbf{y} = \alpha\mathbf{H}\mathbf{F}_i\mathbf{x} + \mathbf{F}_i\mathbf{n}. \tag{19}$$

The question now becomes how to recover $\mathbf{x}$.

Solving (19) would require an optimization. In FIO-Net, we consider a generic regularized least squares:

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{M} \|\mathbf{F}_i\mathbf{y} - \alpha\mathbf{H}\mathbf{F}_i\mathbf{x}\|^2 + \lambda g(\mathbf{x}). \tag{20}$$

where $g(\mathbf{x})$ is the prior. Since an unconstrained optimization problem with a sum of two different functions is difficult to optimize, we split the original problem into two simpler sub-problems. We introduce a set of new variables $\{\mathbf{z}_i = \mathbf{F}_i\mathbf{x}, i = 1, 2, \ldots M\}$, and collectively define $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$. The new constrained optimization problem now becomes

$$\begin{aligned} \{\widehat{\mathbf{x}}, \widehat{\mathbf{z}}\} = \underset{\mathbf{x},\mathbf{z}}{\operatorname{argmin}} & \sum_{i=1}^{M} \left\{\|\mathbf{F}_i\mathbf{y} - \alpha\mathbf{H}\mathbf{z}_i\|^2 + \lambda g(\mathbf{x})\right\} \\ \text{subject to} \quad & \mathbf{z}_i = \mathbf{F}_i\mathbf{x}, \ i = 1, 2, \ldots, M. \end{aligned} \tag{21}$$

Equation (21) is a standard optimization that can be solved using the half-quadratic splitting (HQS) [18]. HQS formulates an alternative optimization:

$$\begin{aligned} \{\widehat{\mathbf{x}}, \widehat{\mathbf{z}}\} = \underset{\mathbf{x},\mathbf{z}}{\operatorname{argmin}} \sum_{i=1}^{M} \Big\{ & \|\mathbf{F}_i\mathbf{y} - \alpha\mathbf{H}\mathbf{z}_i\|^2 + \quad \lambda g(\mathbf{x}) \\ & + \mu_i\|\mathbf{F}_i\mathbf{x} - \mathbf{z}_i\|^2 \Big\}, \end{aligned} \tag{22}$$

where μ_i is the penalty strength.

In what follows, we briefly summarize the equations to solve (22). During the discussion, we will explain how the secrets are used. The algorithm to solve (22) involve two steps:

$$\mathbf{z}_i^k = \underset{\mathbf{z}_i}{\operatorname{argmin}} \|\mathbf{F}_i\mathbf{y} - \mathbf{H}\mathbf{z}_i\|^2 + \mu_i^k\|\mathbf{F}_i\mathbf{x}^{k-1} - \mathbf{z}_i\|^2 \tag{23}$$

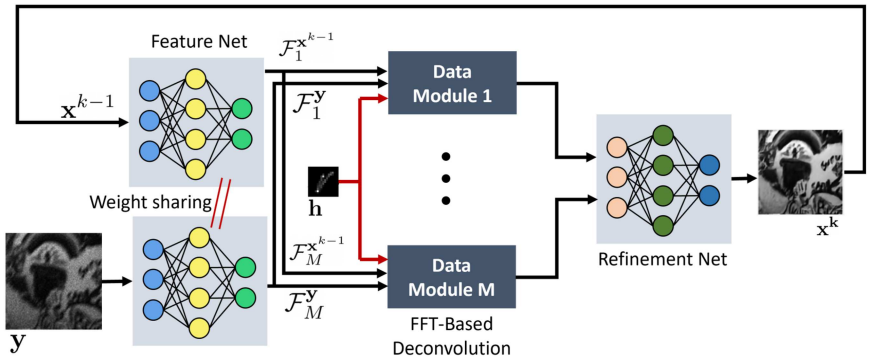


Fig. 17. Schematic diagram of FIO-Net. The Five-in-One Network (FIO-Net) utilizes all the five secrets we observed in the previous section. It is an iterative scheme performing deconvolution in the feature space. It uses Wiener filter, but no Poisson likelihood. Hyperparameters are automatically tuned.

$$\mathbf{x}^k = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{M} \mu_i^k \|\mathbf{F}_i\mathbf{x} - \mathbf{z}_i^k\|^2 + \lambda g(\mathbf{x}), \tag{24}$$

where we use the fact that the optimization of $\mathbf{z}_i$ in (22) is separable so that we can solve for individual $\mathbf{z}_i$'s.

Next, we apply Secret 5 which says that we should learn the hyperparameters end-to-end. Thus, we replace the penalty μ_i with μ_i^k so that they change over iterations. Similar to [28], we use a small fully connected neural network for estimating the hyperparamaters μ_i^k with the kernel $\mathbf{H}$ and the photon level α used as the input.

Let's solve (23) and (24). (23) is a least squares minimization problem, and it has a closed form expression given by

$$\mathbf{z}_i^k = \left(\mathbb{I} + \mu_i^k \mathbf{H}^T\mathbf{H}\right)^{-1}\left(\mathbf{F}_i\mathbf{x}^{k-1} + \mu_i^k \mathbf{H}^T\mathbf{F}_i\mathbf{y}\right). \tag{25}$$

Assuming that the convolution operation represented by $\mathbf{H}$ is carried out with circular boundary conditions, (25) has a FFT based solution given by

$$\mathbf{z}_i^k = \mathcal{F}^{-1}\left[\frac{\mathcal{F}(\mathbf{F}_i\mathbf{x}^{k-1}) + \mu_i^k\overline{\mathcal{F}(\mathbf{H})}\cdot\mathcal{F}(\mathbf{F}_i\mathbf{y})}{1+\mu_i^k|\mathcal{F}(\mathbf{H})|^2}\right], \tag{26}$$

where $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ denote the FFT and inverse FFT respectively, and $\overline{(\cdot)}$ denotes the complex conjugate function. Following the idea of [14], we replace the linear filters with learnable non-linear convolutional neural network $\mathcal{D}_{\text{feat}}(\cdot)$. Similar to [14], we note that while the solution (26) was obtained for linear filters, using non-linear neural networks works well and even better than linear filters. Therefore, (26) is modified as

$$\mathbf{z}_i^k = \mathcal{F}^{-1}\left[\frac{\mathcal{F}(\mathcal{D}_i^{\text{feat}}(\mathbf{x}_{k-1})) + \mu_i^k\overline{\mathcal{F}(\mathbf{H})}\cdot\mathcal{F}(\mathcal{D}_i^{\text{feat}}(\mathbf{y}))}{1+\mu_i^k|\mathcal{F}(\mathbf{H})|^2}\right], \tag{27}$$

where $\{\mathcal{D}_1^{\text{feat}}(\cdot),\ldots,\mathcal{D}_M^{\text{feat}}(\cdot)\} = \mathcal{D}^{\text{feat}}(\cdot)$ represents the features generated by the neural network.

The other subproblem in (24), in the absence of the filters $\mathbf{F}_i$, can be thought of as a image denoising problem [66]. However, the presence of the filters makes this problem not so straightforward. However, we can still think of (24) as a restoration task where we want to recover the image $\mathbf{x}$ from a set of features $\mathbf{z}_i$. We want to minimize the residue between the input features and the features generated from $\mathbf{x}$, while enforcing the prior $g(\mathbf{x})$. Given the complex nature of restoring the image from a set of features, and the difficulty of defining a good prior term $g(\mathbf{x})$, we propose to solve this problem using a convolutional neural network as

$$\mathbf{x}^k = \mathcal{D}_{\text{refine}}\left(\mathbf{z}_1^k,\ldots,\mathbf{z}_M^k,\frac{\lambda}{\mu^k}\right), \tag{28}$$

where we have assumed that the penalties $\mu_i^k = \mu^k$ do not vary over the features. The entire algorithm is summarized in Algorithm 1.

Algorithm 1: *FIO-Net:* Fixed Iteration Unrolling.

Input: Degraded Image $\mathbf{y}$, Kernel $\mathbf{H}$, Photon level α
$\mathbf{x}^0 \leftarrow \mathbf{y}$
$\{\mathcal{F}_i^{\mathbf{y}}\}_{i=1,\ldots,M} = \mathcal{D}^{\text{feat}}(\mathbf{y})$ ▷ Feature extraction from y
$\{\mu_k\}_{k=1,\ldots,K} = \mathcal{D}^{\text{hyp}}(\mathbf{H},\alpha)$ ▷Hyperparameters from $\mathcal{D}^{\text{hyp}}(\cdot)$
for $k = 1, 2, \cdots, K$ **do**
  Update $\mathbf{z}_i^k$ using Equation (26)
  Update $\mathbf{x}^k$ using Equation (28)
end for
return $\mathbf{x}^K$

The method is iterative, based on unrolled optimization that uses the convolutional neural networks only for image refinement and a traditional FFT based method is used for deconvolution. The iterative scheme described in Algorithm 1 is unrolled for K = 8 iterations and then trained end-to-end using the same training process as that described in Section III-A. The training of FIO-Net took roughly 24 hours on a GeForce RTX 2080 Ti graphics card. Both Feature Net and Refinement Net is trained during this. The method incorporates the idea of deconvolving in the feature space, and does not have any specific Poisson design.

V. Experiments

After elaborating on the proposed method, we present the quantitative results on BSD300 dataset in Table VI. We use the same testing process as described in Section III-A so that the testing conditions are fair to all methods. We make three comments:

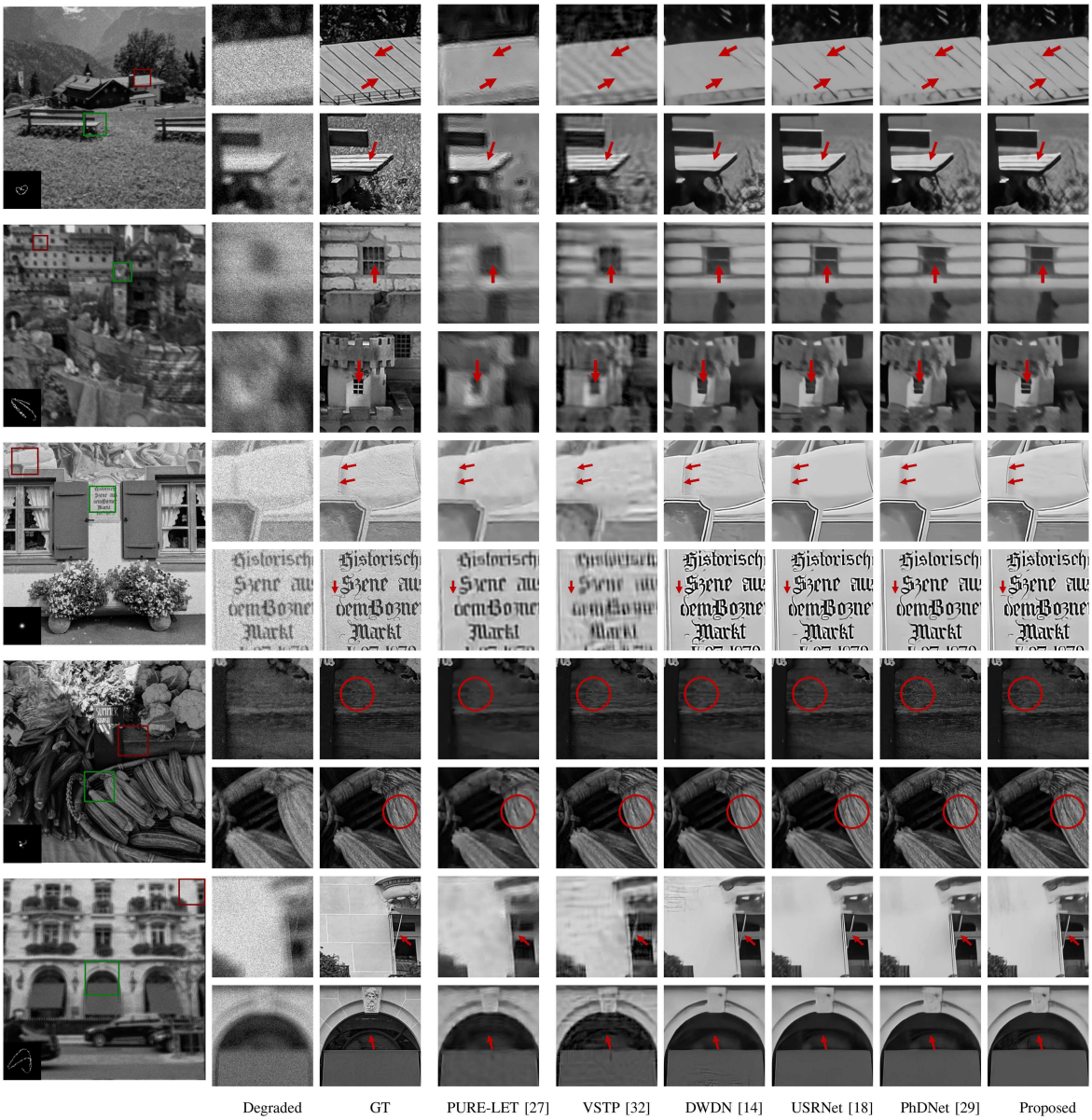


Fig. 18. Experimental results comparing the proposed FIO-Net with several other Poisson deconvolution methods.

- Compared to classical methods such as PURE-LET and VSTP, FIO-Net outperforms by a big margin. This should not be a surprise, because all deep learning methods outperform these two classical methods.
- Compared to a single-pass deep learning method DWDN, the performance of FIO-Net is substantially better, especially for bigger blur kernels. This stresses the importance of iterative methods.
- Compared to PhD-Net and USR-Net, the performance of FIO-Net is marginal. This is caused by the fact that some of the attributes have overlapping influences, e.g., feature space and iteration. While Secret 3 says that feature space deconvolution could help single-iteration methods, its impact may be diminished when more iterations are used. This does not make Secret 3 redundant. For example, when operating with a strict computational budget we may choose to have fewer iterations and in such a scenario Secret 3 will become more important.

In Fig. 18 we show the visual comparisons. The visual comparisons apparently show another perspective of FIO-Net. If we compare USRNet, PhDNet, and FIO-Net, we see that all three perform similarly. However, as we zoom in to see the details, e.g., the lines on the roof in the first image, the bars on the windows in the second image, and the tail of the alphabet in the third image, we can see the visual improvement of FIO-Net. We remark that all models are trained using the exact same training dataset and tested on the same testing dataset. Therefore, the restored details are due to the network itself rather than data overfitting.

VI. Conclusion

With the growth of photon-limited imaging applications, we recognize the importance of understanding the performance limits of Poisson deconvolution algorithms. To this end, we present a systematic analysis of a large number of existing non-blind Poisson deconvolution methods. Based on this analysis, we

deduce five "secrets" that are needed for an effective non-blind Poisson deconvolution algorithm design:

1) Use Wiener filter for spatially invariant blur
2) Use iterative neural networks instead of single forward-pass neural networks
3) Use feature space deblurring instead of image space deblurring
4) No need to use Poisson likelihood in the network architecture design
5) Learn hyperparameters for iterative algorithms in an end-to-end manner.

By combining these five secrets, we obtain a proof-of-concept named the Five-In-One Network (FIO-Net). The results offered by FIO-Net are consistent with the five secrets we presented. Considering that FIO-Net is not a novel design but a combination of five existing ideas, the consistency and the on-par performance with the state-of-the-art result provide additional support to our findings.

ACKNOWLEDGMENT

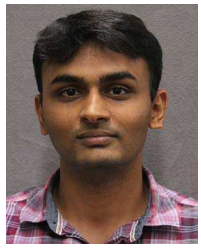
The authors would like to thank Nick Chimitt and Yiheng Chi of Purdue School of ECE for their careful reading of the paper and discussions of the ideas.

REFERENCES

[1] H. C. Andrews and B. R. Hunt, *Digital Image Restoration*. Hoboken, NJ, USA: Prentice-Hall, 1977.
[2] C. R. Vogel and M. E. Oman, "Fast, robust total variation-based reconstruction of noisy, blurred images," *IEEE Trans. Image Process.*, vol. 7, no. 6, pp. 813–824, Jun. 1998.
[3] S. H. Chan and T. Q. Nguyen, "Single image spatially variant out-of-focus blur removal," in *Proc. IEEE Int. Conf. Image Process.*, 2011, pp. 677–680.
[4] N. Dey et al., "Richardson-lucy algorithm with total variation regularization for 3D confocal microscope deconvolution," *Microsc Res Tech.*, vol. 69, pp. 260–266, Apr. 2006.
[5] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *IEEE Trans. Image Process.*, vol. 19, no. 9, pp. 2345–2356, Sep. 2010.
[6] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman, "Removing camera shake from a single photograph," *ACM Trans. Graph.*, vol. 25, pp. 787–794, Jul. 2006.
[7] Q. Shan, J. Jia, and A. Agarwala, "High-quality motion deblurring from a single image," in *Proc. ACM SIGGRAPH Conf.*, 2008, pp. 73.1–73.10.
[8] S. Cho, J. Wang, and S. Lee, "Handling outliers in non-blind image deconvolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 495–502.
[9] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 479–486.
[10] A. Danielyan, V. Katkovnik, and K. Egiazarian, "BM3D frames and variational image deblurring," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1715–1728, Apr. 2012.
[11] L. Xu and J. Jia, "Two-phase kernel estimation for robust motion deblurring," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 157–170.
[12] J. Dong, S. Roth, and B. Schiele, "Learning spatially-variant map models for non-blind image deblurring," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4886–4895.
[13] J. Dong, S. Roth, and B. Schiele, "Deep wiener deconvolution: Wiener meets deep learning for image deblurring," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1048–1059.
[14] J. Dong, S. Roth, and B. Schiele, "DWDN: Deep wiener deconvolution network for non-blind image deblurring," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 9960–9976, Dec. 2022.
[15] D. Gong, Z. Zhang, Q. Shi, A. van den Hengel, C. Shen, and Y. Zhang, "Learning deep gradient descent optimization for image deconvolution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5468–5482, Dec. 2020.
[16] Y. Nan and H. Ji, "Deep learning for handling kernel/model uncertainty in image deconvolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2388–2397.
[17] V. Pronina, F. Kokkinos, D. V. Dylov, and S. Lefkimmiatis, "Microscopy image restoration with deep wiener-kolmogorov filters," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 185–201.
[18] K. Zhang, L. V. Gool, and R. Timofte, "Deep unfolding network for image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3217–3226.
[19] S. H. Chan, O. A. Elgendy, and X. Wang, "Images from bits: Non-iterative image reconstruction for quanta image sensors," *MDPI Sensors*, vol. 16, no. 11, 2016, Art. no. 1961.
[20] J. H. Choi, O. A. Elgendy, and S. H. Chan, "Image reconstruction for quanta image sensors using deep neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2018, pp. 6543–6547.
[21] O. A. Elgendy, A. Gnanasambandam, S. H. Chan, and J. Ma, "Low-light demosaicking and denoising for small pixels using learned frequency selection," *IEEE Trans. Comput. Imag.*, vol. 7, pp. 137–150, 2021.
[22] A. Gnanasambandam and S. H. Chan, "Image classification in the dark using quanta image sensors," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 484–501.
[23] A. Gnanasambandam and S. H. Chan, "HDR imaging with quanta image sensors: Theoretical limits and optimal reconstruction," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 1571–1585, 2020.
[24] 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 *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops*, 2021, pp. 3959–3970.
[25] Y. Chi, A. Gnanasambandam, V. Koltun, and S. H. Chan, "Dynamic low-light imaging with quanta image sensors," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 122–138.
[26] D. Snyder, *Random Point Processes*. Hoboken, NJ, USA: Wiley, 1975.
[27] J. Li, F. Luisier, and T. Blu, "PURE-LET image deconvolution," *IEEE Trans. Image Process.*, vol. 27, no. 1, pp. 92–105, Jan. 2018.
[28] Y. Sanghvi, A. Gnanasambandam, and S. H. Chan, "Photon limited non-blind deblurring using algorithm unrolling," *IEEE Trans. Comput. Imag.*, vol. 8, pp. 851–864, 2022.
[29] Y. Sanghvi, A. Gnanasambandam, Z. Mao, and S. H. Chan, "Photon-limited blind deconvolution using unsupervised iterative kernel estimation," *IEEE Trans. Comput. Imag.*, vol. 8, pp. 1051–1062, 2022.
[30] Y. Sanghvi, Z. Mao, and S. H. Chan, "Structured kernel estimation for photon-limited deconvolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 9863–9872.
[31] A. Rond, R. Giryes, and M. Elad, "Poisson inverse problems by the plug-and-play scheme," *J. Vis. Commun. Image Representation*, vol. 41, pp. 96–108, 2016.
[32] L. Azzari and A. Foi, "Variance stabilization in poisson image deblurring," in *Proc. IEEE Int. Symp. Biomed. Imag.*, 2017, pp. 728–731.
[33] S. H. Chan, R. Khoshabeh, K. B. Gibson, P. E. Gill, and T. Q. Nguyen, "An augmented lagrangian method for total variation video restoration," *IEEE Trans. Image Process.*, vol. 20, no. 11, pp. 3097–3111, Nov. 2011.
[34] T. Eboli, J. Sun, and J. Ponce, "End-to-end interpretable learning of non-blind image deblurring," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 314–331.
[35] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3929–3938.
[36] J.-F. Cai and Z. Shen, "Framelet based deconvolution," *J. Comput. Math.*, vol. 28, pp. 289–308, 2010.
[37] W. Cheng and K. Hirakawa, "Minimum risk wavelet shrinkage operator for poisson image denoising," *IEEE Trans. Image Process.*, vol. 24, no. 5, pp. 1660–1671, May 2015.
[38] Y. Zhang and K. Hirakawa, "Blind deblurring and denoising of images corrupted by unidirectional object motion blur and sensor noise," *IEEE Trans. Image Process.*, vol. 25, no. 9, pp. 4129–4144, Sep. 2016.
[39] M. Rahman Chowdhury, J. Zhang, J. Qin, and Y. Lou, "Poisson image denoising based on fractional-order total variation," *Inverse Problems Imag.*, vol. 14, no. 1, pp. 77–96, 2020.
[40] M. R. Chowdhury, J. Qin, and Y. Lou, "Non-blind and blind deconvolution under poisson noise using fractional-order total variation," *J. Math. Imag. Vis.*, vol. 62, pp. 1238–1255, Nov. 2020.
[41] D. J. Lingenfelter, J. A. Fessler, and Z. He, "Sparsity regularization for image reconstruction with poisson data," *Proc. SPIE*, vol. 7246, pp. 96–105, 2009.

[42] S. Lefkimmiatis and M. Unser, "Poisson image reconstruction with Hessian Schatten-norm regularization," *IEEE Trans. Image Process.*, vol. 22, no. 11, pp. 4314–4327, Nov. 2013.
[43] S. Setzer, G. Steidl, and T. Teuber, "Deblurring poissonian images by split bregman techniques," *J. Vis. Commun. Image Representation*, vol. 21, pp. 193–199, Apr. 2010.
[44] Z. T. Harmany, R. F. Marcia, and R. M. Willett, "This is SPIRAL-TAP: Sparse Poisson intensity reconstruction algorithms–theory and practice," *IEEE Trans. Image Process.*, vol. 21, no. 3, pp. 1084–1096, Mar. 2012.
[45] P. Getreuer, "Total variation deconvolution using split bregman," *Image Process. Online*, vol. 2, pp. 158–174, 2012.
[46] M. A. Figueiredo and J. M. Bioucas-Dias, "Restoration of poissonian images using alternating direction optimization," *IEEE Trans. Image Process.*, vol. 19, no. 12, pp. 3133–3145, Dec. 2010.
[47] L. Sun, S. Cho, J. Wang, and J. Hays, "Good image priors for non-blind deconvolution," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 231–246.
[48] D. Krishnan and R. Fergus, "Fast image deconvolution using hyper-laplacian priors," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009.
[49] N. Joshi, C. L. Zitnick, R. Szeliski, and D. J. Kriegman, "Image deblurring and denoising using color priors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 1550–1557.
[50] L. Yuan, J. Sun, L. Quan, and H.-Y. Shum, "Progressive inter-scale and intra-scale non-blind image deconvolution," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–10, 2008.
[51] U. Schmidt, C. Rother, S. Nowozin, J. Jancsary, and S. Roth, "Discriminative non-blind deblurring," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 604–611.
[52] L. Bar, N. Kiryati, and N. Sochen, "Image deblurring in the presence of impulsive noise," *Int. J. Comput. Vis.*, vol. 70, no. 3, pp. 279–298, 2006.
[53] M. Bertero, P. Boccacci, and C. D. Mol, *Introduction to Inverse Problems in Imaging*. Boca Raton, FL, USA: CRC Press, 2021.
[54] P. J. Green, "Bayesian reconstructions from emission tomography data using a modified em algorithm," *IEEE Trans. Med. Imag.*, vol. 9, no. 1, pp. 84–93, Mar. 1990.
[55] H. Son and S. Lee, "Fast non-blind deconvolution via regularized residual networks with long/short skip-connections," in *Proc. IEEE Int. Conf. Comput. Photography*, 2017, pp. 1–10.
[56] Z. Zhang, Y. Cheng, J. Suo, L. Bian, and Q. Dai, "INFWIDE: Image and feature space wiener deconvolution network for non-blind image deblurring in low-light conditions," *IEEE Trans. Image Process.*, vol. 32, pp. 1390–1402, 2023.
[57] Y. Nan, Y. Quan, and H. Ji, "Variational-em-based deep learning for noise-blind image deblurring," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3626–3635.
[58] Y. Fang, H. Zhang, H. S. Wong, and T. Zeng, "A robust non-blind deblurring method using deep denoiser prior," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 735–744.
[59] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, Mar. 2021.
[60] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 1132–1140.
[61] G. Boracchi and A. Foi, "Modeling the performance of image restoration from motion blur," *IEEE Trans. Image Process.*, vol. 21, no. 8, pp. 3502–3517, Aug. 2012.
[62] 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 *Proc. IEEE Int. Conf. Comput. Vis.*, 2001, pp. 416–423.
[63] A. Gnanasambandam and S. Chan, "One size fits all: Can we train one denoiser for all noise levels?," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3576–3586.
[64] A. Gnanasambandam and S. H. Chan, "Exposure-referred signal-to-noise ratio," *IEEE Trans. Comput. Imag.*, vol. 8, pp. 561–575, 2022.
[65] S. H. Chan, "Computational image formation," 2023. [Online]. Available: https://arxiv.org/abs/2307.11635
[66] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play ADMM for image restoration: Fixed-point convergence and applications," *IEEE Trans. Comput. Imag.*, vol. 3, no. 1, pp. 84–98, Mar. 2017.

Abhiram Gnanasambandam (Member, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology Madras, Chennai, India, in 2017, and the Ph.D. degree from the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, in 2022. His Ph.D. dissertation dealt with computational imaging, image processing, deep learning, and computer vision. He is currently a Senior Engineer with Samsung Research America.



Yash Sanghvi (Student Member, IEEE) received the B.Tech. and M.Tech degrees in electrical engineering from the Indian Institute of Technology, Bombay, Mumbai, India, in 2018. He is currently working toward the Ph.D. degree. In 2019, he joined Purdue University, West Lafayette, IN, USA, where he is currently a Graduate Research Assistant with Intelligent Imaging Lab, Electrical and Computer Engineering, Purdue University. From 2018 to 2019, he was working on the inverse scattering problem as a Project Scientist with the Indian Institute of Technology Madras, Chennai, India. His research interests include inverse problems, computational imaging, and deep learning.

Stanley H. Chan (Senior Member, IEEE) received the B.Eng. degree in electrical engineering from the University of Hong Kong, Hong Kong, in 2007, the M.A. degree in mathematics, and the Ph.D. degree in electrical engineering from the University of California at San Diego, San Diego, CA, USA, in 2009 and 2011, respectively. From 2012 to 2014, he was a Postdoctoral Research Fellow with Harvard University, Cambridge, MA, USA. In 2014, he joined Purdue University, West Lafayette, IN, USA, where he is currently an Elmore Associate Professor of electrical and computer engineering. He is the author of a popular undergraduate textbook *Introduction to Probability for Data Science*, Michigan Publishing 2021. He is a co-author of *Computational Imaging through Atmospheric Turbulence*, Now Publisher 2023. His research interests include single-photon imaging, imaging through atmospheric turbulence, and computational photography. He was the recipient of the IEEE Signal Processing Society Best Paper Award 2022 and IEEE International Conference on Image Processing Best Paper Award 2016. At Purdue, he was also the recipient of multiple teaching awards. He is a Senior Area Editor of IEEE TRANSACTIONS ON COMPUTATIONAL IMAGING (AE 2018-2022, SAE 2022-now) and IEEE OPEN JOURNAL ON SIGNAL PROCESSING since 2022. He is an Advising Member of IEEE Signal Processing Society Technical Committee in Computational Imaging. He was an elected Member of CI-TC in 2015-2020 and IVMSP-TC in 2023-2025.