Nonblind Image Deconvolution via Leveraging Model Uncertainty in An Untrained Deep Neural Network

Mingqin Chen[1], Yuhui Quan[1*], Tongyao Pang[2] and Hui Ji[2]

[1]School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510006, China.

[2]Department of Mathematics, National University of Singapore, 119076, Singapore.

[*]Corresponding author.

Abstract

Nonblind image deconvolution (NID) is about restoring the latent image with sharp details from a noisy blurred one using a known blur kernel. This paper presents a dataset-free deep learning approach for NID using untrained deep neural networks (DNNs), which does not require any external training data with ground-truth images. Based on a spatially-adaptive dropout scheme, the proposed approach learns a DNN with model uncertainty from the input blurred image, and the deconvolution result is obtained by aggregating the multiple predictions from the learned dropout DNN. It is shown that the solution approximates a minimum-mean-squared-error estimator in Bayesian inference. In addition, a self-supervised loss function for training is presented to efficiently handle the noise in blurred images. Extensive experiments show that the proposed approach not only performs noticeably better than existing non-learning-based methods and unsupervised learning-based methods, but also performs competitively against recent supervised learning-based methods.

1 Introduction

Image blurring is an often-seen image degradation in digital imaging. The spatially-invariant blurring effect on an image can be modeled as a convolution process as follows:

$$\boldsymbol{y} = \boldsymbol{k} * \boldsymbol{x} + \boldsymbol{n}, \tag{1}$$

where $\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{k}, \boldsymbol{n}$ denote the degraded image, latent image, blur kernel, and measurement noise respectively, and $*$ is the discrete convolution operator. Nonblind image deconvolution (NID) aims at recovering $\boldsymbol{x}$ from $(\boldsymbol{y}, \boldsymbol{k})$. It is an important technique for many tasks, such as deblurring with a hardware-assisted kernel estimation module and improving the image quality in microscopy imaging. Since convolving with a blur kernel may significantly attenuate or even erase the high-frequency components of the latent image, the problem (1) is ill-posed and the study of NID focuses on how to effectively handle such ill-posedness to have an accurate estimation of $\boldsymbol{x}$.

In recent years, supervised deep learning has emerged as a promising approach for NID with noticeable performance improvement over traditional methods; see *e.g.*, (Bigdeli et al, 2017; Kruse et al, 2017; Zhang et al, 2017a,c; Dong et al, 2018, 2019; Gilton et al, 2020; Nan et al, 2020; Gong et al, 2020; Dong et al, 2021). The basic idea is

This work is supported in part by National Natural Science Foundation of China under Grant 61872151, in part by CCF-Tencent Open Fund 2020, and in part by MOE AcRF Tier 1 Research Grant R146000315114.

training a deep neural network (DNN) over a large dataset of the triplets $(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{k})$, so as to have an effective mapping from $(\boldsymbol{y}, \boldsymbol{k})$ to $\boldsymbol{x}$.

In general, the generalization performance of a DNN model is highly dependent on whether the training data suffices to cover all important characteristics of test data. In practice, blurred images can vary a lot in terms of blurring effect, blurring amount, and noise strength. To ensure good generalization performance on test images, many existing methods train different models for different noise levels (Zhang et al, 2017a,c; Kruse et al, 2017; Bigdeli et al, 2017; Gong et al, 2020; Dong et al, 2021), or for different blur kernels (Dong et al, 2019; Gilton et al, 2020). Such an ad hoc treatment makes the DNNs for NID quite clumsy for practical usage, especially when there exists great variations in test data with respect to blur kernels and noise levels. Furthermore, in many domains, it is challenging or even impossible to collect ground-truth images that are highly related to the images of interest, *e.g.*, biological images of molecules, and medical images of patients.

In the last few years, there has been an increasing interest in utilizing untrained DNNs for solving linear inverse problems in imaging; see *e.g.*, (Ulyanov et al, 2018; Heckel and Hand, 2019; Wang et al, 2019; Zukerman et al, 2020; Hendriksen et al, 2020; Vedaldi et al, 2020). These approaches use a DNN to model the latent image and learn the DNN over the input degraded image itself. In comparison to non-learning-based methods, they exploit the implicit regularizations imposed by the structure of a DNN to have a more sophisticated image prior adaptive to test images, leading to promising performance. In comparison to supervised learning-based methods, they do not require any external sample for training (*i.e.*, dataset-free), which not only eliminates the cost of data collection but also avoids introducing any possible bias caused by a training dataset.

One pioneering work using untrained network priors for image recovery is the *deep image prior* (DIP) (Ulyanov et al, 2018). It showed that when training a convolutional DNN with a random input to fit an image, regular or repeating structures are likely to appear earlier than unstructured patterns (*e.g.*, noise) in the DNN's output. In order to have a restored image, DIP uses early stopping for avoiding possible overfitting. Based on the concept of DIP, there have been several dataset-free deep learning methods proposed for solving NID; see *e.g.*, (Wang et al, 2019; Zukerman et al, 2020). However, their performance is unsatisfactory in comparison to the top non-learning-based methods and not competitive against recent supervised learning-based methods.

1.1 Motivation

Motivated by the benefits of dataset-free learning methods for practical usage, as well as the unsatisfactory performance of the existing ones, this paper aims at developing an untrained DNN-based dataset-free approach for NID with state-of-the-art performance.

There are two main issues to address when developing an untrained DNN-based approach for NID. One is the presence of the measurement noise; and the other is the solution ambiguity (*i.e.*, existence of infinite solutions) of (1) arising from that certain high-frequency components of the latent image can be erased by the blurring. Inspired by the previous work on self-supervised denoising (Quan et al, 2020) which uses blind-spot training and dropout ensemble for untrained DNN-based image denoising, we extended such an idea to solve the NID problem.

Briefly, to handle the measurement noise in NID, this paper proposes a self-supervised loss function built upon a pixel replacement/masking scheme. This loss function, defined over a noisy blurred image, provides an unbiased estimate of the loss function defined over the noise-free blurred version of the ground-truth image. To tackle the solution ambiguity, this paper proposes to leverage dropout for learning a DNN with *model uncertainty*, *i.e.*, the weights of the DNN are treated as random variables. Such a DNN predicts a distribution of images rather than a single image. Then, the image estimates drawn from the distribution are aggregated to handle the solution ambiguity in NID and to have an improved estimate. To further improve the performance of the proposed approach, a spatially-adaptive (SA) dropout scheme is introduced to better embed model uncertainty into the DNN.

1.2 Contributions

There are three main contributions in this work:

- By introducing a SA dropout scheme and a noise-resistant self-supervised training loss, a dataset-free deep learning approach built upon model uncertainty is proposed.
- A theoretical insight is provided to understand the proposed dropout-based NID method, which reveals its connection to the DNN-based MMSE estimator in Bayesian inference.
- Experimental validation on several benchmarks shows the superiority of the proposed approach over existing non-learning-based methods and untrained DNN-based methods. The proposed approach also provides competitive and even better performance than the DNNs trained with external ground-truth images.

While the work presented in this paper drew inspirations from (Quan et al, 2020), the extension to the NID problem has its originality with new techniques and new insights. Recall that there exists the solution ambiguity in NID due to the blurring. Thus, different from image denoising, NID needs to handle both measurement noise and solution ambiguity. In this work, we extend the self-supervised training loss of (Quan et al, 2020) from denoising to NID, introduce a SA dropout scheme for better exploiting model uncertainty, and present a theoretical insight of the proposed work from Bayesian inference. The latter two are not available in (Quan et al, 2020). It is noted that the code of this work implemented with PyTorch v1.6 will be available at https://github.com/scut-mingqinchen/.

2 Related Work

NID with handcrafted/learned image priors. A majority of existing methods for NID address the ill-posedness of the NID problem by imposing certain priors on the estimate to regularize the deconvolution process. The ℓ_1-norm regularization-based methods assume the gradients of the latent image are sparse; see *e.g.*, (Vonesch and Unser, 2008; Krishnan and Fergus, 2009). The non-local methods assume the recurrence of local patches on the latent image; see *e.g.*, (Danielyan et al, 2011). The data-driven methods learn image priors from clear images; see *e.g.*, (Zoran and Weiss, 2011; Arridge et al, 2019). One closely-related work to ours is (Schmidt et al, 2011) which performs Bayesian NID with a learned prior through sample averaging.

NID with supervised DNNs. In recent years, there have been many works on training a DNN for NID using an external dataset containing many triplets of blurry images, latent sharp images and blur kernels. Some methods such as (Schuler et al, 2013; Xu et al, 2014; Son and Lee, 2017; Vasu et al, 2018; Ren et al, 2018; Dong et al, 2021) train a DNN as a post-denoiser to remove the artifacts from the deblurred result of some existing method. A more prominent approach is unrolling some iterative deblurring scheme into a DNN with plug-and-play denoising modules or trainable modules; see *e.g.*, (Schmidt and Roth, 2014; Zhang et al, 2017a,c; Meinhardt et al, 2017; Romano et al, 2017; Kruse et al, 2017; Jin et al, 2017; Bigdeli et al, 2017; Nan et al, 2020; Gilton et al, 2020; Gong et al, 2020; Eboli et al, 2020).

NID with untrained DNN priors. While DIP (Ulyanov et al, 2018; Vedaldi et al, 2020) with untrained DNNs can be directly applied to unsupervised learning-based NID, there are only a few works along this line. Heckel and Hand (2019) proposed a DNN with an under-parameterized structure to prevent the DNN from overfitting the undesired patterns. For performance improvement, Wang et al (2019) combined DIP with the sparsity prior of image gradients, and Zukerman et al (2020) used DIP with back projection. The performance of these unsupervised learning-based methods is not competitive to the supervised ones and even not as good as the top performers in non-learning-based methods.

Unsupervised deep learning for denoising. In relation to NID, there are more studies on the unsupervised deep learning for denoising; see *e.g.*, (Ulyanov et al, 2018; Lehtinen et al, 2018; Ehret et al, 2019; Krull et al, 2019; Batson and Royer, 2019; Laine et al, 2019; Soltanayev and Chun, 2018). These methods either use DIP with early stopping for dataset-free learning (Ulyanov et al, 2018), or use some blind-spot mechanism for preventing the DNN converging to an identity mapping (Krull et al, 2019; Batson and Royer, 2019; Laine et al, 2019), or use dropout-based ensemble for improving single-image learning (Quan et al, 2020). As discussed in Section 1.2, NID is quite different from image denoising as it needs to resolve solution ambiguity. The extension of the

idea from denoising to NID is usually non-trivial. For instance, DIP performs well for image denoising, but its current extensions to NID do not work as well, as shown in our experiments.

3 Proposed Method

3.1 Basic Idea

Consider a DNN denoted by $f_{\boldsymbol{\theta}}(\cdot)$ with parameters $\boldsymbol{\theta}$, which estimates $\boldsymbol{x}$ based on the input $\boldsymbol{y} = \boldsymbol{k} * \boldsymbol{x} + \boldsymbol{n}$. In the presence of Gaussian white noise $\boldsymbol{n}$, the maximum likelihood estimate for $\boldsymbol{x}$ is then $f_{\boldsymbol{\theta}^*}(\boldsymbol{\epsilon}_0)$, where $\boldsymbol{\theta}^*$ is determined by solving

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{k} * f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}_0)\|_2^2, \tag{2}$$

for some seed $\boldsymbol{\epsilon}_0$. Recall that the high-frequency components of $\boldsymbol{x}$ are either significantly attenuated or completely erased by the convolution with the blur kernel $\boldsymbol{k}$. Then, there exists a non-empty approximate null space

$$\mathcal{N}(\boldsymbol{k}) = \{\boldsymbol{e} : \ \|\boldsymbol{k} * \boldsymbol{e}\|_2 \ll 1\} \tag{3}$$

such that any image in the subspace defined by

$$\tilde{\boldsymbol{x}} \in \{\boldsymbol{x} + \boldsymbol{e} : \ \boldsymbol{e} \in \mathcal{N}(\boldsymbol{k})\} \tag{4}$$

will have roughly the same loss as $\boldsymbol{x}$, *i.e.*, $\|\boldsymbol{y} - \boldsymbol{k} * \tilde{\boldsymbol{x}}\|_2^2 \approx \|\boldsymbol{y} - \boldsymbol{k} * \boldsymbol{x}\|_2^2$. In other words, there exists solution ambiguity when training a DNN with (2), where any prediction in the subspace defined by (4) is viewed as a correct one.

Remark 1. *The error $\boldsymbol{e} \in \mathcal{N}(\boldsymbol{k})$ is usually correlated to the latent image structures, which cannot be well handled by DIP (Ulyanov et al, 2018) that prefers regular structures over random noise and unstructured patterns in its prediction. See Fig. 1 for a visualization of the predictions of DIP at different learning stages. There are some structured errors appearing during learning and being kept in the later stages.*

We address the solution ambiguity caused by the approximate null space $\mathcal{N}(\boldsymbol{k})$ via introducing model uncertainty to a DNN model f, *i.e.*, treating its parameters as random variables. Then, f is trained to give different estimates on $\boldsymbol{x}$ which are outputted by f with different instances of its parameters. Once the solution ambiguity arising from $\mathcal{N}(\boldsymbol{k})$ can be captured by these estimates well, we can aggregate these estimates to have an estimate of $\boldsymbol{x}$ with high accuracy.

In order to introduce model uncertainty to the DNN such that it is effective for reducing the solution ambiguity while being efficient in computation, we propose to train f with a SA dropout scheme. The dropout training enables a single DNN to simulate a large number of DNNs of different structures by randomly dropping nodes, which is computationally efficient to have a DNN with model uncertainty. After the dropout training, the estimation on $\boldsymbol{x}$ is done by keeping the SA dropout enabled during test. As a result, one can have multiple estimates of $\boldsymbol{x}$ from multiple instances of f, and the final prediction is obtained by the aggregation over these estimates. Such a dropout-based scheme has its connection to the MMSE estimator, which is shown in Section 3.3. Thus, it can well address the solution ambiguity caused by $\mathcal{N}(\boldsymbol{k})$.

3.2 Training Scheme

The DNN we employ is constructed with its layers configured with dropout. For each layer with dropout, there is a probability for each weight being zeroed at each iteration. In other words, the parameter vector of our DNN with dropout, denoted by $\boldsymbol{\beta}$, can be expressed as $\boldsymbol{\beta} = \boldsymbol{\theta} \odot \boldsymbol{b}$, where the entries of $\boldsymbol{b}$ are a set of independent random variables which obey Bernoulli distributions with different parameters. Let $\mathbb{E}$ denote the expectation of random variables. Let $\boldsymbol{\Omega} = \{\widehat{\boldsymbol{y}}_\ell\}_\ell$ denote the set of seeds used as the DNN's input. Given the kernel $\boldsymbol{k}$, the DNN is trained to map the seeds in $\boldsymbol{\Omega}$ to the blurred image $\boldsymbol{y}$:

$$\min_{\boldsymbol{\theta}} \ \mathbb{E}_{\widehat{\boldsymbol{y}} \sim \boldsymbol{\Omega}} \mathbb{E}_{\boldsymbol{b}} \, \mathcal{C}(\boldsymbol{y}, \boldsymbol{k} * f_{\boldsymbol{\theta} \odot \boldsymbol{b}}(\widehat{\boldsymbol{y}})), \tag{5}$$

for some loss function $\mathcal{C}(\cdot, \cdot)$. The training using (5) is realized by dropout. The training procedure stops if either the maximum iteration number is reached or the approximation error $\|\boldsymbol{y} - \boldsymbol{k} * f_{\boldsymbol{\beta}}(\widehat{\boldsymbol{y}})\|_2$ is less than a pre-defined tolerance value for some $\widehat{\boldsymbol{y}} \in \boldsymbol{\Omega}$ and $\boldsymbol{\beta}$.

There are two items with randomness in (5): the seed set $\boldsymbol{\Omega}$ and the Bernoulli variable set $\boldsymbol{b}$. The former is used for addressing the measurement noise in $\boldsymbol{y}$, together with a specifically-designed loss function $\mathcal{C}$. The latter is used for

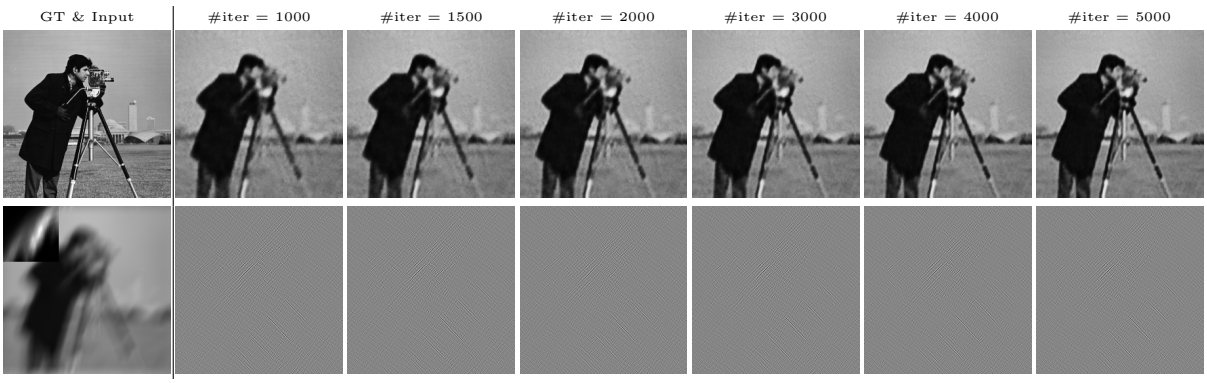


Fig. 1 Illustration of the prediction error in the approximate null space at different iterations of DIP. The top row shows the output images over different iterations. The bottom row shows the the errors in the approximate null space. It can be seen that there are some artifacts around the cameraman' head and the camera tripod, along the learning process.

having a DNN with model uncertainty, which is used for addressing the solution ambiguity in NID. The remaining part of this section is devoted to the design of $\mathbf{\Omega}$ and $\mathcal{C}$.

Seeds. Instead of using random noise as the seeds like DIP, we generate a set of randomized versions of $\boldsymbol{y}$, as the $\mathbf{\Omega}$, using the following scheme:

$$\widehat{\boldsymbol{y}}_\ell = \boldsymbol{m}_\ell \odot \boldsymbol{y} + (\mathbf{1} - \boldsymbol{m}_\ell) \odot (\mathcal{A} \circ (\boldsymbol{m}_\ell \odot \boldsymbol{y})), \quad (6)$$

where $\odot$ denotes the element-wise multiplication operator, $\boldsymbol{m}_\ell$ is a binary random Bernoulli mask with probability $p_{\boldsymbol{m}}$, and $\mathcal{A}$ is the operation that averages the non-zeros in the 8 neighbors of each pixel. In other words, each seed is constructed by randomly selecting some pixels and replacing their pixel values with the average of their neighboring non-selected ones. Each $\widehat{\boldsymbol{y}}_\ell$ encodes a large portion of low-frequency information of $\boldsymbol{x}$, encouraging the model to focus more on the high-frequency parts. See Fig. 2 (b) for two examples of the seeds.

Loss function. It can be seen that some parts of the input $\boldsymbol{y}$, *i.e.*, $(\mathbf{1} - \boldsymbol{m}_\ell) \odot \boldsymbol{y}$, are removed in the construction of a seed $\widehat{\boldsymbol{y}}_\ell$. Define

$$\|\cdot\|_{\boldsymbol{m}} = \|(\mathbf{1} - \boldsymbol{m}) \odot \cdot\|_2. \quad (7)$$

Then, the loss function $\mathcal{C}$ is constructed in a self-supervised manner which measures how well the model predicts the removed part using each seed:

$$\mathcal{C}(\boldsymbol{y}, \boldsymbol{k} * f_{\boldsymbol{\theta} \odot \boldsymbol{b}}(\widehat{\boldsymbol{y}}_\ell)) = \|\boldsymbol{k} * f_{\boldsymbol{\theta} \odot \boldsymbol{b}}(\widehat{\boldsymbol{y}}_\ell) - \boldsymbol{y}\|^2_{\boldsymbol{m}_\ell}. \quad (8)$$

The design of such a loss aims at eliminating the influence of the measurement noise of $\boldsymbol{y}$ during training. As shown in Proposition 1, the loss function (8) used for training the DNN is an unbiased estimate of the one defined over $\boldsymbol{k} * \boldsymbol{x}$, *i.e.*, the noise-free version of $\boldsymbol{y}$. In other words, the measurement noise in the input is effectively removed when training the DNN using the proposed loss.

Proposition 1. *Consider $\boldsymbol{y} = \boldsymbol{k} * \boldsymbol{x} + \boldsymbol{n}$. Assume the measurement noise $\boldsymbol{n}$ is of zero mean. Its entries are independent from each other and also independent from $\boldsymbol{x}$. Let $\widehat{\boldsymbol{y}}$ be defined by (6). Then, the expectation of the loss function defined in (8) over $\boldsymbol{n}$ is the same as that of*

$$\sum_\ell \|\boldsymbol{k} * f_{\boldsymbol{\theta} \odot \boldsymbol{b}}(\widehat{\boldsymbol{y}}_\ell) - \boldsymbol{k} * \boldsymbol{x}\|^2_{\boldsymbol{m}_\ell} + \sum_\ell \|\boldsymbol{\sigma}\|^2_{\boldsymbol{m}_\ell}, \quad (9)$$

for any model $f_{\boldsymbol{\beta}}$, where $\boldsymbol{\sigma}(i)$ denotes the standard deviation of $\boldsymbol{n}(i)$.

Proof See the proof in Appendix A. □

By definition, the loss (8) is defined to measure the prediction error of the DNN on the noisy blurred image $\boldsymbol{y} = \boldsymbol{k} * \boldsymbol{x} + \boldsymbol{n}$ over the un-replaced pixels of a seed. Proposition 1 shows that in terms of expectation, the loss (8) is equivalent to measuring the prediction error of the DNN on the noise-free blurred image $\boldsymbol{k} * \boldsymbol{x}$ over the un-replaced pixels of a seed, in terms of expectation. As the final loss (5) is defined over many perturbed seeds from $\mathbf{\Omega}$, the union of many instances of un-replaced pixels is likely to cover all image pixels. As a result, the final loss indeed measures the prediction error of the DNN on the noise-free blurred image $\boldsymbol{k} * \boldsymbol{x}$ over all image pixels. In other words,

the measurement noise $\boldsymbol{n}$ is effectively removed when training with (5).

Remark 2. *While the self-supervised training with the loss (8) is related to the blind spot training used in self-supervised image denoising (Krull et al, 2019; Batson and Royer, 2019), they indeed play different functions. In denoising, the training without a blind-spot mechanism will fail, as the model will converge to the identity mapping. This is not the case for NID, as the estimate from the identity mapping is not a minimizer. In NID, the self-supervised loss (8) is for better addressing the noise sensitivity of deconvolution. While a blind-spot mechanism is critical for self-supervised image denoising, the loss (8) is not that critical for self-supervised NID. The ablation study shows that it only makes a modest contribution to the performance. The main performance gain comes from the proposed model-uncertainty-based scheme with the SA dropout-based implementation.*

Remark 3. *Proposition 1 assumes that the kernel used for deconvolution is accurate. It sometimes occurs in practice that the kernel is estimated by some method and thus can be inaccurate with error $\Delta\boldsymbol{k}$. In this case, there exists another noise source $\Delta\boldsymbol{k} * \boldsymbol{x}$, as $\boldsymbol{y} = (\boldsymbol{k} + \Delta\boldsymbol{k}) * \boldsymbol{x} + \boldsymbol{n} = \boldsymbol{k} * \boldsymbol{x} + (\Delta\boldsymbol{k} * \boldsymbol{x} + \boldsymbol{n})$. The $\Delta\boldsymbol{k} * \boldsymbol{x}$ is some signal-dependent noise which does not satisfy the signal independence assumed in Proposition 1 for noise. Empirically, the proposed method still works well when the kernel error is small, as shown in Section 4.3.2. However, its performance is limited for large kernel error, as shown in Section 4.5.*

3.3 Prediction Scheme

Once the DNN with dropout is trained using the loss function (5), we utilize such a model with uncertainty in the test to address the solution ambiguity in NID. Let $\boldsymbol{\theta}^*$ be the vectorized model parameters from dropout training. Instead of using $\mathbb{E}_{\boldsymbol{b}}[\boldsymbol{\theta}^* \odot \boldsymbol{b}]$ for prediction (Srivastava et al, 2014), *i.e.*, $\overline{\boldsymbol{x}} = \mathbb{E}_{\widehat{\boldsymbol{y}}\sim\boldsymbol{\Omega}} f_{\mathbb{E}_{\boldsymbol{b}}[\boldsymbol{\theta}^*\odot\boldsymbol{b}]}(\widehat{\boldsymbol{y}})$, we define the prediction by

$$\boldsymbol{x}^* = \mathbb{E}_{\widehat{\boldsymbol{y}}\sim\boldsymbol{\Omega}}\mathbb{E}_{\boldsymbol{b}}\ f_{\boldsymbol{\theta}^*\odot\boldsymbol{b}}(\widehat{\boldsymbol{y}}), \tag{10}$$

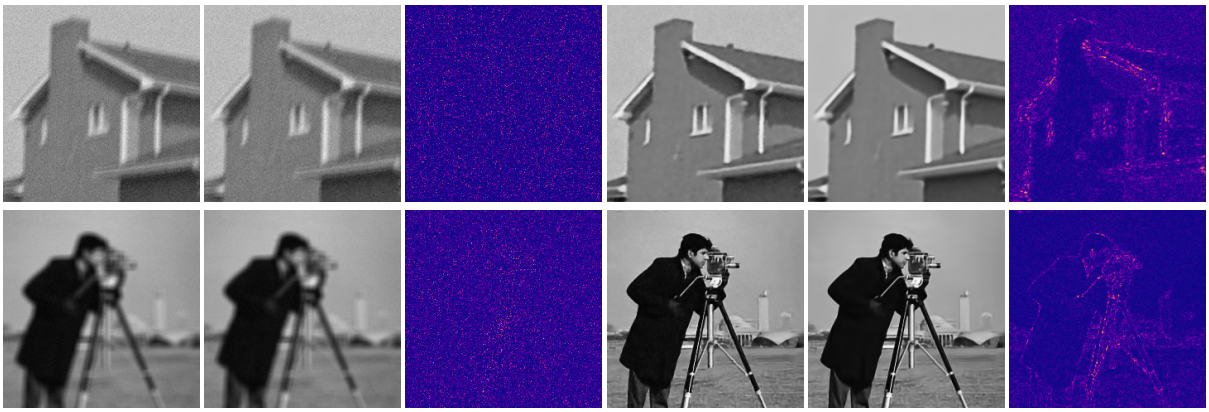
which is approximated by the Monte Carlo integration in practice, *i.e.*, inferring with dropout many times and averaging the inference results as the final prediction. In other words, at test time, the original dropout strategy is to evaluate the input with the expectation of the weights. Whereas in our case, we compute the expectation over the outputs with masking weights still active. This approach is closely related to computing an MMSE estimate of the deblurred image, as shown in the later part of this section. See Fig. 2 for a visual inspection of the images generated in different stages of the proposed method.

See Fig. 3 for an illustration of the relation between the solution ambiguity in NID and the model uncertainty in our dropout-trained DNN. With a DNN trained on a blurred image using the proposed scheme, we generate 100 predictions of the latent image by 100 dropout-based inferences. Such 100 predictions and their average, together with the ground-truth image, are visualized in Fig. 3 (a) via PCA projection. It can be seen that the predictions are randomly scattered around the ground-truth image, and their average is the one closest to the ground-truth image. The PSNRs of individual and average predictions *vs.* the inference times are plotted in Fig. 3 (b). The PSNR of the average prediction increases as more inferences are averaged. See Fig. 3 (c) and (d) for an additional example. All these empirical observations indicate the soundness of our idea, *i.e.*, using model uncertainty to resolve the solution ambiguity induced by $\mathcal{N}(\boldsymbol{k})$.

The rationale of (10) comes from the interpretation of dropout from the perspective of Bayesian approximation. Roughly speaking, the dropout training tries to approximate the posterior probability distribution of model parameters by the Bernoulli distribution. If the approximation accuracy is very high, the prediction $\boldsymbol{x}^*$ given by (10) is nearly an MMSE estimate for the ground-truth image $\boldsymbol{x}$. To be more specific, assume that $\boldsymbol{x}$ can be represented by

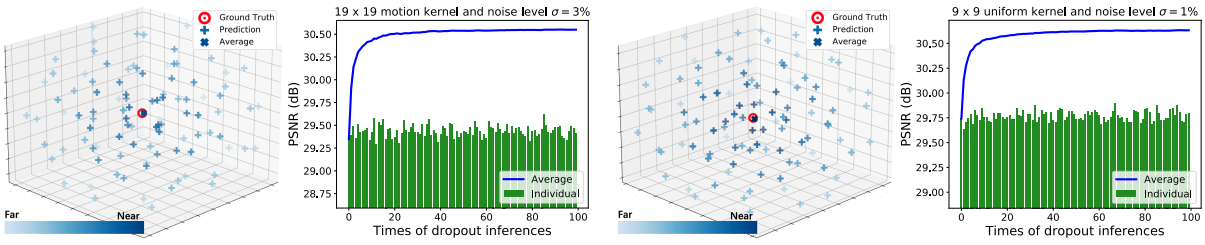
$$\boldsymbol{x} = f_{\boldsymbol{\beta}}(\widehat{\boldsymbol{y}}),\ \widehat{\boldsymbol{y}} \sim \boldsymbol{\Omega}, \tag{11}$$

with the model parameter $\boldsymbol{\beta}$ obeying some prior probability distribution when $\boldsymbol{y}$ is given. Recall that our training is using $\widehat{\boldsymbol{y}}_\ell$ in the form of (6) to predict its removed part in $\boldsymbol{y}$, *i.e.*, $(\boldsymbol{1}-\boldsymbol{m}_\ell)\odot\boldsymbol{y}$. We denote $\mathcal{D} = \{\widehat{\boldsymbol{y}}_\ell, (\boldsymbol{1}-\boldsymbol{m}_\ell)\odot\boldsymbol{y}\}_\ell$ as the training data



(a) Input (b) Seed (c) Difference (d) Single Inference (e) Average (f) Difference

Fig. 2 Visualization of (a) input image; (b) seed; (c) difference between (a) and (b); (d) single prediction from the trained dropout DNN; (e) average of multiple predictions from the trained dropout DNN; and (f) difference between (d) and (e).



(a) PCA on 'House' (b) Results on 'House' (c) PCA on 'Cameraman' (d) Results on 'Cameraman'

Fig. 3 Demonstration of proposed model-uncertainty-based scheme for resolving the solution ambiguity in nonblind image deconvolution. A dropout-based DNN is trained on two blurred images in Fig. 2: 'House' and 'Cameraman'. Then, 100 predictions by 100 are generated from the DNN with dropout. The 100 predictions, their average, and the ground-truth image, are visualized in (a) and (c) via PCA projection. Their PSNR values vs. inference times are plotted in (b) and (d).

generated from $\boldsymbol{y}$. Given $\mathcal{D}$, the MMSE estimate for $\boldsymbol{x}$ is known to be

$$\begin{aligned}\widehat{\boldsymbol{x}} = \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y}} &= \int \boldsymbol{x} p(\boldsymbol{x}|\boldsymbol{\beta}, \widehat{\boldsymbol{y}}) p(\boldsymbol{\beta}|\mathcal{D}) d\boldsymbol{x} d\boldsymbol{\beta} d\widehat{\boldsymbol{y}} \\ &= \mathbb{E}_{\widehat{\boldsymbol{y}}\sim\boldsymbol{\Omega}} \int f_{\boldsymbol{\beta}}(\widehat{\boldsymbol{y}}) p(\boldsymbol{\beta}|\mathcal{D}) d\boldsymbol{\beta}.\end{aligned} \tag{12}$$

Due to the high dimensionality of $\boldsymbol{\beta}$ and the complex structure of the DNN f, the posterior probability $p(\boldsymbol{\beta}|\mathcal{D})$ is intractable. The Bayesian approximation uses a distribution $q(\boldsymbol{\beta}|\boldsymbol{\theta})$ to approximate it (Blei et al, 2017). In the case of dropout, $q(\boldsymbol{\beta}|\boldsymbol{\theta})$ is defined based on the Bernoulli distribution (Gal and Ghahramani, 2016):

$$\boldsymbol{\beta} = \boldsymbol{\theta} \odot \boldsymbol{b}, \text{ where } \boldsymbol{b}(i) \sim \text{Bernoulli}(p_i). \tag{13}$$

That is, the $\boldsymbol{x}^*$ of (10) is predicted by using $q(\boldsymbol{\beta}|\boldsymbol{\theta}^*)$ to approximate $p(\boldsymbol{\beta}|\mathcal{D})$, which is

$$\boldsymbol{x}^* = \mathbb{E}_{\widehat{\boldsymbol{y}}\sim\boldsymbol{\Omega}} \int f_{\boldsymbol{\beta}}(\widehat{\boldsymbol{y}}) q(\boldsymbol{\beta}|\boldsymbol{\theta}^*) d\boldsymbol{\beta} = \mathbb{E}_{\widehat{\boldsymbol{y}}\sim\boldsymbol{\Omega}} \mathbb{E}_{\boldsymbol{b}} f_{\boldsymbol{\theta}^*\odot\boldsymbol{b}}(\widehat{\boldsymbol{y}}). \tag{14}$$

If two functions $p(\boldsymbol{\beta}|\mathcal{D})$ and $q(\boldsymbol{\beta}|\boldsymbol{\theta}^*)$ are the same, $\boldsymbol{x}^*$ is indeed exactly the MMSE estimate. The approximation of $q(\boldsymbol{\beta}|\boldsymbol{\theta}^*)$ to $p(\boldsymbol{\beta}|\mathcal{D})$ is done by minimizing the KL divergence between $p(\boldsymbol{\beta}|\mathcal{D})$ and $q(\boldsymbol{\beta}|\boldsymbol{\theta})$. Under proper assumptions, minimizing the KL divergence between $p(\boldsymbol{\beta}|\mathcal{D})$ and $q(\boldsymbol{\beta}|\boldsymbol{\theta})$ is identical to the training with (5). See Appendix B for more details.

3.4 Spatially-Adaptive Dropout

In the standard dropout (Srivastava et al, 2014), the dropout ratios are all the same with a predefined constant, which implies that the uncertainty degrees of all neurons of the DNN are kept

the same. Such a uniform uncertainty degree over all neurons is not consistent with the spatially-varying uncertainty empirically observed in the prediction. This inconsistency motivated the SA dropout scheme in our approach. It is based on the observation that after the DNN is trained by dropout, the empirical uncertainty degree of each neuron is closely related to the variance of sampled instances of each neuron. Then, such a variance can be used to approximate the optimal uncertainty degree (dropout ratio) of each neuron.

Let $\boldsymbol{p}^{(l)}$ denote the dropout ratio in the l-th dropout layer, which is of the same shape as the neurons. During learning, we sample multiple inputs from $\boldsymbol{\Omega}$ and multiple models from the currently-trained DNN. Then we can generate multiple instances of each neuron in the dropout layers of the DNN such that their empirical variances, denoted by $\boldsymbol{v}^{(l)}$, can be calculated. The theoretical variances of neurons with dropout are given by $\boldsymbol{p}^{(l)} \odot (\mathbf{1} - \boldsymbol{p}^{(l)})$, which attains the largest value at 0.5. When the dropout ratio is set to be below or equal to 0.5, then the larger the dropout ratio is, the larger the variance is.

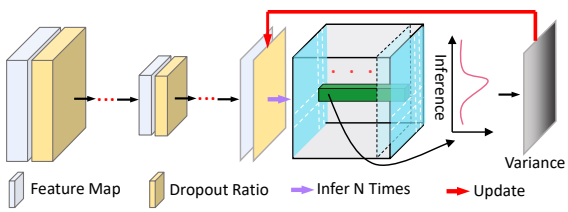


Fig. 4 Flowchart of proposed SA dropout.

Considering the relation between theoretical variances and empirical variances, we first scale $\boldsymbol{v}^{(l)}$ to the range $[0.5, 1.5]$ and update $\boldsymbol{p}^{(l)}$ according to $\boldsymbol{v}^{(l)}$ as follows:

$$\boldsymbol{p}^{(l)} = \min\{\boldsymbol{v}^{(l)} \odot \boldsymbol{p}_0^{(l)}, 0.5\},\ l = 1, \cdots, L, \quad (15)$$

where $\boldsymbol{p}_0^{(l)}$ denotes the former dropout ratio. Afterwards, we continue the training using the updated $\boldsymbol{p}^{(l)}$. Such an update process is performed every 1000 iterations. See Fig. 4 for the diagram. The empirical variance of each neuron in the dropout layers is indeed that of each pixel in the feature map. From the last dropout DNN, we can generate multiple instances for each feature map, which are then used to calculate the empirical variance of each pixel in the feature map. In Fig. 5, we plot the pixel-wise variance maps (*i.e.*, the empirical variances of neurons) in some intermediate layers of our DNN.

4 Experiments

In this section, experiments are done to evaluate the performance of the proposed method in various scenarios, including the ideal cases with accurate blur kernels of motion types and non-motion types, and the real-world cases including the ones with real-world degradations and the ones with inaccurate kernels. Ablation studies are then conducted to verify the effectiveness of each component of the proposed method. Finally, a limitation analysis is given.

4.1 Settings and Details

Implementation. The DNN $f_{\boldsymbol{\beta}}$ we use through all the experiments is an encoder-decoder fully-convolutional network with common blocks. There are six (five) blocks in the encoder (decoder) part. Each of these blocks contains a convolutional layer equipped with dropout on its input. Since the feature sizes are reduced (enlarged) in the intermediate layers of the encoder (decoder), the dropout ratios are initialized as a symmetric increasing (decreasing) sequence for the encoder (decoder): from 7.5% to 45% with an increment 7.5% on the encoder, and from 37.5% to 7.5% with a decrement 7.5% on the decoder.

See Table 1 for the detailed DNN architecture. Following the common settings, all convolutional layers are with kernel size 3×3, stride 1, and zero padding of length 2. All max pooling layers are set with the CEIL mode (Paszke et al, 2017), and the feature maps are clipped to the same size before concatenation. The bi-linear interpolation is used for upsampling. The hyper-parameter of each LReLU is set to 0.01.

In training, the probability for generating Ω is set to 0.3. Unless specified, the Adam optimizer is used with initial learning rate of 10^{-4}. The stopping threshold is set by the estimated noise level from the estimator (Chen et al, 2015) with a stabilizer 5×10^{-4}, and the maximal epoch number is set to 10^5. In prediction, we use 50 times of dropout inferences.

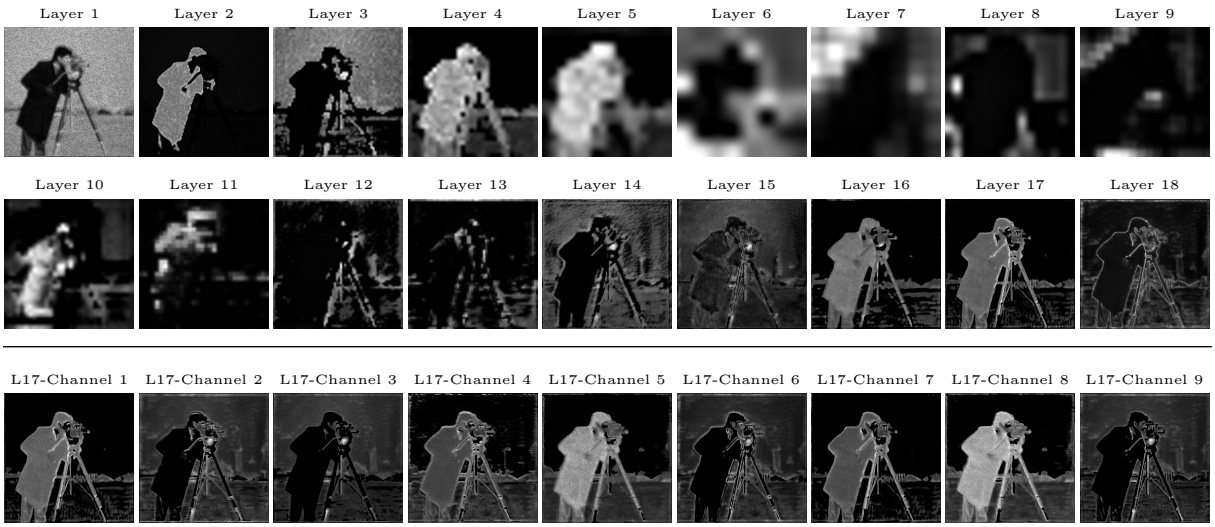


Fig. 5 Visual results of pixel-wise variance maps: The top two rows show the variance maps in different network layers and the last row shows the variance maps in different channels of the 17th layer. This demo is evaluated on 'Cameraman' with the 5th motion kernel from (Levin et al, 2011) and Gaussian noise level $\sigma = 0.05$.

Table 1 DNN architecture used in the proposed method. Parameters: initial dropout ratio for Dropout, receptive field size for Max Pooling, scaling factor for Upsampling, and two block sequence numbers (outputs of which blocks to be concatenated) for Concatenation. The dropout ratios are the initial ones and will be updated during learning.

NO.	Function	# Channels	Parameter(s)	NO.	Function	# Channels	Parameter(s)
	ENCODER			17	Dropout + Conv. + LReLU	96	37.5%
1	-	C	-	18	Upsampling	96	2X
2	Dropout + Conv. + LReLU	48	7.5%	19	Concatenation	144	#8, #18
3	Dropout + Conv. + LReLU	48	7.5%	20	Dropout + Conv. + LReLU	96	30%
4	Max Pooling	48	2×2	21	Dropout + Conv. + LReLU	96	30%
5	Dropout + Conv. + LReLU	48	15%	22	Upsampling	96	2X
6	Max Pooling	48	2×2	23	Concatenation	144	#6, #22
7	Dropout + Conv. + LReLU	48	22.5%	24	Dropout + Conv. + LReLU	96	22.5%
8	Max Pooling	48	2×2	25	Dropout + Conv. + LReLU	96	22.5%
9	Dropout + Conv. + LReLU	48	30%	26	Upsampling	96	2X
10	Max Pooling	48	2×2	27	Concatenation	144	#4, #26
11	Dropout + Conv. + LReLU	48	37.5%	28	Dropout + Conv. + LReLU	96	15%
12	Max Pooling	48	2×2	29	Dropout + Conv. + LReLU	96	15%
13	Dropout + Conv. + LReLU	48	45%	30	Upsampling	96	2X
	DECODER			31	Concatenation	96 + C	#1, #30
14	Upsampling	48	2X	32	Dropout + Conv. + LReLU	64	7.5%
15	Concatenation	96	#10, #14	33	Dropout + Conv. + LReLU	32	7.5%
16	Dropout + Conv. + LReLU	96	37.5%	34	Dropout + Conv. + Sigmoid	C	7.5%

Running time. On an RTX 2080Ti GPU with auto mixed precision and parallel computation, our PyTorch code on average takes about 0.57 minutes for an 128×128 image, 1.84 minutes for a 256×256 image, 3.31 minutes for a 384×384 image, and 8.83 minutes for a 512×512 image, under a 15×15 blur kernel and noise level $\sigma = 5\%$; and it takes about 0.63 minutes for a 128×128 image, 2.77 minutes for a 256×256 image, 5.96 minutes for a 384×384 image, and 10.23 minutes for a 512×512 image with a 27×27 blur kernel and noise level $\sigma = 5\%$. Our code infers 320 times per second for a 256×256 image on average.

Methods included for comparison. The following NID methods are selected for comparison in the experiments: IBM3D (Danielyan et al, 2011), FID (Krishnan and Fergus, 2009), EPLL (Zoran and Weiss, 2011), DIP (Vedaldi et al, 2020), DIKP (Wang et al, 2019), DDec (Heckel, 2019), BPDIP (Zukerman et al, 2020), FCNN

(Zhang et al, 2017a), IRCNN (Zhang et al, 2017c), FDN (Kruse et al, 2017), DMSP (Bigdeli et al, 2017), DPDNN (Dong et al, 2019), RGDN (Gong et al, 2020), VEM (Nan et al, 2020), CPCR (Eboli et al, 2020) and DWDN (Dong et al, 2021). The IBM3D, FID and EPLL are representative non-learning-based methods. The DIP, DIKP, DDec and BPDIP are DIP-based unsupervised deep learning methods with different strategies for addressing the solution ambiguity when training the DNN using (2). The FCNN, IRCNN, FDN, DMSP, DPDNN, RGDN,VEM, CPCR and DWDN are supervised deep learning-based methods with different DNN structures.

Through all experiments, whenever possible, we directly quote the results reported in the literature. Otherwise, we obtain the results using the trained models provided by the authors. If only the code is available, we made efforts on training the models for optimal performance. See below for more details: *(i)* The results of IBM3D are obtained using the updated code published on the authors' website. *(ii)* The DIP uses an over-parameterized DNN with the same architecture as the authors' implementation for super-resolution. Early stopping is used when solving (2). We made the effort to find its optimal iteration number for different noise levels: 10000 for $\sigma \leq 0.01$, 6000 for $\sigma = 0.03$, and 4000 for $\sigma = 0.05$. *(iii)* The DDec uses an under-parameterized DNN for solving (2), whose architecture is the same as the one from the authors' implementation for super-resolution. *(iv)* The DIKP combines DIP with TV regularization, whose network architecture is the same as the authors' implementation. One of its key hyper-parameters is the weight α for TV regularization term. Instead of using its original setting, $\alpha = 2 \times 10^{-2}$, we make it adaptive to noise level $\alpha = \sigma/10$ to have better performance for different noise levels. *(v)* The DPDNN only provides pretrained models on several settings. We follow its original work and use its training code to obtain a model on each motion blur kernel and each noise level. *(vi)* The DWDN only provides a model pre-trained on a much larger dataset. For fairness, we use its training code to retrain a model on the same dataset as other supervised learning-based methods. *(vii)* For other supervised learning-based methods, we use the pre-trained models released by their authors to obtain the results. *(viii)* For other non-learning-based methods and unsupervised learning-based methods, the official implementations from their authors are used in the experiments.

Evaluation protocol. The performance of NID is measured by the average PSNR and SSIM of the deblurred images on the test set. It is noted that many existing works have different schemes to cut off boundary pixels during calculating PSNR and SSIM to focus on the deblurring effect on the main region. For a fair comparison, we follow the standard boundary cutting scheme used in (Zhang et al, 2017a; Kruse et al, 2017; Bigdeli et al, 2017) for evaluating the results from all methods.

4.2 NID with Accurate Kernels

4.2.1 Motion Blur Kernels

The evaluation on motion deblurring is conducted on three datasets: Set12 (Zhang et al, 2017b), Levin *et al.*'s dataset (Levin et al, 2011) and Sun *et al.*'s dataset (Sun et al, 2013). There is a lot of diversity among the images in these three datasets. Following standard protocol, the eight motion-blur kernels from (Levin et al, 2011), together with the additive Gaussian white noise (AWGN) of level $\sigma = 1\%, 3\%, 5\%$, are used to generate the degraded images for test. The EdgeTapper (Schmidt and Roth, 2014; Zhang et al, 2017a) is called for simulating practical boundary conditions in blurring. See Table 2 for quantitative comparison and Fig. 6 for visual inspection.

In the quantitative comparison, the proposed method outperforms IBM3D in all settings, which shows the advantage of deep learning over hand-crafted regularization. Compared with the other four unsupervised learning methods, ours outperforms them by a large margin (>0.77dB). Such results indicate that the strategies used in these methods for regularizing network training, including early-stopping in DIP/BPDIP, TV regularization in DIKP, and under-parameterization in DDec, are not effective on addressing the solution ambiguity and measurement noise. In contrast, based on the model with uncertainty and the proposed self-supervised loss, the proposed method can effectively alleviate the solution ambiguity and suppress the measurement noise. Its quantitative performance advantage is also consistent with its advantage on visual quality.

Table 2 PSNR(dB) (odd rows) and SSIM (even rows) results of NID in the case of motion kernels and AWGN. The best results in each category of methods are underlined. The best results in all compared methods are boldfaced. The column "Margin" denotes the performance gap between the proposed method and the best supervised learning-based method, where a positive value implies the proposed method is better while the negative one implies the opposite.

Dataset	σ	Non-Learning / Unsupervised Learning								Supervised Learning									Margin
		IBM3D	FID	EPLL	DIP	DIKP	DDec	BPDIP	Ours	FCNN	IRCNN	FDN	DMSP	DPDNN	RGDN	VEM	CPCR	DWDN	
Set12	1%	29.66	27.68	30.70	27.21	28.44	28.91	28.86	**31.56**	29.76	30.38	31.02	31.20	30.12	31.02	30.97	27.81	30.63	0.36
		0.851	0.796	0.876	0.784	0.812	0.841	0.830	**0.886**	0.870	0.874	0.876	0.870	0.862	0.885	0.882	0.839	0.877	0.001
	3%	27.09	25.19	26.68	26.42	26.69	27.32	26.09	28.09	27.19	27.94	27.91	27.94	27.79	28.17	**28.26**	25.20	27.32	-0.17
		0.787	0.671	0.778	0.742	0.740	0.792	0.728	0.808	0.787	0.814	0.802	0.787	0.813	**0.818**	0.817	0.748	0.799	-0.010
	5%	25.65	23.18	24.75	24.06	23.81	25.03	24.67	26.56	25.88	26.50	26.46	26.44	26.44	26.49	**26.78**	24.85	26.53	-0.22
		0.744	0.519	0.718	0.585	0.562	0.693	0.664	0.770	0.754	0.756	0.757	0.755	0.756	0.754	**0.772**	0.738	**0.772**	-0.002
Levin *et al.*'s	1%	30.11	27.64	32.01	29.58	31.14	27.39	30.63	**33.86**	30.19	31.10	32.60	32.60	31.14	33.57	31.64	28.11	32.19	0.29
		0.871	0.841	0.912	0.852	0.887	0.790	0.884	**0.932**	0.891	0.886	0.897	0.904	0.891	0.925	0.905	0.845	0.920	0.007
	3%	28.41	26.93	28.32	28.10	28.29	26.22	27.54	**29.89**	28.03	28.98	29.31	29.31	28.94	29.75	29.32	26.16	29.45	0.14
		0.832	0.732	0.833	0.804	0.794	0.725	0.785	**0.863**	0.833	0.848	0.829	0.833	0.854	0.852	0.856	0.778	**0.863**	0.000
	5%	27.02	24.54	26.13	24.27	23.87	24.72	25.84	**28.09**	26.83	27.57	27.46	27.83	27.56	27.22	27.78	25.54	27.86	0.23
		0.793	0.582	0.770	0.600	0.559	0.652	0.716	0.816	0.786	0.803	0.804	0.813	0.814	0.781	0.813	0.761	**0.826**	-0.010
Sun *et al.*'s	1%	31.42	29.70	32.04	27.17	26.48	27.08	27.47	32.09	31.61	31.81	**32.22**	32.00	31.25	31.16	32.20	29.71	31.75	-0.13
		0.866	0.809	0.878	0.710	0.694	0.698	0.729	0.875	0.880	0.876	**0.893**	0.871	0.851	0.873	0.888	0.859	0.880	-0.018
	3%	27.87	26.83	28.17	26.32	26.32	26.58	26.84	28.97	28.72	28.94	28.89	28.63	29.08	28.54	**29.23**	26.62	28.79	-0.26
		0.753	0.665	0.758	0.687	0.684	0.680	0.706	0.798	0.778	0.804	0.781	0.768	**0.802**	0.797	0.801	0.733	0.785	-0.004
	5%	26.46	24.40	26.57	26.05	26.15	25.83	25.70	27.75	27.47	27.51	27.62	27.47	27.91	27.35	**27.93**	26.39	27.61	-0.18
		0.699	0.493	0.699	0.667	0.678	0.638	0.655	0.750	0.733	0.735	0.732	0.735	**0.757**	0.737	**0.757**	0.730	0.741	-0.007

Compared to the DNNs trained with supervision, surprisingly, even without accessing ground-truth images for training, the proposed method is very competitive to its supervised counterparts. In particular, on Set12 and Levin *et al.*'s dataset, the proposed method outperforms the DNNs trained with supervision in terms of PSNR (>0.29dB) when $\sigma = 1\%$. For the higher noise level $\sigma = 3\%, 5\%$, the proposed method achieved the highest PSNR on Levin *et al.*'s dataset.

4.2.2 Non-Motion Blur Kernels

The performance of the proposed method in NID with non-motion blur kernels is evaluated using the six scenarios designed in (Danielyan et al, 2011), which contain five non-motion blur kernels and the AWGN of different noise levels. For the supervised learning-based methods, we do not retrain their models due to the diversity of the test kernels, but use their models trained on motion kernels to test their generalization on the kernels unseen in training data. For FCNN and IRCNN whose models are trained on individual noise levels, we take the optimal results of their trained models across different noise levels. The DPDNN is not included for comparison, as it is individually trained on each kernel in motion deblurring and thus not suitable to this experiment. Instead, the NNet (Gilton et al, 2020) trained on defocus kernels is included for comparison. Note that NNet is not included in the previous experiment due to its noticeably inferior performance. The evaluation is done on Set12 (Zhang et al, 2017b) and Sun *et al.*'s dataset (Sun et al, 2013). See Table 3 for quantitative comparison and Fig. 7 for visual inspection.

The proposed method is overall the best performer. *(i)* It outperforms all non-learning-based and unsupervised learning-based methods by a large margin (>0.69dB on Set12 and >0.31dB on Sun *et al.*'s dataset). *(ii)* The performance of the models trained with supervision is not as good as that in the previous experiment on motion deblurring, which implies that they do not generalize well when there exist noticeable variations of blur kernels between training and test data. Indeed, this is one motivation for studying dataset-free learning for NID, which is independent of the type of blur kernels.

Table 3 PSNR(dB) (odd rows) and SSIM (even rows) results of NID in six scenarios from (Danielyan et al, 2011). The best results in each category of methods are underlined. The best results in all compared methods are boldfaced. The column "Margin" denotes the performance gap between the proposed method and the best supervised learning-based method, where a positive value implies the proposed method is better while the negative one implies the opposite.

Dataset	Non-Learning / Unsupervised Learning								Supervised Learning								Margin
	BM3D	FID	EPLL	DIP	DIKP	DDec	BPDIP	Ours	FCNN	IRCNN	FDN	DMSP	NNet	RGDN	VEM	CPCR	DWDN	
Set12	30.25	27.74	30.41	27.83	27.81	28.14	28.45	**31.10**	29.49	30.96	29.27	30.44	27.58	29.29	30.35	29.93	30.14	0.14
	0.852	0.766	0.860	0.788	0.793	0.808	0.815	**0.881**	0.871	0.870	0.869	0.845	0.770	0.846	0.853	0.856	0.870	0.010
Sun	31.59	29.25	32.20	28.76	30.11	29.17	29.52	**32.51**	30.91	32.07	29.24	30.78	29.00	30.89	31.18	31.22	32.29	0.22
	0.870	0.768	0.880	0.749	0.813	0.772	0.792	**0.894**	0.848	0.881	0.754	0.825	0.744	0.838	0.842	0.854	0.888	0.006

4.2.3 Poisson Measurement Noise

To see how well our method performs on other types of noise, this experiment is conducted on Set12 (Zhang et al, 2017b) and Sun *et al.*'s dataset (Sun et al, 2013), with the real motion blur kernels from (Levin et al, 2011) and with Poisson noise of different peak values. Our method is mainly compared to the aforementioned unsupervised learning-based methods. Regarding the supervised ones, only the previous trained models of FDN, DMSP, RGDN, VEM, CPCR and DWDN are included for comparison, as they are designed to be noise-blind. In addition, two non-learning-based methods specifically designed for handling Poisson noise are also included in the comparison: VST-BM3D (Azzari and Foi, 2016) and RWL2 (Li et al, 2015). In these experiments, the noise level is estimated using the estimator (Chen et al, 2015), if the method requires it (*e.g.*, DIKP, FDN). See Table 4 for quantitative comparison and Fig. 8 for visual inspection. The proposed method consistently performs noticeably better than the other unsupervised learning-based methods (>1.47dB on Set12 and >1.24dB on Sun *et al.*'s dataset), and it also performs competitively against the best performers of the supervised learning-based methods.

4.3 NID in Real-World Cases

4.3.1 Real-world Degradation

In real-world scenarios, motion blur is introduced in the very early stage of the full pipeline of image acquisition, which includes other consequent processes such as quantization and gamma correction. As a result, the degradation model of a motion-blurred image cannot be simply expressed as a convolution. In addition, sometimes there are also other degradation effects presented, *e.g.*, pixel saturation. The robustness of the proposed method to such real-world degradations is evaluated on *(i)* the saturation category from Lai *et al.*'s dataset (Lai et al, 2016) which includes 20 saturated images generated from four motion kernels; and *(ii)* Anger *et al.*'s dataset (Anger et al, 2018) which contains eight images generated from eight motion kernels with gamma correction and quantization.

The proposed method is mainly compared to the aforementioned unsupervised learning-based methods and some selected supervised ones. We also include the MRD method (Anger et al, 2018) for comparison, which is specifically designed for such kinds of real-world degradation. In the experiment on Anger *et al.*'s dataset, following (Anger et al, 2018), we consider the inclusion of gamma correction in the image formation model: $\boldsymbol{y} = g_\gamma(\boldsymbol{k} * \boldsymbol{x}) + \boldsymbol{n}$ where $(g_\gamma(\boldsymbol{z}))_j = \boldsymbol{z}_j^{\frac{1}{\gamma}}, \forall j$, and modify the loss functions of all unsupervised learning-based methods accordingly using the gamma values provided by the dataset. For instance, the loss function in the proposed method becomes $\mathcal{C} := \|g_\gamma(f_{\boldsymbol{\theta} \odot \boldsymbol{b}}(\widehat{\boldsymbol{y}}_\ell) * \boldsymbol{k}) - \boldsymbol{y}\|^2_{\boldsymbol{m}_\ell}$. See Table 5 for quantitative comparison and Fig. 9 for visual inspection. The proposed method is the best performer among all compared methods. Particularly, it performs noticeably better than other unsupervised methods (>5.37dB on Lai *et al.*'s dataset and >0.96dB on Anger *et al.*'s dataset).

It is worth mentioning that many modern cameras use precomputed functions instead of gamma correction. When such functions accounting for the full pipeline of image acquisition are known, it would be beneficial to include them into g.

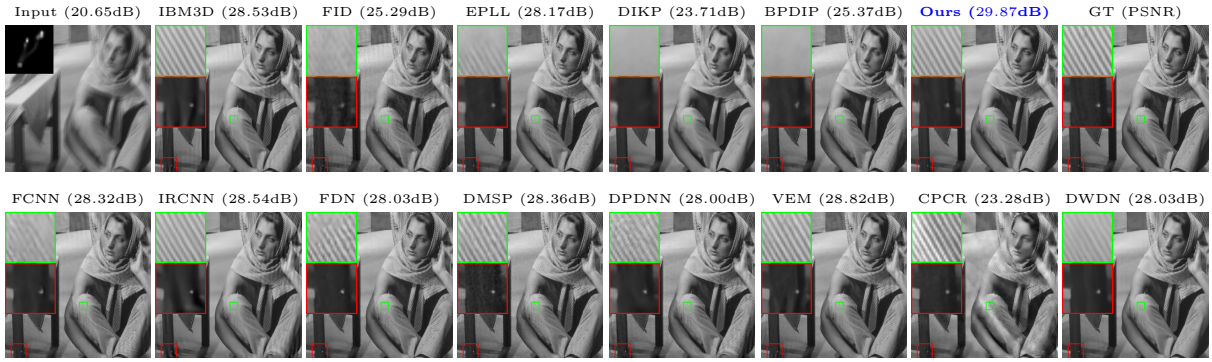


Fig. 6 Visual results of NID using the 8th motion kernel from (Levin et al, 2011) in the presence of AWGN with $\sigma = 1\%$.

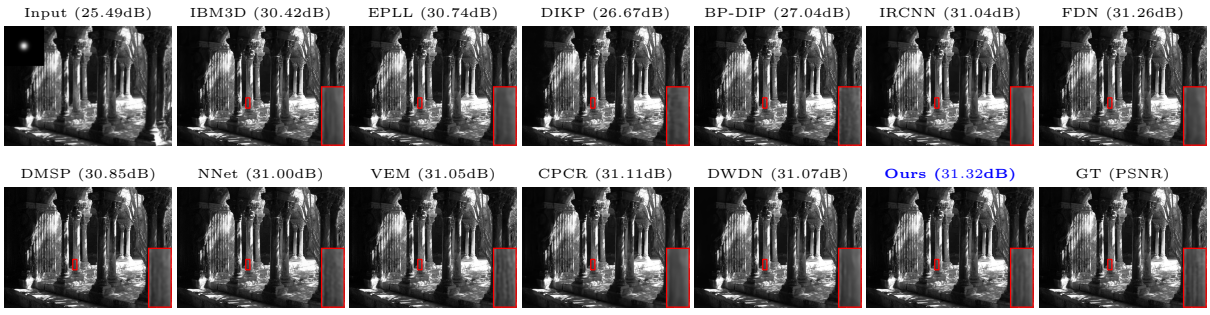


Fig. 7 Visual results of NID on the 5th scenario from (Danielyan et al, 2011).

Table 4 PSNR(dB) (odd rows) and SSIM (even rows) results of NID in the case of motion kernels and Poisson noise. The best results in each category of methods are underlined. The best results in all compared methods are boldfaced. The column "Margin" denotes the performance gap between the proposed method and the best supervised learning-based method, where a positive value implies the proposed method is better while the negative one implies the opposite.

Dataset	Peak	Non-Learning / Unsupervised Learning							Supervised Learning						
		VST-BM3D	RWL2	DIP	DIKP	DDec	BPDIP	Ours	FDN	DMSP	RGDN	VEM	CPCR	DWDN	Margin
Set12	128	24.45	24.83	22.28	23.37	23.75	24.06	25.84	25.71	25.56	25.49	25.73	19.46	**26.00**	-0.16
		0.690	0.712	0.514	0.618	0.603	0.625	**0.774**	0.709	0.725	0.691	0.721	0.416	0.757	0.017
	256	25.25	25.70	24.84	24.71	25.04	24.95	26.84	26.74	26.53	26.58	26.69	21.80	**26.90**	-0.06
		0.723	0.742	0.629	0.633	0.665	0.677	**0.808**	0.767	0.764	0.777	0.735	0.537	0.782	0.026
	512	25.58	26.00	26.29	26.45	26.00	25.97	**27.92**	27.74	27.41	27.66	27.79	24.81	27.79	0.13
		0.738	0.746	0.733	0.757	0.721	0.718	**0.835**	0.798	0.768	0.823	0.782	0.688	0.808	0.012
	1024	25.65	26.11	26.84	27.12	26.65	26.75	**29.01**	28.74	28.35	28.66	28.84	26.12	28.71	0.17
		0.743	0.754	0.758	0.767	0.760	0.761	**0.855**	0.821	0.783	0.845	0.816	0.724	0.832	0.010
Sun	128	26.66	26.36	25.24	25.55	24.92	23.19	**26.98**	25.29	26.46	26.53	26.74	20.43	26.96	0.02
		0.714	0.689	0.679	0.688	0.671	0.473	**0.734**	0.684	0.710	0.703	0.708	0.403	0.729	0.005
	256	27.17	27.13	26.43	26.66	25.79	25.23	27.90	27.74	27.04	27.28	27.81	23.00	**28.05**	-0.15
		0.729	0.722	0.717	0.720	0.691	0.662	**0.767**	0.744	0.722	0.726	0.743	0.541	0.766	0.001
	512	27.41	27.33	26.91	27.35	26.88	26.73	28.87	28.64	27.65	28.01	28.84	25.95	**29.03**	-0.16
		0.739	0.717	0.725	0.732	0.726	0.717	**0.799**	0.778	0.729	0.745	0.784	0.702	**0.799**	0.000
	1024	27.44	27.40	27.85	28.41	27.66	28.06	29.81	29.62	28.56	29.17	29.79	27.77	**30.02**	-0.21
		0.742	0.724	0.750	0.761	0.741	0.758	0.829	0.805	0.754	0.792	0.812	0.797	**0.832**	-0.003

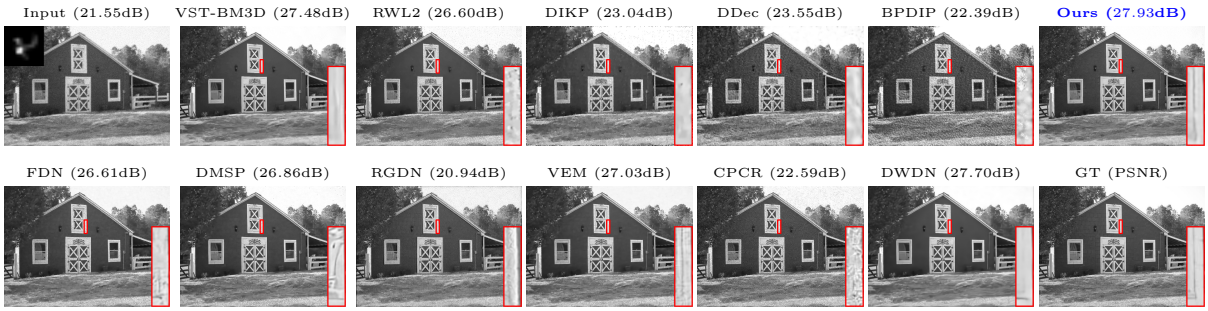


Fig. 8 Visual results of NID using 5th motion kernel from (Levin et al, 2011) in the presence of Poisson noise with peak 128.

Table 5 PSNR(dB) (odd rows) and SSIM (even rows) results of NID on the images with saturated pixels from Lai *et al.*'s dataset and the images with realistic degradations from Anger *et al.*'s dataset. The best results in each category of methods are underlined. The best results in all compared methods are boldfaced. The column "Margin" denotes the performance gap between the proposed method and the best supervised learning-based method, where a positive value implies the proposed method is better while the negative one implies the opposite.

Dataset	Non-Learning / Unsupervised Learning									Supervised Learning								Margin
	IBM3D	EPLL	FID	MRD	DIP	DIKP	DDec	BPDIP	Ours	FCNN	IRCNN	FDN	DMSP	RGDN	VEM	CPCR	DWDN	
Lai *et al.*'s	14.77	17.31	16.24	15.69	12.00	12.21	11.87	11.66	**17.58**	14.84	15.01	7.19	16.36	17.34	14.56	16.34	17.54	0.04
	0.62	0.74	0.69	0.64	0.57	0.58	0.57	0.56	**0.75**	0.61	0.64	0.26	0.75	0.69	0.59	0.67	**0.75**	0.00
Anger *et al.*'s	22.78	23.82	22.44	28.04	27.46	26.59	18.09	23.89	**28.42**	23.58	23.30	24.29	24.12	17.33	23.70	23.07	24.69	0.38
	0.82	0.85	0.80	0.92	0.90	0.89	0.56	0.83	**0.93**	0.85	0.83	0.87	0.86	0.57	0.85	0.84	0.88	0.01

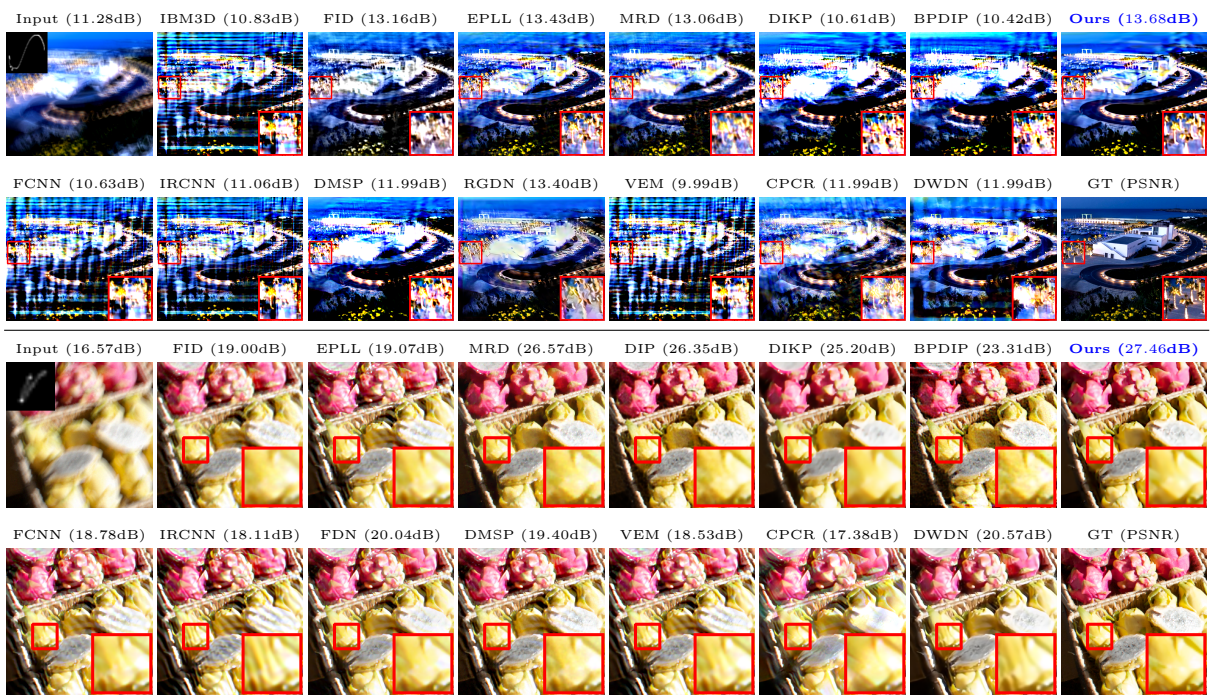


Fig. 9 Visual results of NID with inaccurate kernels on the sample images from Lai *et al.*'s dataset and Anger *et al.*'s real dataset, respectively.

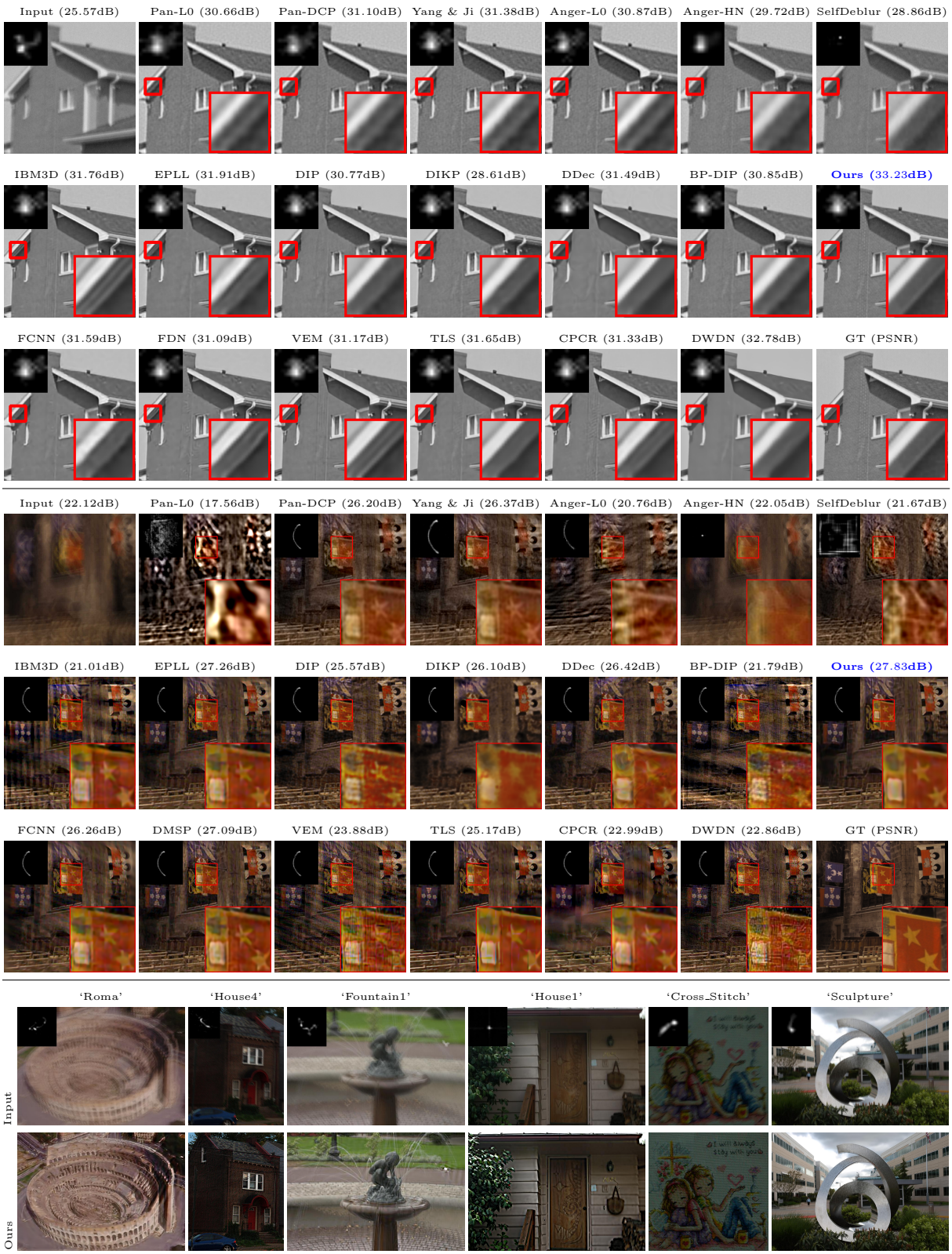


Fig. 10 Visual results of NID with inaccurate kernels, which are from Set12, Kohler *et al.*'s dataset and Lai *et al.*'s real dataset, respectively.

Table 6 PSNR(dB) (odd rows) and SSIM (even rows) results of NID with inaccurate kernels. The upper part contains the blind deconvolution methods, the middle part contains the non-learning-based or unsupervised learning-based NID methods, and the bottom part contains the supervised learning-based NID methods. The best results in each category of methods are underlined. The best results in all compared methods are boldfaced. The column "Margin" denotes the performance gap between the proposed method and the best one in corresponding category, where a positive value implies the proposed method is better while the negative one implies the opposite.

Dataset	Pan-L0	Pan-DCP	Yang & Ji	Anger-L0	Anger-HN	SelfDeblur	ASN	Cho *et al.*	Margin
Set12	25.43/0.77	26.62/0.80	26.05/0.78	25.76/0.77	24.92/0.75	24.23/0.73	26.09/0.83	20.83/0.62	2.17/0.09
Lai *et al.*'s	18.54/0.62	19.16/0.62	19.42/0.68	18.65/0.54	18.36/0.50	21.13/0.73	20.22/0.65	17.60/0.46	0.81/0.04
Kohler *et al.*'s	26.90/0.85	28.85/0.91	28.32/0.89	27.96/0.86	27.21/0.83	23.41/0.76	28.94/0.89	24.60/0.76	0.58/0.02
Dataset	IBM3D	EPLL	FID	DIP	DIKP	DDec	BPDIP	Ours	Margin
Set12	27.77/0.86	27.21/0.85	26.05/0.78	26.35/0.81	26.64/0.82	27.25/0.84	25.91/0.81	**28.79**/**0.89**	1.02/0.03
Lai *et al.*'s	19.36/0.67	21.81/0.76	21.02/0.72	21.18/0.72	21.42/0.73	21.13/0.72	20.06/0.69	21.94/0.77	0.13/0.01
Kohler *et al.*'s	23.65/0.84	28.90/0.91	28.85/0.91	27.44/0.88	28.51/0.90	27.71/0.88	24.11/0.82	**29.52**/**0.93**	0.62/0.02
Dataset	FCNN	IRCNN	FDN	DMSP	VEM	TLS	CPCR	DWDN	Margin
Set12	27.66/0.86	27.12/0.85	27.81/0.87	27.48/0.84	27.73/0.87	28.37/0.88	26.31/0.83	27.01/0.85	0.42/0.01
Lai *et al.*'s	21.66/0.76	21.22/0.73	17.40/0.55	**22.15**/**0.78**	21.11/0.72	20.57/0.73	20.87/0.73	21.76/0.77	-0.21/-0.01
Kohler *et al.*'s	28.03/0.90	26.92/0.88	26.84/0.86	28.19/0.90	25.89/0.86	29.05/**0.93**	26.99/0.89	24.39/0.86	0.47/0.00

4.3.2 Inaccurate Kernels

The proposed method is also tested with inaccurate kernels to evaluate its robustness to kernel error. Four datasets are used for evaluation: Set12 (Zhang et al, 2017b) with 96 images generated by the eight motion blur kernels from (Levin et al, 2011) and noise level $\sigma = 1\%$, Lai *et al.*'s synthetic dataset (Lai et al, 2016) with 100 images generated by the four motion blur kernels from (Lai et al, 2016) and noise level $\sigma = 1\%$, Kohler *et al.*'s dataset (Köhler et al, 2012) with 48 blurry images generated from twelve motion blur kernels with certain degree of spatial variation, and Lai *et al.*'s real dataset (Lai et al, 2016) with 100 motion blurred images. The inaccurate kernels are obtained by applying some SOTA motion blur kernel estimation method, *i.e.*, (Yang and Ji, 2019) for Set12, (Ren et al, 2020) for Lai *et al.*'s synthetic dataset, and (Pan et al, 2016) for both Kohler *et al.*'s dataset and Lai *et al.*'s real dataset. For Kohler *et al.*'s dataset, we use its official benchmark code to calculate the PSNR and SSIM. For the other three datasets, following the standard protocol (Levin et al, 2011; Ren et al, 2020), we first align the output images to the sharp images with sub-pixel shift and then cut off the boundary pixels for calculating the PSNR and SSIM. In this experiment, the iteration number of the proposed method is fixed to 2.5×10^4.

The representative methods mentioned above are included for comparison, except DPDNN as its implementation assumes that the same kernel is used in both training and test. Instead, we include TLS (Nan and Ji, 2020), a very recent method specifically designed for NID with inaccurate kernels. We use its published trained model for the test with our setting. In addition, some blind deblurring methods also included for comparison, including Pan-L0 (Pan et al, 2014), Pan-DCP (Pan et al, 2016), Yang & Ji (Yang and Ji, 2019), Anger-L0 (Anger et al, 2019b), Anger-HN (Anger et al, 2019a), SelfDeblur (Ren et al, 2020), ASN (Kaufman and Fattal, 2020) and Cho *et al.* (Cho et al, 2021). See Table 6 for quantitative comparison and Fig. 10 for visual inspection. The proposed method outperforms all other methods, except that it performs slightly worse than DMSP on Lai *et al.*'s dataset with less than 0.2dB PSNR gap.

Table 7 Empirical analysis on the robustness improvement to kernel error brought by the standard dropout scheme and the proposed SA droput scheme. The values in the brackets indicate the drop in PSNR.

Kernels	w/o SA	w/o Dropout	Ours
Perfect	31.51 (↓ 0.05)	29.73 (↓ 1.83)	31.56
Inaccurate	28.28 (↓ 0.51)	26.24 (↓ 2.55)	28.79

To investigate how much the dropout-based model uncertainty contributes to the robustness of

The multi-scale SSIM is used in the benchmark of Kohler *et al.*'s dataset. For simplicity, we also call it SSIM in the tables.

the proposed method to kernel error, we conduct an ablation study by comparing the proposed one against the same network with the SA dropout being replaced by the standard dropout, as well as the same network with dropout being disabled. See Table 7 for the results on Set 12 in both cases of perfect kernels and inaccurate kernels. The first observation is that the model uncertainty introduced by dropout improves the robustness to kernel error, as the PSNR gain from introducing dropout on inaccurate kernels is noticeably (around 0.72dB) larger than that on perfect kernels. The second observation is that the model uncertainty implemented using the proposed SA dropout brings further improvement to the robustness over using the standard dropout, as the PSNR gain from the SA dropout on inaccurate kernels is noticeably (around 0.46dB) than that on perfect kernels. Particularly, the PSNR gain from the SA dropout is very minor (0.05dB) for the case of perfect kernels. Such results have clearly verified that model uncertainty benefits handling kernel error, for which the SA dropout is more effective than the standard one.

4.4 Ablation Study

4.4.1 SA Dropout vs. Regular Dropout

In the previous experiment, we have analyzed the benefit of SA dropout for handling kernel error. In this ablation study, we further study the benefit of the proposed SA dropout over the regular (standard) one in other NID settings. See Table 8 for the results. The spatial adaption in SA dropout leads to improvement in various settings. Overall, the larger amount of the measurement noise is, the bigger improvement it can bring. We can also see that, the PSNR gain obtained by the SA dropout for Poisson noise (about $0.40-0.47$dB), is greater than that for AWGN (about $0.04-0.16$dB).

An illustration is given in Fig. 11 to give some insight, where we show the pixel-wise variance maps of 100 dropout inferences from the models trained on two blurred images with different noise levels. The higher the blur degree a region has, the higher the variance (uncertainty) of the inferences about the region has. These results have further demonstrated the effectiveness of the proposed SA dropout scheme.

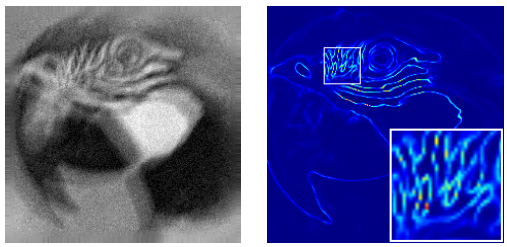


Fig. 11 Variance map (right) of 100 dropout inferences from our dropout-trained model on a blurred image (left). Brighter color indicates higher variance.

Table 8 PSNR(dB) results in ablation studies on Sun *et al.*'s dataset with accurate motion kernels and AWGN ($\sigma = 5\%$).

Dropout	Noise Level σ			Peak Value			
	1%	3%	5%	128	256	512	1024
Standard	31.51	28.00	26.40	25.37	26.43	27.52	28.61
SA	31.56	28.09	26.56	25.84	26.84	27.92	29.01
Margin	0.04	0.09	0.16	0.47	0.41	0.40	0.40

4.4.2 Contribution of Each Component

The ablation study conducted in this section is to evaluate how much the components included in the proposed method contribute to the performance improvement, which include: *(i)* the contribution from using dropout in training; *(ii)* the contribution from using dropout-based aggregation in testing; *(iii)* the contribution from using pixel replacement for the input seeds; *(iv)* the contribution from using the masking in the self-supervised loss. Such a study is conducted on Sun *et al.*'s dataset in NID with accurate motion kernels, using the following baselines: *(i)* w/o Dropout: disabling dropout training and dropout inference; *(ii)* w/o Aggregat.: using the model from dropout training and $\mathbb{E}_{\boldsymbol{b}}[\boldsymbol{\theta}^* \odot \boldsymbol{b}]$ for prediction, without dropout inference and averaging. *(iii)* w/o Replace. : using non-random $\boldsymbol{y}$ without replacement as input (the corresponding loss is the squared ℓ_2 loss w/o masking as no pixels are replaced) *(iv)* w/o Masking: using standard squared ℓ_2 reconstruction without masking in (8).

See Table 9 for the results on motion deblurring with AWGN of noise level $\sigma = 5\%$. Clearly, each component makes substantial contribution to the performance. Among them, the dropout-based aggregation during test contributes the most (0.93dB for "Ours" over "w/o Aggregat.") and the pixel replacement strategy contributes the least yet noticeably (0.14dB for "w/o Masking."

over "w/o Replace."). Without aggregation, the dropout training only serves as a regularization which yields some improvement (0.31dB for "w/o Aggregat." over "w/o Dropout").

Table 9 Quantitative results in ablation studies on Sun *et al.*'s dataset with accurate motion kernels and AWGN ($\sigma = 5\%$).

w/o	Dropout	Aggregat.	Replace.	Masking	Ours
PSNR(dB)	26.51	26.82	27.34	27.48	27.75
SSIM	0.700	0.704	0.733	0.741	0.750

4.4.3 PSNR w.r.t. Inference Number

See Fig. 12 for the graph of performance impact caused by the number of dropout inferences. Both the PSNR and SSIM increase very fast when more predictions are used for aggregation in the range from 1 to 20, which indicates that there is sufficient independence among these instances to improve the prediction on the latent image via average. After 20 inferences, the gain is much less, and it saturates with more than 50 inferences.

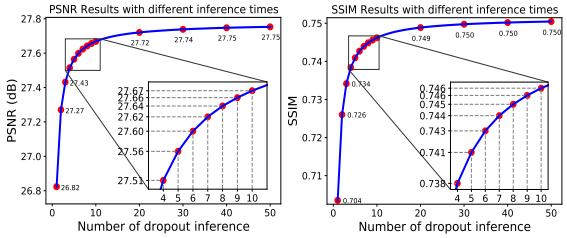


Fig. 12 Average PSNR/SSIM versus number of inferences, calculated in the experiment on Sun *et al.*'s dataset with accurate motion kernels and AWGN ($\sigma = 5\%$).

4.5 Limitation Analysis

While exhibiting SOTA performance among the non-learning-based and unsupervised learning-based methods in previous experiments, the proposed approach has limitations in comparison to its supervised counterparts. As discussed in Remark 3, Proposition 1 is not applicable in the presence of kernel estimation error. When such error is large, the performance of the DNN trained by the proposed loss (8) may not be as good as some supervised learning-based methods. See Fig. 13 for such a case where the performance of the proposed approach is better than TLS (Nan and Ji, 2020), but is worse than ASN (Kaufman and Fattal, 2020), a supervised learning-based kernel-blind method without explicit utilization of the convolution model of blurring.

Another limitation is the higher computational cost for testing in comparison to supervised learning methods, as supervised learning allows one to use a pre-trained model to process all test data, rather than perform training on test data.

Fig. 13 Failure case of the proposed method when deblurring an image from Lai *et al.*'s synthetic dataset in the presence of large kernel error.

5 Conclusion

While deep learning over a large dataset is an effective approach to NID, its practicability is limited in the scenarios where training samples are hard to collect. In this paper, based on untrained DNNs and model uncertainty, we proposed a dataset-free deep learning approach for NID which has no dependence on external training samples. The main idea is to introduce model uncertainty implemented by a specific SA dropout scheme to handle the solution ambiguity and a self-supervised loss to deal with the measurement noise. The prediction is obtained by averaging multiple outputs of the DNN with model uncertainty, which can be viewed as an approximate MMSE estimator of the problem in Bayesian inference. The proposed approach achieved noticeable performance improvement over existing non-learning-based methods and unsupervised learning-based methods in extensive

experiments, and it also competed well against recent supervised deep learning-based methods.

The techniques developed in this paper have potential applications to other image recovery problems. The proposed self-supervised loss can be extended to handle the measurement noise in other imaging models. The SA-dropout-based scheme that leverages model uncertainty to address solution ambiguity can be applied for further improvement when using untrained DNNs to solve the inverse problems where the observations only provide partial information of the latent image, *e.g.*, image super-resolution and compressed sensing. We will investigate such extensions in the future.

There are two limitations in the proposed approach. One is its weak robustness to large kernel error. The other is its higher processing time than supervised learning-based methods, which is a common weakness of existing dataset-free learning methods. In the future, we would like to study how to introduce specific mechanisms to handle large kernel errors better. Also, we would like to study how to improve the computational efficiency. One possible approach we will investigate is to quickly adapt some pre-trained model for the self-supervised learning process.

A Proof of Proposition 1

First, we rewrite the loss function as follows.

$$\begin{aligned}&\sum_{\ell}\|\boldsymbol{k}*f_{\boldsymbol{\beta}}(\widehat{\boldsymbol{y}}_\ell)-\boldsymbol{y}\|^2_{\boldsymbol{m}_\ell}\\ =&\sum_{\ell}\|\boldsymbol{k}*f_{\boldsymbol{\beta}}(\widehat{\boldsymbol{y}}_\ell)-\boldsymbol{k}*\boldsymbol{x}\|^2_{\boldsymbol{m}_\ell}+\sum_{\ell}\|\boldsymbol{n}\|^2_{\boldsymbol{m}_\ell}\\ &-2\boldsymbol{n}^\top\big(\sum_{\ell}(\boldsymbol{1}-\boldsymbol{m}_\ell)\odot(\boldsymbol{k}*f_{\boldsymbol{\beta}}(\widehat{\boldsymbol{y}}_\ell)-\boldsymbol{k}*\boldsymbol{x})\big).\end{aligned}\tag{16}$$

The expectation of the second term is given by

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{n}}\big[\sum_{\ell}\|\boldsymbol{n}\|^2_{\boldsymbol{m}_\ell}\big]=\mathbb{E}_{\boldsymbol{n}}\big[\sum_{\ell}\|(\boldsymbol{1}-\boldsymbol{m}_\ell)\odot\boldsymbol{n}\|_2^2\big]\\ =&\sum_{\ell}\|(\boldsymbol{1}-\boldsymbol{m}_\ell)\odot\boldsymbol{\sigma}\|_2^2=\sum_{\ell}\|\boldsymbol{\sigma}\|^2_{\boldsymbol{m}_\ell}.\end{aligned}\tag{17}$$

Regarding the last term, for simplicity we define

$$\begin{aligned}\boldsymbol{r}&=\sum_{\ell}(\boldsymbol{1}-\boldsymbol{m}_\ell)\odot(\boldsymbol{k}*f_{\boldsymbol{\beta}}(\widehat{\boldsymbol{y}}_\ell)-\boldsymbol{k}*\boldsymbol{x})\\ &=\sum_{\ell}(\boldsymbol{1}-\boldsymbol{m}_\ell)\odot(\boldsymbol{k}*f_{\boldsymbol{\beta}}(\boldsymbol{m}_\ell\odot(\boldsymbol{k}*\boldsymbol{x})+\boldsymbol{m}_\ell\odot\boldsymbol{n}\\ &\quad+(\boldsymbol{1}-\boldsymbol{m}_\ell)\odot(\mathcal{A}\circ(\boldsymbol{m}_\ell\odot\boldsymbol{y})))-\boldsymbol{k}*\boldsymbol{x}).\end{aligned}\tag{18}$$

It can be seen that $\boldsymbol{k}*f_{\boldsymbol{\beta}}(\boldsymbol{m}_\ell\odot(\boldsymbol{k}*\boldsymbol{x})+\boldsymbol{m}_\ell\odot\boldsymbol{n}+(\boldsymbol{1}-\boldsymbol{m}_\ell)\odot(\mathcal{A}\circ(\boldsymbol{m}_\ell\odot\boldsymbol{y})))$ contributes to $\boldsymbol{r}(i)$ only if $\boldsymbol{m}_\ell(i)=0$. But in this case, $\boldsymbol{n}(i)$ is erased by $\boldsymbol{m}_\ell(i)$. This means that $\boldsymbol{n}(i)$ has no contribution to $\boldsymbol{r}(i)$. Together with that $\boldsymbol{n}(i)$ is independent of $\boldsymbol{n}(j)$ for any $i\neq j$, It is concluded that $\boldsymbol{r}(i)$ is independent to $\boldsymbol{n}(i)$ for all i. Therefore, we have

$$\mathbb{E}_{\boldsymbol{n}}\big[\boldsymbol{n}^\top\boldsymbol{r}\big]=(\mathbb{E}_{\boldsymbol{n}}\big[\boldsymbol{n}\big])^\top(\mathbb{E}_{\boldsymbol{n}}\big[\boldsymbol{r}\big])=0.\tag{19}$$

Combining (16), (17) and (19) gives that

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{n}}[\sum_{\ell}\|(1-\boldsymbol{m}_\ell)\odot(\boldsymbol{k}*f_{\boldsymbol{\beta}}(\widehat{\boldsymbol{y}}_\ell)-\boldsymbol{y})\|_2^2]\\ =&\sum_{\ell}\|\boldsymbol{k}*f_{\boldsymbol{\beta}}(\widehat{\boldsymbol{y}})-\boldsymbol{k}*\boldsymbol{x}\|^2_{\boldsymbol{m}_\ell}+\sum_{\ell}\|\boldsymbol{\sigma}\|^2_{\boldsymbol{m}_\ell}.\end{aligned}\tag{20}$$

The proof is done.

B MMSE Approximation

The derivation is based on the theory of variational inference; please see (Blei et al, 2017) for a comprehensive introduction. First, we take $p(\boldsymbol{\beta})$ as a uniform distribution on a sufficiently large bounded set $\mathbb{S}=[-C/2,C/2]^d$, where C is a sufficient large positive number and d is the dimensionality of $\boldsymbol{\beta}$. Additionally, the entries of the noise $\boldsymbol{n}$ are assumed to be *i.i.d.* variables following the Gaussian distribution of mean zero and variance σ^2. The KL divergence between $p(\boldsymbol{\beta}|\mathcal{D})$ and $q(\boldsymbol{\beta}|\boldsymbol{\theta})$ is given by

$$\begin{aligned}&\mathrm{KL}(q(\boldsymbol{\beta}|\boldsymbol{\theta})\|p(\boldsymbol{\beta}|\mathcal{D}))\\ =&\mathbb{E}_{q(\boldsymbol{\beta}|\boldsymbol{\theta})}\log q(\boldsymbol{\beta}|\boldsymbol{\theta})-\mathbb{E}_{q(\boldsymbol{\beta}|\boldsymbol{\theta})}\log p(\boldsymbol{\beta}|\mathcal{D})\\ =&\mathbb{E}_{q(\boldsymbol{\beta}|\boldsymbol{\theta})}\log q(\boldsymbol{\beta}|\boldsymbol{\theta})-\mathbb{E}_{q(\boldsymbol{\beta}|\boldsymbol{\theta})}\big(\log p(\boldsymbol{\beta})\\ &+\log p(\mathcal{D}|\boldsymbol{\beta})-\log p(\mathcal{D})\big)\\ =&\mathrm{KL}(q(\boldsymbol{\beta}|\boldsymbol{\theta})\|p(\boldsymbol{\beta}))-\mathbb{E}_{\boldsymbol{\beta}\sim q(\boldsymbol{\beta}|\boldsymbol{\theta})}\log p(\mathcal{D}|\boldsymbol{\beta})\\ &+\mathbb{E}_{\boldsymbol{\beta}\sim q(\boldsymbol{\beta}|\boldsymbol{\theta})}\log p(\mathcal{D}).\end{aligned}\tag{21}$$

Since $\log p(\mathcal{D})$ is irrelevant to the model parameter $\boldsymbol{\beta}$, we have $\mathbb{E}_{\boldsymbol{\beta}\sim q(\boldsymbol{\beta}|\boldsymbol{\theta})} \log p(\mathcal{D}) = \log p(\mathcal{D})$ which is a constant. Therefore, we have

$$\begin{aligned}&\mathrm{KL}(q(\boldsymbol{\beta}|\boldsymbol{\theta})\|p(\boldsymbol{\beta}|\mathcal{D})) = \\ &\mathrm{KL}(q(\boldsymbol{\beta}|\boldsymbol{\theta})\|p(\boldsymbol{\beta})) - \mathbb{E}_{\boldsymbol{\beta}\sim q(\boldsymbol{\beta}|\boldsymbol{\theta})} \log p(\mathcal{D}|\boldsymbol{\beta}) + \text{const.},\end{aligned} \tag{22}$$

where the KL divergence between $q(\boldsymbol{\beta}|\boldsymbol{\theta})$ and $p(\boldsymbol{\beta})$ can be written as

$$\begin{aligned}&\mathrm{KL}(q(\boldsymbol{\beta}|\boldsymbol{\theta})\|p(\boldsymbol{\beta})) = \\ &\mathbb{E}_{q(\boldsymbol{\beta}|\boldsymbol{\theta})} \log q(\boldsymbol{\beta}|\boldsymbol{\theta}) - \mathbb{E}_{q(\boldsymbol{\beta}|\boldsymbol{\theta})} \log p(\boldsymbol{\beta}).\end{aligned} \tag{23}$$

The first term on the right hand of the above equation is the negative entropy of $q(\boldsymbol{\beta}|\boldsymbol{\theta})$. Recall that $q(\boldsymbol{\beta}|\boldsymbol{\theta})$ is the probability of Bernoulli:

$$\boldsymbol{\beta} = \boldsymbol{\theta} \odot \boldsymbol{b}, \text{ where } \boldsymbol{b}(i) \sim \text{Bernoulli}(p_i). \tag{24}$$

Its entropy is given by

$$\begin{aligned}&- \mathbb{E}_{q(\boldsymbol{\beta}|\boldsymbol{\theta})} \log q(\boldsymbol{\beta}|\boldsymbol{\theta}) = \\ &- \prod_i \big(p_i \log p_i + (1-p_i)\log(1-p_i)\big),\end{aligned} \tag{25}$$

which is a constant irrelevant to the parameter $\boldsymbol{\theta}$ and can be ignored during training. Since $p(\boldsymbol{\beta}) = 0$ outside $\mathbb{S}$, $\log p(\boldsymbol{\beta}) = -\infty$ outside $\mathbb{S}$. Then $\int q(\boldsymbol{\beta}|\boldsymbol{\theta}) \log p(\boldsymbol{\beta}) = -\infty$ if $\int_{\mathbb{R}^d \setminus \mathbb{S}} q(\boldsymbol{\beta}|\boldsymbol{\theta}) \neq 0$ (which means that $\boldsymbol{\theta} \notin \mathbb{S}$). When $\boldsymbol{\theta} \in \mathbb{S}$, we have $\int_{\mathbb{S}} q(\boldsymbol{\beta}|\boldsymbol{\theta}) = 1$ and thus

$$\begin{aligned}&\mathbb{E}_{q(\boldsymbol{\beta}|\boldsymbol{\theta})} \log p(\boldsymbol{\beta}) \\ =&\int q(\boldsymbol{\beta}|\boldsymbol{\theta}) \log p(\boldsymbol{\beta}) = \int_{\mathbb{S}} q(\boldsymbol{\beta}|\boldsymbol{\theta}) \log \frac{1}{C^d} \\ =&\log \frac{1}{C^d}.\end{aligned} \tag{26}$$

It yields that

$$\int q(\boldsymbol{\beta}|\boldsymbol{\theta}) \log p(\boldsymbol{\beta}) = \begin{cases} \log \frac{1}{C^d}, & \boldsymbol{\theta} \in \mathbb{S} \\ -\infty, & \boldsymbol{\theta} \notin \mathbb{S}. \end{cases} \tag{27}$$

Finally, we obtained that

$$\mathrm{KL}(q(\boldsymbol{\beta}|\boldsymbol{\theta})\|p(\boldsymbol{\beta})) = \delta_{\mathbb{S}}(\boldsymbol{\theta}) + \text{const}, \tag{28}$$

where

$$\delta_{\mathbb{S}}(\boldsymbol{\theta}) = \begin{cases} 0, & \text{if } \boldsymbol{\theta} \in \mathbb{S}, \\ +\infty, & \text{otherwise.} \end{cases} \tag{29}$$

Recall that the samples in $\mathcal{D} = \{\widehat{\boldsymbol{y}}_\ell, (\boldsymbol{1} - \boldsymbol{m}_\ell) \odot \boldsymbol{y}\}_\ell$ are related by

$$(\boldsymbol{1} - m_\ell) \odot \boldsymbol{y} = (\boldsymbol{1} - m_\ell) \odot (\boldsymbol{k} * f_{\boldsymbol{\beta}}(\widehat{\boldsymbol{y}}_\ell) + \boldsymbol{n}). \tag{30}$$

Roughly, we assume these samples are independent from each other. Then we can obtain

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{\beta}\sim q(\boldsymbol{\beta}|\boldsymbol{\theta})} \log p(\mathcal{D}|\boldsymbol{\beta}) \\ =&\mathbb{E}_{\boldsymbol{\beta}\sim q(\boldsymbol{\beta}|\boldsymbol{\theta})} \sum_{\widehat{\boldsymbol{y}}_\ell \sim \boldsymbol{\Omega}} \log p((\boldsymbol{1} - \boldsymbol{m}_\ell) \odot \boldsymbol{y} | \boldsymbol{\beta}, \widehat{\boldsymbol{y}}_\ell) \\ &+ \mathbb{E}_{\boldsymbol{\beta}\sim q(\boldsymbol{\beta}|\boldsymbol{\theta})} \sum_{\widehat{\boldsymbol{y}}_\ell \sim \boldsymbol{\Omega}} \log p(\widehat{\boldsymbol{y}}_\ell | \boldsymbol{\beta}) \\ =&-\frac{1}{2\sigma^2} \mathbb{E}_{\boldsymbol{b}} \sum_{\widehat{\boldsymbol{y}} \sim \boldsymbol{\Omega}} \mathcal{C}(\boldsymbol{y}, \boldsymbol{k} * f_{\boldsymbol{\theta}\odot\boldsymbol{b}}(\widehat{\boldsymbol{y}})) + \text{const.}\end{aligned} \tag{31}$$

Finally, we have

$$\begin{aligned}&\mathrm{KL}(q(\boldsymbol{\beta}|\boldsymbol{\theta})\|p(\boldsymbol{\beta}|\mathcal{D})) = \\ &\frac{1}{2\sigma^2} \mathbb{E}_{\boldsymbol{b}} \sum_{\widehat{\boldsymbol{y}} \sim \boldsymbol{\Omega}} \mathcal{C}(\boldsymbol{y}, \boldsymbol{k} * f_{\boldsymbol{\theta}\odot\boldsymbol{b}}(\widehat{\boldsymbol{y}})) + \delta_{\mathbb{S}}(\boldsymbol{\theta}) + \text{const.}\end{aligned} \tag{32}$$

Thus, minimizing the KL divergence between $p(\boldsymbol{\beta}|\mathcal{D})$ and $q(\boldsymbol{\beta}|\boldsymbol{\theta})$ is equivalent to

$$\min_{\boldsymbol{\theta}\in\mathbb{S}} \mathbb{E}_{\widehat{\boldsymbol{y}}\sim\boldsymbol{\Omega}} \mathbb{E}_{\boldsymbol{b}}\, \mathcal{C}(\boldsymbol{y}, \boldsymbol{k} * f_{\boldsymbol{\theta}\odot\boldsymbol{b}}(\widehat{\boldsymbol{y}})). \tag{33}$$

Since the feasible set $\mathbb{S}$ is sufficiently large, the constraint $\boldsymbol{\theta} \in \mathbb{S}$ can be omitted in practice, which results in the our training loss in (5).

References

Anger J, Facciolo G, Delbracio M (2018) Modeling realistic degradations in non-blind deconvolution. In: Proceedings of the IEEE International Conference on Image Processing, IEEE, pp 978–982

Anger J, Delbracio M, Facciolo G (2019a) Efficient blind deblurring under high noise levels. In: International Symposium on Image and Signal Processing and Analysis, IEEE, pp 123–128

Anger J, Facciolo G, Delbracio M (2019b) Blind image deblurring using the l0 gradient prior. Image Processing on Line 9:124–142

Arridge S, Maass P, Öktem O, Schönlieb CB (2019) Solving inverse problems using data-driven models. Acta Numerica 28:1–174

Azzari L, Foi A (2016) Variance stabilization for noisy+estimate combination in iterative poisson denoising. IEEE Signal Processing Letters 23(8):1086–1090

Batson J, Royer L (2019) Noise2self: Blind denoising by self-supervision. Proceedings of the International Conference on Machine Learning

Bigdeli SA, Zwicker M, Favaro P, Jin M (2017) Deep mean-shift priors for image restoration. In: Proceedings of the International Conference on Neural Information Processing Systems, pp 763–772

Blei DM, Kucukelbir A, McAuliffe JD (2017) Variational inference: A review for statisticians. Journal of the American statistical Association 112(518):859–877

Chen G, Zhu F, Ann Heng P (2015) An efficient statistical method for image noise level estimation. In: Proceedings of the IEEE International Conference on Computer Vision, pp 477–485

Cho SJ, Ji SW, Hong JP, Jung SW, Ko SJ (2021) Rethinking coarse-to-fine approach in single image deblurring. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 4641–4650

Danielyan A, Katkovnik V, Egiazarian K (2011) BM3D frames and variational image deblurring. IEEE Transactions on Image Processing 21(4):1715–1728

Dong J, Pan J, Sun D, Su Z, Yang MH (2018) Learning data terms for non-blind deblurring. In: Proceedings of the European Conference on Computer Vision, pp 748–763

Dong J, Roth S, Schiele B (2021) Deep Wiener deconvolution: Wiener meets deep learning for image deblurring. Proceedings of the International Conference on Neural Information Processing Systems 33

Dong W, Wang P, Yin W, Shi G, Wu F, Lu X (2019) Denoising prior driven deep neural network for image restoration. IEEE Transactions on Pattern Analysis and Machine Intelligence 41(10):2305–2318

Eboli T, Sun J, Ponce J (2020) End-to-end interpretable learning of non-blind image deblurring. In: Proceedings of the European Conference on Computer Vision

Ehret T, Davy A, Morel JM, Facciolo G, Arias P (2019) Model-blind video denoising via frame-to-frame training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 11,369–11,378

Gal Y, Ghahramani Z (2016) Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: Proceedings of the International Conference on Machine Learning, pp 1050–1059

Gilton D, Ongie G, Willett R (2020) Neumann networks for linear inverse problems in imaging. IEEE Transactions on Computational Imaging 6:328–343

Gong D, Zhang Z, Shi Q, van den Hengel A, Shen C, Zhang Y (2020) Learning deep gradient descent optimization for image deconvolution. IEEE Transactions on Neural Networks and Learning Systems pp 1–15

Heckel R (2019) Regularizing linear inverse problems with convolutional neural networks. arXiv preprint arXiv:190703100v1

Heckel R, Hand P (2019) Deep decoder: Concise image representations from untrained nonconvolutional networks. Proceedings of the International Conference on Learning Representations

Hendriksen A, Pelt DM, Batenburg KJ (2020) Noise2inverse: Self-supervised deep convolutional denoising for linear inverse problems in imaging. IEEE Transactions on Computational Imaging abs/2001.11801

Jin M, Roth S, Favaro P (2017) Noise-blind image deblurring. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp 3834–3842

Kaufman A, Fattal R (2020) Deblurring using analysis-synthesis networks pair. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 5811–5820

Köhler R, Hirsch M, Mohler B, Schölkopf B, Harmeling S (2012) Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In: Proceedings of the European Conference on Computer Vision, Springer, pp 27–40

Krishnan D, Fergus R (2009) Fast image deconvolution using hyper-laplacian priors. In: Proceedings of the International Conference on Neural Information Processing Systems, pp 1033–1041

Krull A, Buchholz TO, Jug F (2019) Noise2void-learning denoising from single noisy images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp

2129–2137

Kruse J, Rother C, Schmidt U (2017) Learning to push the limits of efficient FFT-based image deconvolution. In: Proceedings of the IEEE International Conference on Computer Vision, IEEE, pp 4586–4594

Lai WS, Huang JB, Hu Z, Ahuja N, Yang MH (2016) A comparative study for single image blind deblurring. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1701–1709

Laine S, Lehtinen J, Aila T (2019) High-quality self-supervised deep image denoising. Proceedings of the International Conference on Neural Information Processing Systems

Lehtinen J, Munkberg J, Hasselgren J, Laine S, Karras T, Aittala M, Aila T (2018) Noise2noise: Learning image restoration without clean data. Proceedings of the International Conference on Machine Learning

Levin A, Weiss Y, Durand F, Freeman WT (2011) Efficient marginal likelihood optimization in blind deconvolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp 2657–2664

Li J, Shen Z, Yin R, Zhang X (2015) A reweighted $l2$ method for image restoration with poisson and mixed poisson-gaussian noise. Inverse Problem and Imaging 9(3):875–894

Meinhardt T, Möller M, Hazirbas C, Cremers D (2017) Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. Proceedings of the IEEE International Conference on Computer Vision pp 1799–1808

Nan Y, Ji H (2020) Deep learning for handling kernel/model uncertainty in image deconvolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 2388–2397

Nan Y, Quan Y, Ji H (2020) Variational-EM-based deep learning for noise-blind image deblurring. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 3626–3635

Pan J, Hu Z, Su Z, Yang MH (2014) Deblurring text images via L0-regularized intensity and gradient prior. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 2901–2908

Pan J, Sun D, Pfister H, Yang MH (2016) Blind image deblurring using dark channel prior. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1628–1636

Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A (2017) Pytorch document for maxpool2d. URL https://pytorch.org/docs/stable/generated/torch.nn.MaxPool2d.html

Quan Y, Chen M, Pang T, Ji H (2020) Self2self with dropout: Learning self-supervised denoising from single image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 1890–1898

Ren D, Zhang K, Wang Q, Hu Q, Zuo W (2020) Neural blind deconvolution using deep priors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 3341–3350

Ren W, Zhang J, Ma L, Pan J, Cao X, Zuo W, Liu W, Yang MH (2018) Deep non-blind deconvolution via generalized low-rank approximation. In: Proceedings of the International Conference on Neural Information Processing Systems, pp 295–305

Romano Y, Elad M, Milanfar P (2017) The little engine that could: Regularization by denoising (red). SIAM Journal on Imaging Sciences 10(4):1804–1844

Schmidt U, Roth S (2014) Shrinkage fields for effective image restoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 2774–2781

Schmidt U, Schelten K, Roth S (2011) Bayesian deblurring with integrated noise estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 2625–2632

Schuler CJ, Christopher Burger H, Harmeling S, Scholkopf B (2013) A machine learning approach for non-blind image deconvolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1067–1074

Soltanayev S, Chun SY (2018) Training deep learning based denoisers without ground truth data. In: Proceedings of the International Conference on Neural Information Processing Systems, pp 3257–3267

Son H, Lee S (2017) Fast non-blind deconvolution via regularized residual networks with long/short skip-connections. In: IEEE International Conference on Computational Photography, IEEE, pp 1–10

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1):1929–1958

Sun L, Cho S, Wang J, Hays J (2013) Edge-based blur kernel estimation using patch priors. In: IEEE International Conference on Computational Photography, IEEE, pp 1–8

Ulyanov D, Vedaldi A, Lempitsky V (2018) Deep image prior. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 9446–9454

Vasu S, Maligireddy VR, Rajagopalan AN (2018) Non-blind deblurring: Handling kernel uncertainty with CNNs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, pp 3272–3281

Vedaldi A, Lempitsky V, Ulyanov D (2020) Deep image prior. International Journal of Computer Vision

Vonesch C, Unser M (2008) A fast thresholded landweber algorithm for wavelet-regularized multidimensional deconvolution. IEEE Transactions on Image Processing 17:539–549

Wang Z, Wang Z, Li Q, Bilen H (2019) Image deconvolution with deep image and kernel priors. Proceedings of the IEEE/CVF International Conference on Computer Vision Workshop

Xu L, Ren JSJ, Liu C, Jia J (2014) Deep convolutional neural network for image deconvolution. In: Proceedings of the International Conference on Neural Information Processing Systems, pp 1790–1798

Yang L, Ji H (2019) A variational EM framework with adaptive edge selection for blind motion deblurring. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 10,159–10,168

Zhang J, s Pan J, Lai WS, Lau RWH, Yang MH (2017a) Learning fully convolutional networks for iterative non-blind deconvolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp 6969–6977

Zhang K, Zuo W, Chen Y, Meng D, Zhang L (2017b) Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. IEEE Transactions on Image Processing 26(7):3142–3155

Zhang K, Zuo W, Gu S, Zhang L (2017c) Learning deep CNN denoiser prior for image restoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 3929–3938

Zoran D, Weiss Y (2011) From learning models of natural image patches to whole image restoration. In: Proceedings of the IEEE International Conference on Computer Vision, IEEE, pp 479–486

Zukerman J, Tirer T, Giryes R (2020) BP-DIP: A backprojection based deep image prior. In: Proceedings of the European Conference on Computer Vision Workshop