Learning Weighted Least Squares Data Term for Poisson Image Deconvolution

Abhijit Singh, Emmanuel Soubies, Caroline Chaux

Learning Weighted Least Squares Data Term for Poisson Image Deconvolution

Abhijit Singh
CNRS@CREATE
Singapore
abhijit.singh@cnrsatcreate.sg

Emmanuel Soubies
CNRS, IRIT, Univ Toulouse
Toulouse, France
emmanuel.soubies@cnrs.fr

Caroline Chaux
CNRS, IPAL
Singapore
caroline.chaux@cnrs.fr

***Abstract*—Weighted least squares are often used to approximate log-likelihoods when solving inverse problems involving non-Gaussian noise as they are more appealing from an optimization perspective. Although a theoretical expression of the weights can be derived for specific noises, this may become intractable for more general noises. Moreover, such theoretical weights can be detrimental to the efficiency of optimization algorithms. To remedy these issues, we propose in this work to learn the weights from data so as to adapt to any general noise while maintaining the efficiency of optimization. The proposed pipeline combines a weight estimation module with an unrolled optimization algorithm. The weight estimation module and a few parameters of the unrolled algorithm are trained together in an end-to-end manner. We demonstrate the effectiveness of the proposed methodology in the context of Poisson image deconvolution.**

***Index Terms*—restoration, unrolled networks, hyperparameter learning, adaptive discrepancy, sparsity.**

I. Introduction

Let us consider the generic image formation model

$$\mathbf{y} = \mathcal{N}(\mathbf{A}\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^N$ is the unknown observed object, $\mathbf{A} \in \mathbb{R}^{M\times N}$ is a matrix that represents a linear operator, $\mathcal{N}$ is a noise model (e.g., multiplicative, additive, or mixed), and $\mathbf{y} \in \mathbb{R}^M$ is the vector of measurements.

Recovering the unknown observed object $\mathbf{x}$ from the noisy real-world observation $\mathbf{y}$ is an inverse problem which is addressed by solving an optimization problem formulated as

$$\hat{\mathbf{x}} \in \left\{ \underset{\mathbf{x}\in\mathbb{R}^N}{\arg\min}\ \mathcal{F}(\mathbf{x}) := \mathcal{D}_{\mathbf{y}}(\mathbf{A}\mathbf{x}) + \lambda\mathcal{R}(\mathbf{x}) \right\}, \tag{2}$$

where $\mathcal{D}_{\mathbf{y}}(\mathbf{A}\mathbf{x})$ is a data fidelity term which measures the discrepancy between the observed image and its model, $\mathcal{R}(\mathbf{x})$ is the regularization term which encodes the prior knowledge that we may have about the final solution, and $\lambda \in (0, \infty)$ is a hyperparameter which determines the trade-off between the data fidelity and regularization.

From a Bayesian point of view, the data fidelity term is related to the statistic of the noise. It corresponds to the negative log-likelihood function of the observation, given the model. For instance, when $\mathcal{N}$ represents an additive independent and identically distributed (i.i.d.) Gaussian noise, we have $\mathcal{D}_{\mathbf{y}}(\mathbf{A}\mathbf{x}) = \frac{1}{2}\left\|\mathbf{A}\mathbf{x} - \mathbf{y}\right\|_2^2$, the well-known least squares measure of fit. If, instead, $\mathcal{N}$ corresponds to a Poisson noise degradation, the negative log-likelihood is known to be the Kullback-Leibler (KL) divergence (see Section II for more details) [1], [2].

In practice, the noise affecting the data can be more sophisticated than simply Gaussian or Poisson, leading to more complex log-likelihood functions that are either very challenging to manipulate numerically (for example, mixed Poisson-Gaussian noise [3], [4]) or unknown explicitly. It is worth mentioning that even in the case of pure Poisson noise, the KL divergence already brings a few challenges in terms of optimization (e.g., lack of/bad Lipschitzity) [2].

Because of these difficulties, the use of complex log-likelihood functions remains scarce in practice. The least squares function is preferred for its good properties, even in cases where it is not compliant with the nature of the data. To strike a balance between modelling noise adequately while ensuring computational efficiency, there are alternatives that aim to approximate these complex log-likelihoods with simpler functions. These include the use of non-linear data transformations such as Bartlett [5] or Anscombe [6] transforms, as well as the consideration of weighted least squares data terms obtained through a second order Taylor expansion of the negative log-likelihood [7], [8].

With the recent advent of learning-based methods, some authors have proposed to directly learn the data fidelity term from data. The authors of [9] approximate the image prior with a deep neural network, while also training a separate deep neural network to learn the entire data fidelity term, as it is non-trivial to analytically model the prior distribution of the blurring kernel when doing blind image deconvolution. They unroll the Douglas-Rachford iterations and learn the proximal operators involved in it. In [10], the authors utilize a plug and play approach [11] to learn discriminative shrinkage functions for the data and regularization terms using deep convolutional neural networks (CNNs) with Maxout layers, for a non-blind image deconvolution task. Both these methods are computationally expensive as they require millions of parameters to be learnt.

This work was supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) Programme.

Contributions and outline. In this paper, we propose to learn a weighted data fidelity term which can adapt to any type of noise present in images. To do so, we perform end-to-end supervised training of a deep neural network made of a weight estimation module cascaded with an unrolled algorithm. More precisely, learned quantities include parameters defining the estimation module as well as some parameters of the unrolled optimization algorithm. As such, we ensure the interpretability of the proposed overall network with, in addition, the luxury of learning very few parameters if we are working under resource constraints or in a scenario with data scarcity. While this method can adapt to any type of noise, we focus in our experiments on the Poisson image deconvolution problem. This allows us to compare the learned weights with the theoretical weights obtained from the Bayesian interpretation.

II. Preliminaries

Assume that $\mathcal{D}_{\mathbf{y}}$ is twice differentiable, separable such that $\mathcal{D}_{\mathbf{y}}(\mathbf{z}) = \sum_{m=1}^{M} d_{y_m}(z_m)$, and admits a minimizer $\mathbf{c} \in \mathbb{R}^M$. Then, the second-order Taylor approximation of $\mathcal{D}_{\mathbf{y}}$ around $\mathbf{c}$ can be written as

$$\mathcal{D}_{\mathbf{y}}(\mathbf{z}) \approx \mathcal{D}_{\mathbf{y}}(\mathbf{c}) \; + \; \langle \mathbf{z}-\mathbf{c}, \nabla \mathcal{D}_{\mathbf{y}}(\mathbf{c}) \rangle + \; \langle \mathbf{z}-\mathbf{c}, \operatorname{diag}(\mathbf{w})\,(\mathbf{z}-\mathbf{c}) \rangle, \quad (3)$$

where $\operatorname{diag}(\mathbf{w}) \in \mathbb{R}^{M \times M}$, with $\mathbf{w} = (d''_{y_m}(c_m))_{m=1}^{M} \in \mathbb{R}^M$, denotes the Hessian matrix of $\mathcal{D}_{\mathbf{y}}$ at $\mathbf{c}$. The fact that the Hessian is diagonal comes from the separability of $\mathcal{D}_{\mathbf{y}}$. Ignoring the constant term and given that, by definition of $\mathbf{c}$, we have $\nabla \mathcal{D}_{\mathbf{y}}(\mathbf{c}) = \mathbf{0}$, we get,

$$\mathcal{D}_{\mathbf{y}}(\mathbf{A}\mathbf{x}) \approx \|\mathbf{A}\mathbf{x}-\mathbf{c}\|_{\mathbf{w}}^2 := \sum_{m=1}^{M} w_m \left([\mathbf{A}\mathbf{x}]_m - c_m\right)^2. \quad (4)$$

In other words, the data fidelity functional can be approximated by a weighted least squares where the weights are given by the diagonal entries of the Hessian of $\mathcal{D}_{\mathbf{y}}$ [7]. Then, we can consider the following problem in place of (2)

$$\hat{\mathbf{x}} \in \left\{ \underset{\mathbf{x} \in \mathbb{R}^N}{\arg\min} \; \|\mathbf{A}\mathbf{x}-\mathbf{c}\|_{\mathbf{w}}^2 + \lambda \mathcal{R}(\mathbf{x}) \right\}, \quad (5)$$

which is more appealing from an optimization point of view.

The case of Poisson noise. As mentioned in the introduction, although the proposed methodology is more general, in our experiments we test it in the case where $\mathcal{N}$ models a Poisson noise. In this case, $\mathbf{y} \in \mathbb{R}_+^M$ and $\mathcal{D}_{\mathbf{y}}$ corresponds to the KL divergence defined as: $\forall \mathbf{z} \in \mathbb{R}_+^M$, $\mathcal{D}_{\mathbf{y}}(\mathbf{z}) = \sum_{m=1}^{M} d_{y_m}(z_m)$, with

$$d_{y_m}(z_m) = z_m - y_m \log(z_m), \quad (6)$$

where constant terms are ignored. The gradient and diagonal Hessian of $\mathcal{D}_{\mathbf{y}}$ are fully defined by the one-dimensional first and second order derivatives

$$d'_{y_m}(z) = 1 - \frac{y_m}{z} \quad \text{and} \quad d''_{y_m}(z) = \frac{y_m}{z^2}. \quad (7)$$

From the strict convexity of $\mathcal{D}_{\mathbf{y}}$, we deduce from these expressions that $\mathcal{D}_{\mathbf{y}}$ admits a unique minimizer at $\mathbf{y}$ (indeed, $\nabla \mathcal{D}_{\mathbf{y}}(\mathbf{y}) = \mathbf{0}$) and that the diagonal entries of the Hessian at this minimizer are given by $w_m = 1/y_m$. Injecting these expressions in (4), we obtain the following weighted least squares approximation

$$\mathcal{D}_{\mathbf{y}}(\mathbf{A}\mathbf{x}) \approx \sum_{m=1}^{M} \frac{1}{y_m} \left([\mathbf{A}\mathbf{x}]_m - y_m\right)^2. \quad (8)$$

III. Proposed method

While analytic derivations may provide closed-form approximations for specific types of noise as shown above, they can be intractable (theoretically or numerically) for more complex noises. Moreover, when deploying, for instance, proximal gradient methods to solve (5), the step size should be of the order of $1/\|\mathbf{w}\|_\infty$ to ensure convergence, which can be very small (e.g., for Poisson noise) and lead to slow convergence. To overcome these two limitations, we propose in this work to learn the weights $\mathbf{w} \in \mathbb{R}^M$ along with some of the optimization parameters (e.g., step size) from data. This way, the weighted least squares approximation (4) can adapt to any general noise while ensuring the efficiency of optimization, provided one has access to a training set with pairs of clean and corrupted images.

To that end, we propose to cascade a weight estimation module (Section III-A) together with an unrolled algorithm to reconstruct the image (Section III-B). Parameters defining the weight estimation module as well as some optimization parameters of the unrolled algorithm are trained in an end-to-end manner using the whole pipeline (Section III-C).

A. Weight Estimation Module

We have seen in Section II that, theoretically, the weights $\mathbf{w}$ are nothing but the diagonal entries of the Hessian matrix of $\mathcal{D}_{\mathbf{y}}$. As a result, they are very likely to depend on the data $\mathbf{y}$, as it is the case for Poisson noise in (8). This motivates the definition of a mapping $g_{\boldsymbol{\theta}} : \mathbb{R}^M \to \mathbb{R}^M$, parameterized by the learnable vector $\boldsymbol{\theta} \in \mathbb{R}^P$, such that

$$\mathbf{w} = g_{\boldsymbol{\theta}}(\mathbf{y}), \quad (9)$$

provides an estimation of the weights from the data image $\mathbf{y}$.

In this work, we set $g_{\boldsymbol{\theta}}$ as an autoencoder-like architecture using convolutional layers. The motivation behind doing so was that we want to use an initial $\mathbf{w}$ (which in the case of Poisson noise corresponds to $1/\mathbf{y}$) as an input to $g_{\boldsymbol{\theta}}$, and get back an adjusted output which has the same dimensions as $\mathbf{w}$. An autoencoder-like architecture allows us to do so, while being parameter-efficient. The architecture chosen by us has only 3211 learnable parameters P. The details of this architecture are as follows.

The encoder part of $g_{\boldsymbol{\theta}}$ consists of two convolution layers, each of which is followed by ReLU activations and max-pooling layers. The first convolution layer takes a 3-channel input to match the coloured images we use for our experiments and produces 16 feature maps as the output, while the second

convolution layer uses those 16 feature maps to produce 8 feature maps as the output. The kernel size is 3×3 with a stride and padding of 1 for both the convolution layers. The max-pooling layers which follow each of the convolution layers use a kernel size of 2×2 with a stride of 2, to downsample the spatial dimensions of the feature maps.

The decoder part of $g_{\boldsymbol{\theta}}$ uses two layers of transposed convolutions to upsample the feature maps back to the original dimensions. The first transposed convolution layer uses the 8 feature maps of the encoder's output to produce 16 feature maps, while the second layer uses those to produce a 3-channel final output. Both transposed convolution layers use a kernel size of 3×3 with a stride of 2 and padding of 1. While the first transposed convolution layer uses a ReLU activation, the final layer uses a sigmoid activation. This fixes the upper bound of the values in the output matrix to 1 (in our implementation, we relax this upper bound to 2 by multiplying the outputs by 2). This is necessary to ensure that the magnitude of the values in the learned $\mathbf{w}$ are not very large, or else we will run into the same challenges of being restricted to very slow convergence, because as mentioned earlier, the step size needs to be of the order of $1/\|\mathbf{w}\|_\infty$ to ensure convergence.

B. Unrolled Forward Backward Splitting Algorithm

To solve (5), we propose to unroll the forward backward splitting (FBS) algorithm whose iteration is given by [12]

$$\begin{aligned}\mathbf{x}^{k+1} &= I(\lambda,\gamma,\mathbf{w};\mathbf{x}^k)\\ &:= \operatorname{prox}_{\gamma\lambda\mathcal{R}}(\mathbf{x}^k - \gamma\mathbf{A}^T\operatorname{diag}(\mathbf{w})(\mathbf{A}\mathbf{x}^k-\mathbf{y})),\end{aligned} \quad (10)$$

where $\operatorname{prox}_{\gamma\lambda\mathcal{R}}$ is the proximal operator of $\gamma\lambda\mathcal{R}$ and $\gamma > 0$ is the step size. To ensure convergence, it should be set as $\gamma \in (0, 1/L)$, where L is the Lipschitz constant of the weighted least squares term. It is bounded by $L \leq \|\mathbf{w}\|_\infty\sigma_{\max}(\mathbf{A})^2$ where $\sigma_{\max}(\mathbf{A})$ refers to the largest singular value of $\mathbf{A}$.

Given an initial point $\mathbf{x}^0 \in \mathbb{R}^N$ (we take $\mathbf{x}^0 = \mathbf{y}$ in this work), we define $\mathcal{T}(\lambda,\gamma,\mathbf{w};\cdot) : \mathbb{R}^M \to \mathbb{R}^N$ the unrolled FBS (with K iterations) as

$$\mathcal{T}(\lambda,\gamma,\mathbf{w};\mathbf{y}) = \underbrace{I(\lambda,\gamma,\mathbf{w};\cdots I(\lambda,\gamma,\mathbf{w};I(\lambda,\gamma,\mathbf{w};\mathbf{x}^0)))}_{K \text{ recursions}}.$$

Once trained (see Section III-C) and given $\mathbf{y}$, we get an estimate of the object $\mathbf{x}$ as

$$\hat{\mathbf{x}} = \mathcal{T}(\lambda,\gamma,\mathbf{w};\mathbf{y}). \quad (11)$$

C. Training

We learn the parameters $\boldsymbol{\theta}$ of the weight estimation module as well as the hyperparameters λ and step size γ through the resolution of

$$(\hat{\boldsymbol{\theta}},\hat{\lambda},\hat{\gamma}) = \arg\min_{\boldsymbol{\theta},\lambda,\gamma}\sum_{q=1}^{Q}\|\mathcal{T}(\lambda,\gamma,g_{\boldsymbol{\theta}}(\mathbf{y}^q_{\text{tr}});\mathbf{y}^q_{\text{tr}}) - \mathbf{x}^q_{\text{tr}}\|_2^2, \quad (12)$$

where $\{\mathbf{x}^q_{\text{tr}},\mathbf{y}^q_{\text{tr}}\}_{q=1}^Q$ is a set of input-target image pairs. This problem can be solved using standard ADAM or SGD optimizers. We implemented this within the framework of the DeepInverse library[1].

Remark 1: Theoretically, we do not need to learn λ because we can see in (5) that we can absorb λ within $\mathbf{w}$ by dividing the entire equation with λ. However, for small values of λ, the magnitude of the values in $\mathbf{w}$ can become too large if we do so, which will cause the same optimization problems of very slow convergence as discussed earlier. Additionally, as a result of our architectural choice of using the sigmoid activation in the final layer of $g_{\boldsymbol{\theta}}$ and multiplying it by 2, the upper bound of the values in $\mathbf{w}$ is fixed to 2. Thus, we will implicitly place a lower bound on λ if we fix λ and do not learn it. Further, for certain choices of regularizers (for instance, the ℓ_1 wavelet coefficient prior), we may have multi-dimensional λ, which would again make it infeasible to absorb it within $\mathbf{w}$.

IV. Experiments

A. Context

While the main idea proposed in Section III can be applied to any optimization problem that can be represented as (2), in this work we focus on wavelet-based deconvolution. This corresponds to the situation where

$$\mathbf{A} = \mathbf{H}\mathbf{W}^* \text{ and } \mathcal{R} = \|.\|_1, \quad (13)$$

with $\mathbf{H} \in \mathbb{R}^{N\times N}$ being a convolution operator and $\mathbf{W} \in \mathbb{R}^{N\times N}$ being an orthonormal wavelet operator. Thus, in our case, $\mathbf{x}$ represents the wavelet coefficients of the target image $\mathbf{q} = \mathbf{W}^*\mathbf{x}$. Moreover, the proximal operator in (10) is the well known soft thresholding operator defined as $\operatorname{prox}_{\gamma\lambda\|\cdot\|_1}(\mathbf{u}) = \operatorname{sign}(\mathbf{u})\max(|\mathbf{u}| - \lambda\gamma, 0)$.

We use the TAMPERE17 dataset [13] for our experiments, which has colour images of size $512\times512\times3$. We consider 10 unrolled iterations for the reconstruction part of our pipeline. We use the ADAM optimizer with a learning rate of 0.001 for 10 epochs on images corrupted by Poisson noise (of gain 0.01) in addition to a Gaussian blur (sigma=(1,1)). The training set is composed of 240 images, while the test set comprises of 60 images which are also corrupted by Poisson noise along with the same Gaussian blur as that used on the images in the training set.

B. Results

We compare the performance of the proposed method (denoted WLS) with the one obtained using a standard least squares (denoted LS) term (i.e., $\mathbf{w} = \mathbf{1}$), for which λ and step size γ are also trained in the same way as we do for the proposed method. Over the test dataset (60 images), the average PSNR is 26.91 dB for LS and 27.71 dB for WLS, which is a 0.8 dB improvement. However, it is worth mentioning that the performance is also highly dependant on the choice of the prior, which is a simple ℓ_1 norm on wavelet coefficients in these experiments. While we argue that we could obtain overall improved results through the consideration of more sophisticated (learned) priors, the point

[1]https://github.com/deepinv/deepinv

Fig. 1. Comparison (with zooms) of visual performance of some images in the test set.

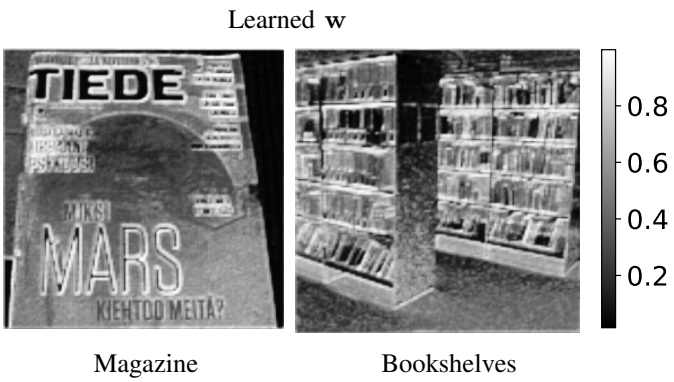

Fig. 2. Visualizing the learned $\mathbf{w}$ as greyscale images.

here is to show that learning a weighted least squares data fidelity term allows us to improve the performance, relative to the limitation imposed by the considered prior.

Beyond the PSNR metric, it is of interest to do a visual inspection of the deconvolved images. Indeed, the effect of adapting the data fidelity term to the nature of the noise can be better appreciated visually than in terms of metrics. For example, it is shown in [1] that for Poisson image denoising, the consideration of a KL data fidelity term allows for a better adjustment of the trade-off between reducing noise while preserving edges. We can observe a similar effect on the deconvolved images reported in Figure 1.

Further, in Figure 2 we also report the weights returned by our trained estimation module (i.e., $\mathbf{w} = g_{\hat{\theta}}(\mathbf{y})$). We can observe larger weights in regions where the signal in $\mathbf{y}$ is low. This is in line with the theoretical value of $\mathbf{w} = 1/\mathbf{y}$ in the case of Poisson noise. At the same time, the proposed strategy avoids the explosion of weights for low data values and thus ensures that we get a relevant deconvolution with only 10 unrolled iterations, which is not possible by setting $\mathbf{w} = 1/\mathbf{y}$.

As expected, the performance also depends on the strength of the Gaussian blur. When we use a stronger Gaussian blur (sigma=(3,3)), the average PSNR scores are lower (24.47 dB for LS and 24.56 dB for WLS). The improvement provided by WLS is likely lower because the strong Gaussian blur in this case makes it very difficult to recover edges and details that have been smoothed out by it.

V. Conclusion

In this work, we investigated the idea of learning a weighted least squares data fidelity term in order to adapt to many types of noise when solving an inverse problem. To that end, we train in an end-to-end manner the combination of a weight estimation module and an unrolled FBS algorithm where the step size and regularization hyperparameters are learnt. We thus end up with a light and highly interpretable architecture. We tested its performance on a classical Poisson image deconvolution problem with a sparse wavelet based prior. Numerical experiments show the benefits of learning a weighted least squares data term.

Future work will consider the use of more complex and general noises, in addition to more sophisticated (learnt) priors as well as possibly non-diagonal and non-linear weighting operations.

REFERENCES

[1] T. Le, R. Chartrand, and T. J. Asaki, "A variational approach to reconstructing images corrupted by poisson noise," *J. Math. Imaging Vision*, vol. 27, no. 3, pp. 257–263, 2007.

[2] 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, 2011.

[3] F. Benvenuto, A. La Camera, C. Theys, A. Ferrari, H. Lantéri, and M. Bertero, "The study of an iterative method for the reconstruction of images corrupted by poisson and gaussian noise," *Inverse Problems*, vol. 24, no. 3, p. 035016, 2008.

[4] E. Chouzenoux, A. Jezierska, J.-C. Pesquet, and H. Talbot, "A convex approach for image restoration with exact Poisson–Gaussian likelihood," *SIAM J. Imaging Sci.*, vol. 8, no. 4, pp. 2662–2682, 2015.

[5] M. S. Bartlett, "The square root transformation in analysis of variance," *Supplement to the Journal of the Royal Statistical Society*, vol. 3, no. 1, pp. 68–78, 1936.

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

[7] A. Sawatzky, C. Brune, J. Müller, and M. Burger, "Total variation processing of images with poisson statistics," in *International Conference on Computer Analysis of Images and Patterns*. Springer, 2009, pp. 533–540.

[8] S. Hurault, U. Kamilov, A. Leclaire, and N. Papadakis, "Convergent Bregman Plug-and-Play Image Restoration for Poisson Inverse Problems," *Adv. Neural Inf. Process. Syst.*, vol. 36, 2024.

[9] R. Aljadaany, D. K. Pal, and M. Savvides, "Douglas-rachford networks: Learning both the image prior and data fidelity terms for blind image deconvolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, 2019, pp. 10 235–10 244.

[10] P.-H. Kuo, J. Pan, S.-Y. Chien, and M.-H. Yang, "Learning discriminative shrinkage deep networks for image deconvolution," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2022, pp. 217–234.

[11] U. S. Kamilov, C. A. Bouman, G. T. Buzzard, and B. Wohlberg, "Plug-and-play methods for integrating physical and learned models in computational imaging: Theory, algorithms, and applications," *IEEE Signal Process. Mag.*, vol. 40, no. 1, pp. 85–97, 2023.

[12] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Model. Simul.*, vol. 4, no. 4, pp. 1168–1200, 2005.

[13] M. Ponomarenko, N. Gapon, V. Voronin, and K. Egiazarian, "Blind estimation of white gaussian noise variance in highly textured images," *arXiv preprint arXiv:1711.10792*, 2017.