What Does Your Computational Imaging Algorithm Not Know?: A Plug-and-Play Model Quantifying Model Uncertainty

Canberk Ekmekci
University of Rochester
Rochester, NY, USA.
cekmekci@ur.rochester.edu

Mujdat Cetin
University of Rochester
Rochester, NY, USA.
mujdat.cetin@rochester.edu

Abstract

Plug-and-Play is an algorithmic framework developed to solve image recovery problems. Thanks to the empirical success of convolutional neural network (CNN) denoisers, numerous Plug-and-Play algorithms utilizing CNN denoisers have been proposed to solve various image recovery tasks. Unfortunately, those Plug-and-Play algorithms lack representing the uncertainty on the parameters of CNN denoisers because their training procedure yields only a point estimate for the parameters of the CNN denoiser. In this paper, we present a novel Plug-and-Play model that quantifies the uncertainty on the parameters of the CNN denoiser. The proposed model places a probability distribution on the parameters of the CNN denoiser and carries out approximate Bayesian inference to obtain the posterior distribution of the parameters to characterize their uncertainty. The uncertainty information provided by the proposed Plug-and-Play model allows characterizing how certain the model is for a given input. The proposed Plug-and-Play model is applicable to a broad set of computational imaging problems, with the requirement that the data fidelity term is differentiable, and has a simple implementation in deep learning frameworks. We evaluate the proposed Plug-and-Play model on a magnetic resonance imaging reconstruction problem and demonstrate its uncertainty characterization capability.

1. Introduction

Plug-and-Play (PnP) [35] is a widely used algorithmic framework to solve image recovery problems. The main idea is to solve a regularized optimization problem using a splitting method, such as alternating direction method of multipliers (ADMM) [4] or proximal gradient descent (PGD) [25], and replace the subproblem that involves the regularizer with an off-the-shelf denoiser. Recently, several studies [41, 43, 22, 26] have shown that convolutional neural network (CNN) denoisers achieve state-of-the-art performance for image denoising problems. As a result, the use of CNN denoisers within the PnP framework has attracted attention, and PnP algorithms leveraging CNN denoisers have been successfully used in multiple studies [42, 38, 34, 15, 1, 2].

The primary advantage of PnP methods that use CNN denoisers over end-to-end models [13] is that PnP methods are highly *modular* for image recovery problems. After training a CNN denoiser for a problem, we can use the trained CNN denoiser for different variants of the problem by only modifying the data dependent part of the iterative reconstruction algorithm. Because the trained CNN denoiser is often stored to be reused later, it is crucial to know what the trained denoiser does not know including, *e.g.*, features that the denoiser has not encountered at the training stage. One way to detect cases for which the input of the CNN denoiser contains features that are not well-represented by the data used at the training stage is to quantify the uncertainty on the parameters of the denoiser, which is often referred to as the *model uncertainty* or *epistemic uncertainty* [19]. Unfortunately, training a denoiser by minimizing an empirical loss function yields only a *point* estimate for the parameters of the denoiser; therefore, existing PnP methods cannot characterize the uncertainty on the parameters of the denoiser.

In this paper, we propose a novel PnP model that is capable of quantifying model uncertainty. The proposed PnP model uses PGD as the splitting strategy and defines a probability distribution on the parameters of the CNN denoiser. Then, the proposed PnP model obtains a tractable approximation of the posterior distribution of the parameters of the CNN denoiser. The resulting approximation of the posterior distribution would integrate the uncertainty on the parameters of the denoiser into the predictive distribution of the PnP algorithm by marginalization. Finally, we use the mean of the predictive distribution as the reconstructed image and compute the standard deviation of the each entry of the predictive distribution to obtain the model uncertainty for each pixel of the reconstructed image.

We demonstrate the capabilities of the proposed PnP model on a magnetic resonance imaging reconstruction problem. Our results show that the proposed PnP model is able to detect the features that are not well-represented by the training dataset and that the proposed PnP model can be utilized to guide the process of building the training dataset. To the best of the authors' knowledge, this is the first time the problem of quantifying model uncertainty is considered for PnP algorithms.

2. Related Work

Plug-and-Play framework: The idea of Plug-and-Play (PnP) priors was first presented by Venkatakrishnan *et al*. [35] for the ADMM algorithm. Later, the idea was also used for different splitting methods to obtain variants of the original PnP-ADMM, such as PnP-half-quadratic splitting (PnP-HQS) [42], PnP-proximal gradient descent (PnP-PGD) [23], PnP-primal-dual splitting (PnP-PDS) [23], and PnP-FISTA [17]. Another design choice for a PnP algorithm besides the splitting method is the type of denoiser used in the update equations. One line of work on PnP methods used non-learning based denoisers such as BM3D [9] and NLM [6] and achieved significant empirical success on various image recovery problems such as Poisson inverse problems [27], electron tomographic reconstruction and sparse image interpolation [31], nonlinear inverse scattering [17], single photon imaging [7], single image super-resolution [5], and Fourier ptychographic microscopy [33].

Recently, multiple studies [41, 43, 22, 26] reported that CNN denoisers could achieve state-of-the-art performance on image denoising problems. Hence, utilizing CNN denoisers in PnP algorithms elicited increasing attention from the research community. PnP algorithms that use CNN denoisers were successfully applied to several image recovery problems including image deblurring [42, 34, 15], image inpainting [34], single image super-resolution [42, 15], magnetic resonance imaging [1], radar imaging [2], and computed tomography [38].

Model uncertainty for neural networks: Characterizing the model uncertainty for a neural network requires placing a distribution on the parameters of the neural network and computing the posterior distribution of the parameters. Unfortunately, computing the posterior distribution of the parameters exactly is not tractable for deep neural networks due to large number of parameters and complex hierarchical structures. Accordingly, several methods were proposed to approximate the posterior distribution of the parameters such as Markov Chain Monte Carlo methods [24, 36, 8, 21, 44] and variational inference methods [3, 14, 12]. One particular variational inference method, Monte Carlo Dropout (MC Dropout) [12], stands out from the crowd for its simplicity and scalability for deep neural networks. By using MC Dropout, variational inference can be carried out by simply applying Dropout [32] after the parameters of the neural network that we aim to perform variational inference for. MC Dropout has been successfully applied to problems in computer vision such as depth completion [19] and semantic segmentation [19, 18].

Model uncertainty for image recovery: Parallel to the work done in the computer vision community, Schlemper *et al*. [30] developed U-Net [28] and DC-CNN [29] based models that use MC Dropout as the variational inference method to quantify model uncertainty for the magnetic resonance imaging problem. Later, Xue *et al*. [37] proposed a similar U-Net based model, which also utilizes MC Dropout, to characterize the model uncertainty for the phase imaging problem. Recently, Ekmekci and Cetin [10] proposed an unfolding based model that leverages MC Dropout to quantify model uncertainty for linear inverse problems.

The fundamental difference between the models in [30, 37, 10] and the proposed PnP model lies in the difference between end-to-end models and PnP methods. End-to-end models must be trained for different variants of the recovery problem. On the other hand, PnP methods require training a CNN denoiser, and we can use the CNN denoiser for different variants of the recovery problem.

3. Method

The proposed PnP model uses PGD as the splitting strategy and places a prior distribution on the parameters of a CNN denoiser. After approximating the posterior distribution of the parameters given data, the proposed PnP model integrates the posterior distribution of the parameters into the predictive distribution by marginalization. The resulting predictive distribution allows generating the reconstructed image as well as model uncertainty information.

In this section, we present the details of the proposed PnP model. First, we define the predictive distribution for the PnP-PGD algorithm and state our assumptions. Next, we approximate the predictive distribution by approximating the posterior distribution of the parameters of the CNN denoiser via MC Dropout and applying series of Monte Carlo integrations. Finally, we approximate the mean of the approximation to use as the reconstructed image and compute the standard deviation of the each entry of the approximation of the predictive distribution to represent the model uncertainty for each pixel of the reconstructed image. The overall procedure is illustrated in Figure 1.

3.1. PnP-PGD method

We consider the following unconstrained optimization problem

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left\{ f\left(\mathbf{x}\right) + \gamma g\left(\mathbf{x}\right) \right\}, \tag{1}$$

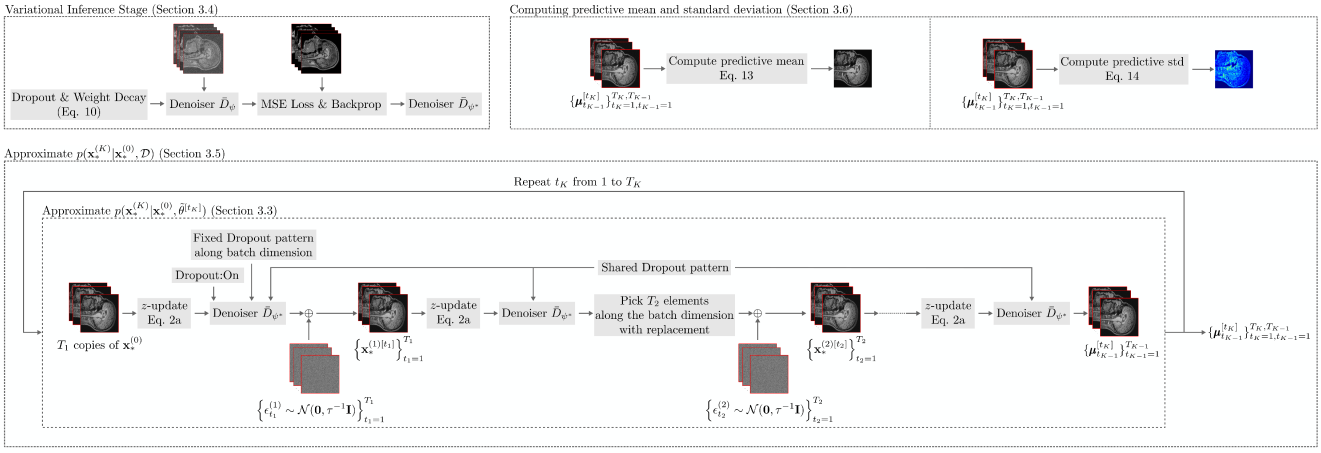


Figure 1. **Overview of the proposed PnP model.** We first perform variational inference to obtain the posterior distribution of the parameters of the denoiser used in the PnP-PGD algorithm. Then, we approximate the distribution of the reconstruction at the K^{th} iteration given reconstruction at the 0^{th} iteration and the training dataset containing noisy and clean images. Finally, we compute the mean and standard deviation of the predictive distribution to obtain the reconstructed image and the uncertainty on the parameters of the denoiser, respectively. Best viewed in color.

where the vector $\hat{\mathbf{x}} \in \mathbb{R}^S$ represents the reconstructed image in vectorized form, the vector $\mathbf{x} \in \mathbb{R}^S$ is the optimization variable, the function $f : \mathbb{R}^S \to \mathbb{R}$ is the data fidelity term, the scalar $\gamma \geq 0$ is the regularization constant, and the function $g : \mathbb{R}^S \to \mathbb{R}$ is the regularizer.

After solving the unconstrained optimization problem in (1) using PGD and replacing the proximal operator of the regularizer with a denoiser, we obtain the following iterative reconstruction algorithm.

$$\mathbf{z}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla f\left(\mathbf{x}^{(k)}\right) \tag{2a}$$

$$\mathbf{x}^{(k+1)} = D_\theta\left(\mathbf{z}^{(k+1)}\right), \tag{2b}$$

where the scalar $\alpha > 0$ is the step size, and the vectors $\mathbf{z}^{(k)}$ and $\mathbf{x}^{(k)}$ are the intermediary variable and the reconstructed image at the k^{th} iteration, respectively. D_θ is the CNN denoiser, where θ is the set of tunable parameters of the denoiser.

3.2. Predictive distribution for PnP-PGD

For the PnP-PGD algorithm in (2), we define the *predictive distribution* by the following integral.

$$p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \mathcal{D}) = \int p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \theta) p(\theta|\mathcal{D}) d\theta, \tag{3}$$

where the integer K is the number of iterations, and the vectors $\mathbf{x}_*^{(K)}$ and $\mathbf{x}_*^{(0)}$ are the reconstructed images at the K^{th} and 0^{th} iterations *at the inference stage*, respectively. The set $\mathcal{D}$ is the training dataset containing noisy and clean images, and the set θ contains the parameters of the CNN denoiser used in the PnP-PGD algorithm. In this definition, we have used the fact that *at the inference stage*, the reconstruction at the K^{th} iteration, $\mathbf{x}_*^{(K)}$, is independent of the dataset, $\mathcal{D}$, given the starting point, $\mathbf{x}_*^{(0)}$, and the set of parameters of the denoiser θ, and that the parameters of the denoiser, θ, is independent of the starting point, $\mathbf{x}_*^{(0)}$, given the dataset, $\mathcal{D}$.

In this definition, the posterior distribution of the parameters of the denoiser, $p(\theta|\mathcal{D})$, incorporates the uncertainty on the parameters of the denoiser into the predictive distribution. Thus, we can use, *e.g.*, the standard deviation of the predictive distribution to quantify the model uncertainty, and we can compute the mean of the predictive distribution to obtain the reconstructed image. Unfortunately, the integral that defines the predictive distribution cannot be obtained in closed-form because the distribution $p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \theta)$ and the posterior distribution of the parameters, $p(\theta|\mathcal{D})$, do not have closed-form representations if the denoiser is a deep neural network. Hence, we have to approximate the distributions $p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \theta)$ and $p(\theta|\mathcal{D})$ first, and then we can approximate the integral in (3) using Monte Carlo integration.

3.3. Approximating the distribution $p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \theta)$

To approximate the integral in (3), we start by approximating the distribution $p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \theta)$. Our *main* assumption is that the output of the denoiser, given the parameters of the denoiser and an input to the denoiser, has the following multivariate Gaussian form.

$$p(\mathbf{x}^{(k)}|\mathbf{z}^{(k)}, \theta) = \mathcal{N}(\mathbf{x}^{(k)}|D_\theta(\mathbf{z}^{(k)}), \tau^{-1}\mathbf{I}), \tag{4}$$

where we have used the vectors $\mathbf{x}^{(k)}$ and $\mathbf{z}^{(k)}$ to denote the output and the input of the denoiser D_θ, respectively, in accordance with the update equation in (2b). The set θ contains the parameters of the denoiser D_θ, and the scalar $\tau \geq 0$ is a fixed model parameter. We can justify this assumption by examining the similarity between the empirical loss function that is often used to train denoisers for PnP applications and the objective function of the maximum likelihood estimation problem for the parameters of the denoiser. If we assume that the likelihood function has the form (4), and the dataset, $\mathcal{D}$, contains i.i.d. examples, finding the maximum likelihood estimate of the parameters boils down to minimizing the mean squared error between the output of the denoiser and clean images, which is one of the most commonly used empirical loss functions in training denoisers.

Because the update step in (2a) is a deterministic operation for a given vector $\mathbf{x}^{(k)}$, by the assumption in (4), we can determine the form of the distribution $p(\mathbf{x}^{(k)}|\mathbf{x}^{(k-1)}, \theta)$ as follows:

$$p(\mathbf{x}^{(k)}|\mathbf{x}^{(k-1)}, \theta) = \mathcal{N}(\mathbf{x}^{(k)}|D_\theta(\mathbf{zu}(\mathbf{x}^{(k-1)})), \tau^{-1}\mathbf{I}) \quad (5)$$

where $\mathbf{zu}$ is the function performing the update step (2a).

Using the fact that the reconstruction at the k^{th} iteration depends only on the reconstruction at the $(k-1)^{th}$ iteration and the parameters of the denoiser θ, we can approximate the distribution $p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \theta)$ recursively as follows

$$p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \theta) = \int p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(K-1)}, \theta) \, p(\mathbf{x}_*^{(K-1)}|\mathbf{x}_*^{(0)}, \theta) d\mathbf{x}_*^{(K-1)}, \quad (6)$$

where the form of the distribution $p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(K-1)}, \theta)$ is given in (5). We calculate the result of the integral by using Monte Carlo integration and compute the distribution $p(\mathbf{x}_*^{(K-1)}|\mathbf{x}_*^{(0)}, \theta)$ by carrying out the recursion. The resulting iterative procedure is given in Algorithm 1 in detail.

Algorithm 1: Approximation of the distribution $p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \theta)$

Input: D_θ, τ^{-1}, K, $\{T_k\}_{k=1}^{K-1}$, $\mathbf{x}_*^{(0)}$, and $\mathbf{zu}$.
Output: Approximation of $p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \theta)$

$\mathbf{DZ} \leftarrow D_\theta \circ \mathbf{zu}$
for $k \leftarrow 1$ ***to*** $K-1$ **do**
  if k = *1* **then**
    $\{\mathbf{x}_*^{(1)[t_1]}\}_{t_1=1}^{T_1} \leftarrow T_1$ samples from $\mathcal{N}(\mathbf{x}_*^{(1)}|\,\mathbf{DZ}(\mathbf{x}_*^{(0)}), \tau^{-1}\mathbf{I})$
  else
    $\{\mathbf{x}_*^{(k)[t_k]}\}_{t_k=1}^{T_k} \leftarrow T_k$ samples from $\frac{1}{T_{k-1}} \sum_{t_{k-1}=1}^{T_{k-1}} \mathcal{N}(\mathbf{x}_*^{(k)}|\,\mathbf{DZ}(\mathbf{x}_*^{(k-1)[t_{k-1}]}), \tau^{-1}\mathbf{I})$
  end
end
$\{\boldsymbol{\mu}_{t_{K-1}}\}_{t_{K-1}=1}^{T_{K-1}} \leftarrow \{\mathbf{DZ}(\mathbf{x}_*^{(K-1)[t_{K-1}]})\}_{t_{K-1}=1}^{T_{K-1}}$
$approx \leftarrow \frac{1}{T_{K-1}} \sum_{t_{K-1}=1}^{T_{K-1}} \mathcal{N}(\mathbf{x}_*^{(K)}|\boldsymbol{\mu}_{t_{K-1}}, \tau^{-1}\mathbf{I})$
return $approx$

3.4. Approximating the posterior distribution of the parameters

In addition to an approximation of the distribution $p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \theta)$, we have to compute the posterior distribution of the parameters of the denoiser, $p(\theta|\mathcal{D})$, to approximate the integral in (3) using Monte Carlo integration. However, exact computation of the posterior distribution is not an easy task because the denoiser D_θ is a deep neural network, which has a large number of parameters and a complex structure.

In this paper, we utilize a commonly used variational inference method called MC Dropout to approximate the posterior distribution of the parameters of the denoiser. The main advantages of using MC Dropout are (i) the inference procedure is fast, (ii) it is scalable for deep neural networks since it does not introduce any parameters besides the parameters of the neural network, and (iii) it leads to variational inference and inference procedures that can be easily implemented in deep learning frameworks.

The main goal is to approximate the posterior distribution of the parameters of the denoiser, $p(\theta|\mathcal{D})$, with a parametrized distribution $q_\psi(\theta)$, where ψ is the set of adjustable parameters. The set of optimal parameters of the parametrized distribution, ψ^*, is found by minimizing the Kullback-Leibler (KL) divergence between the two distributions or, equivalently, maximizing the log-evidence lower bound.

$$\begin{aligned}\psi^* &= \underset{\psi}{\operatorname{argmin}} \{\mathrm{KL}(q_\psi(\theta)||p(\theta|\mathcal{D}))\} \\ &= \underset{\psi}{\operatorname{argmax}} \{\mathbb{E}_{\theta\sim q_\psi(\theta)}[\log p(S|Y,\theta)] - \mathrm{KL}(q_\psi(\theta)||p(\theta))\},\end{aligned} \quad (7)$$

where the distribution $p(\theta)$ is the prior distribution of the parameters of the denoiser, and the sets $S \triangleq \{\mathbf{s}_i\}_{i=1}^N$ and $Y \triangleq \{\mathbf{y}_i\}_{i=1}^N$ contain clean and noisy images in the dataset, $\mathcal{D}$, respectively. Assuming that the dataset, $\mathcal{D}$, contains i.i.d. examples and defining $k(\psi) \triangleq \mathrm{KL}(q_\psi(\theta)||p(\theta))$, we can approximate the log-evidence lower bound using Monte Carlo integration and obtain the following optimization problem.

$$\psi^* \approx \underset{\psi}{\operatorname{argmax}} \left\{ \sum_{i=1}^{N} \log p(\mathbf{s}_i|\mathbf{y}_i, \tilde{\theta}^{[i]}) - k(\psi) \right\}, \quad (8)$$

where the set $\{\tilde{\theta}^{[i]}\}$ contains i.i.d. samples from the distri-

bution $q_\psi(\theta)$.

To be able to compute the objective function in (8), we have to define only the form of the parametric distribution $q_\psi(\theta)$ since we have already defined the assumed form of the distribution $p(\mathbf{s}_i|\mathbf{y}_i, \tilde{\theta}^{[i]})$ in (4). MC Dropout defines the parametric distribution to be a Bernoulli variational distribution [12, 11]. Assuming that the denoiser has L convolutional layers, *i.e.*, $\theta = \{\mathbf{W}_i\}_{i=1}^L$, where the rows of the matrix $\mathbf{W}_i$ contain the coefficients of the filters in the i^{th} convolutional layer, the reparametrized version of the Bernoulli variational distribution is defined as

$$[\mathbf{B}_i]_{j,k} \sim \text{Bernoulli}(p_i) \tag{9a}$$

$$\mathbf{W}_i = \mathbf{B}_i \odot \mathbf{M}_i, \tag{9b}$$

where the set $\{p_i\}_{i=1}^L$ contains the success probabilities, the operator $\odot$ denotes the Hadamard product, and the set $\{\mathbf{M}_i\}_{i=1}^L$ contains the adjustable parameters of the Bernoulli variational distribution, *i.e.*, $\psi = \{\mathbf{M}_i\}_{i=1}^L$.

For the Bernoulli variational distribution in (9), the objective function of the optimization problem in (8) boils down to the following problem (see [12] for the details).

$$\psi^* = \underset{\psi}{\text{argmin}} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{s}_i - D_{\tilde{\theta}^{[i]}}(\mathbf{y}_i)\|_2^2 + \sum_{i=1}^{L} \delta_i \|\mathbf{M}_i\|_F^2, \tag{10}$$

where the scalar is defined as $\delta_i \triangleq (\tau^{-1} p_i)/N$ and the set $\{\tilde{\theta}^{[i]}\}_{i=1}^N$ contains N i.i.d. samples from the Bernoulli variational distribution $q_\psi(\theta)$.

Interestingly, generating a sample $\tilde{\theta}^{[i]}$ from the Bernoulli variational distribution defined in (9) requires generating realizations of Bernoulli random variables, see (9a), and multiplying them with the adjustable parameters of the Bernoulli variational distribution, see (9b). This process resembles the Dropout operation. If we consider *a copy* of the denoiser D_θ, with the exception that we add Dropout after convolutional layers, where the Dropout rate for the i^{th} convolutional layer is set to be $1 - p_i$, the parameters of the Dropout-added copy can be perceived as the adjustable parameters of the Bernoulli variational distribution. Consequently, generating a sample $\tilde{\theta}^{[i]}$ from the Bernoulli variational distribution defined in (9) and evaluating the denoiser with the sampled parameters at the point $\mathbf{y}_i$, *i.e.*, computing the term $D_{\tilde{\theta}^{[i]}}(\mathbf{y}_i)$, would be equal to evaluating the Dropout-added copy at the point $\mathbf{y}_i$ while the Dropout is enabled, *i.e.*, computing the term $\bar{D}_\psi(\mathbf{y}_i)$, where $\bar{D}_\psi$ is the Dropout-added copy. Therefore, solving the optimization problem in (10) boils down to training the Dropout-added copy, $\bar{D}_\psi$, using mean squared error loss function with the weight decay parameter of δ_i for the i^{th} convolutional layer.

3.5. Approximating the predictive distribution

So far, we have obtained approximations of the distributions $p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \theta)$ and $p(\theta|\mathcal{D})$. Now, we can approximate the predictive distribution in (3) using Monte Carlo integration as follows.

$$p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \mathcal{D}) \approx \int p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \theta) q_{\psi^*}(\theta) d\theta \tag{11a}$$

$$\approx \frac{1}{T_K} \sum_{t_K=1}^{T_K} p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \tilde{\theta}^{[t_K]}), \tag{11b}$$

$$\approx \frac{1}{T_K} \sum_{t_K=1}^{T_K} \frac{1}{T_{K-1}} \sum_{t_{K-1}=1}^{T_{K-1}} \mathcal{N}(\mathbf{x}_*^{(K)}|\boldsymbol{\mu}_{t_{K-1}}^{[t_K]}, \tau^{-1}\mathbf{I}), \tag{11c}$$

where the first line (11a) results from replacing the true posterior distribution of the parameters of the denoiser $p(\theta|\mathcal{D})$ with its approximation $q_{\psi^*}(\theta)$, and the second line (11b) is due to approximating the integral in (11a) using Monte Carlo integration with T_K samples, where the set $\{\tilde{\theta}^{[t_K]}\}_{t_K=1}^{T_K}$ contains T_K i.i.d. samples from $q_{\psi^*}(\theta)$. The last line (11c) follows from replacing the distribution $p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \tilde{\theta}^{[t_K]})$ with its approximation obtained by Algorithm 1.

Note that approximation of the distribution $p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \tilde{\theta}^{[t_K]})$, see Algorithm 1, requires the use of the denoiser with the sampled parameters, $D_{\tilde{\theta}^{[t_K]}}$. Based on the discussion in Section 3.4, we can use the Dropout-added copy, $\bar{D}_{\psi^*}$, which has been trained by using mean squared error loss function with weight decay, see (10), in lieu of the denoiser with the sampled parameters, $D_{\tilde{\theta}^{[t_K]}}$, in Algorithm 1. Hence, at the inference stage, we simply use the Dropout-added copy, $\bar{D}_{\psi^*}$, to obtain the mean of each mixture component of the mixture of Gaussians approximation of the predictive distribution.

3.6. Reconstructed image and model uncertainty information

The form of the approximation of the predictive distribution in (11c) is a mixture of Gaussians distribution with $T_{K-1}T_K$ mixture components:

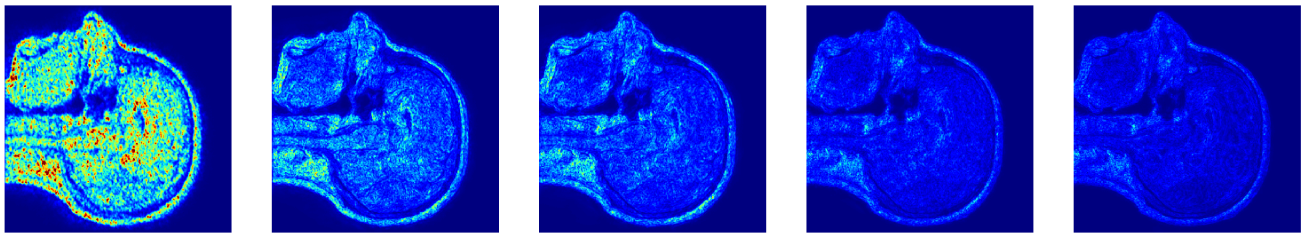
$$p(\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \mathcal{D}) \approx \sum_{t_K=1}^{T_K} \sum_{t_{K-1}=1}^{T_{K-1}} \beta \mathcal{N}(\mathbf{x}_*^{(K)}|\boldsymbol{\mu}_{t_{K-1}}^{[t_K]}, \tau^{-1}\mathbf{I}), \tag{12}$$

where $\beta \triangleq 1/(T_K T_{K-1})$. Therefore, we can compute the mean of the approximation of the predictive distribution as

$$\mathbb{E}[\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \mathcal{D}] \approx \beta \sum_{t_K=1}^{T_K} \sum_{t_{K-1}=1}^{T_{K-1}} \boldsymbol{\mu}_{t_{K-1}}^{[t_K]}, \tag{13}$$

which would be the reconstructed image by the proposed PnP model.

To represent the model uncertainty for each pixel in the reconstructed image, we can compute the standard deviation of the each entry of the predictive distribution. Because



(a) $N = 10$ (b) $N = 50$ (c) $N = 125$ (d) $N = 250$ (e) $N = 530$

Figure 2. **Uncertainty maps obtained by the proposed PnP model using five training datasets with increasing sizes.** At the variational inference stage, we used five training datasets having the sizes $10, 50, 125, 250$, and 530, respectively. At the inference stage, we obtained the uncertainty maps for a test image. The colormap of each plot is normalized between 0 and 0.5. Best viewed in color.

the approximation of the predictive distribution is a mixture of Gaussians distribution, its covariance matrix has the following form.

$$\mathbf{C} \approx \tau^{-1}\mathbf{I} + \beta \sum_{t_K=1}^{T_K} \sum_{t_{K-1}=1}^{T_{K-1}} (\boldsymbol{\mu}_{t_{K-1}}^{[t_K]})(\boldsymbol{\mu}_{t_{K-1}}^{[t_K]})^\top - \left(\mathbb{E}[\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \mathcal{D}]\right)\left(\mathbb{E}[\mathbf{x}_*^{(K)}|\mathbf{x}_*^{(0)}, \mathcal{D}]\right)^\top \tag{14}$$

Hence, the square root of the diagonal elements of the covariance matrix in (14) would be the model uncertainty information provided by the proposed PnP model.

4. Experiments

In this section, we evaluate the proposed PnP model on simulated MRI experiments and demonstrate the capabilities of the proposed PnP model. For the experiments, we use publicly available data from the IXI Dataset [16] and consider the following forward problem:

$$\mathbf{y} = \mathbf{F}_\Omega \mathbf{x} + \mathbf{n}, \tag{15}$$

where $\mathbf{y}$ is the measurement vector, the matrix $\mathbf{F}_\Omega$ denotes the subsampled Fourier transform operator, the set Ω determines the subsampling pattern, $\mathbf{x}$ is the underlying image, and the vector $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma_n^2\mathbf{I})$ represents the noise in the system. The image recovery problem for the forward problem in (15) takes the form (1) with the data fidelity term $f(\mathbf{x}) = \|\mathbf{F}_\Omega\mathbf{x} - \mathbf{y}\|_2^2$.

4.1. Experimental Setup

We obtained the training and test datasets by extracting 530 and 15 MR images from the IXI Dataset, respectively. Training and test images were normalized within a range between 0 and 1.

For the variational inference procedure described in Section 3.4, we implemented the Dropout-added copy by using the available implementation of DRUNet [40] in [39], replacing each transposed convolutional layer with a cascade of upscaling and a convolutional layer, and adding Dropout after every convolutional layer with the Dropout rate of 0.1, except the last convolutional layer. Unless otherwise stated, the precision parameter τ^{-1} was set to 10^{-6}, and the Dropout-added denoiser was trained by minimizing the objective function in (10) with the Adam algorithm [20]. During training, noisy images were obtained by adding Gaussian noise, whose standard deviation was randomly chosen from $[0, 50/255]$, to training images. The learning rate was set to 0.001, and size of the mini-batches was fixed to be 16.

At the inference stage, for a given test image, we obtained the test measurement vector $\mathbf{y}_*$ by using the the forward problem in (15). We set the starting point, $\mathbf{x}_*^{(0)}$, to be $\mathbf{F}_\Omega^H\mathbf{y}_*$, the step size, α, to be 0.001, and the number of iterations, K, to be 10. We fixed $\{T_k\}_{k=1}^K = 10$ and obtained the uncertainty map by following the procedure described in Section 3.5 and Section 3.6. For the experiments in Section 4.2 and Section 4.3, we used a radial mask choosing 50% of the Fourier coefficients for the undersampling pattern Ω and set the standard deviation of the noise, σ_n, so that the signal-to-noise ratio (SNR) is 45 dB.

4.2. Model uncertainty and size of the dataset

In this section, we illustrate the effect of the size of the training dataset on the model uncertainty. For this purpose, we generated five subsets of the training dataset having the sizes of $10, 50, 125, 250$, and 530. For each subset, we carried out the variational inference step and obtained the uncertainty map for a test image. The resulting uncertainty maps are presented in Figure 2.

For the case where we had 10 training images, we observed the highest overall model uncertainty. As we increased the size of the training dataset, the model uncertainty started decreasing gradually. We obtained the lowest overall model uncertainty for the largest training dataset. Our results indicate that we can use the proposed PnP model to visualize the change in the model uncertainty as a function of changes in the size of the training dataset, and we

can reduce the model uncertainty by adding more data to the training dataset. Hence, uncertainty characterization and visualization by the proposed PnP model can be used to guide the process of building the training dataset.

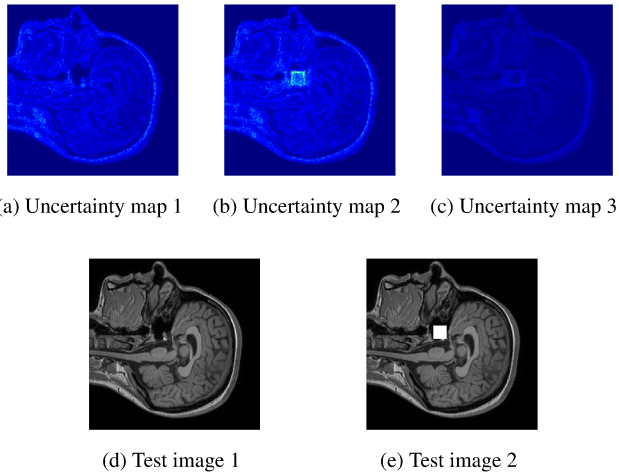
4.3. Model uncertainty and abnormal features

A feature that is not well-represented by the training dataset might occur at the inference stage of a PnP method that uses a CNN denoiser. We expect our proposed uncertainty characterization approach to detect such features. In this section, we use the proposed PnP model in a scenario where the test image contains a feature that is not present in the training dataset. We refer to these type of features as *abnormal* features in the rest of this section.

First, we carried out the variational inference stage using 530 training images. In the inference process, we obtained the model uncertainty information for the test image illustrated in Figure 3d. Figure 3a shows the resulting uncertainty map. Next, to simulate a scenario where the proposed PnP model encounters a test image containing features that are not present in the training dataset, we added a 20×20 white square on a test image and obtained the uncertainty map for the square-added test image. The square-added test image and the uncertainty map are depicted in Figure 3e and Figure 3b, respectively. As can be seen by comparing the uncertainty maps in Figure 3a and Figure 3b, the model uncertainty was increased around the abnormal feature since the training dataset did not contain such features. Finally, we inserted 20×20 white squares on the training images to see whether the proposed PnP model gets confident about this feature once it is represented well in the training dataset. We carried out the variational inference stage using square-added training images and obtained the model uncertainty information for the square-added test image. Figure 3c illustrates the uncertainty map obtained by the proposed PnP model that had been further trained with test images containing white squares. As can be observed in Figure 3c, the model uncertainty around the white square was significantly decreased because the final proposed PnP model encountered images containing squares in the training dataset. Our results showed that the proposed PnP model is capable of detecting abnormal features not represented well in the training data, and producing high uncertainty about such features in the reconstruction. We have also verified that if the type of feature considered in this experiment is present in the training data, the approach does not produce high uncertainty about it.

4.4. Reconstruction performance

In this section, we compare the reconstruction performance of the proposed PnP model with zero-filling (ZF), total variation (TV) based reconstruction method, and a PnP-PGD algorithm that uses DRUNet denoiser. The goal of this analysis is to verify that the image reconstruction performance is not significantly adversely affected by the addition of uncertainty characterization into the image formation process. We tested these methods on 15 test images, for which measurements were generated according to the procedure described in Section 4.1. Table 1 lists the undersampling rates for the radial masks and SNR values used in the experiments.

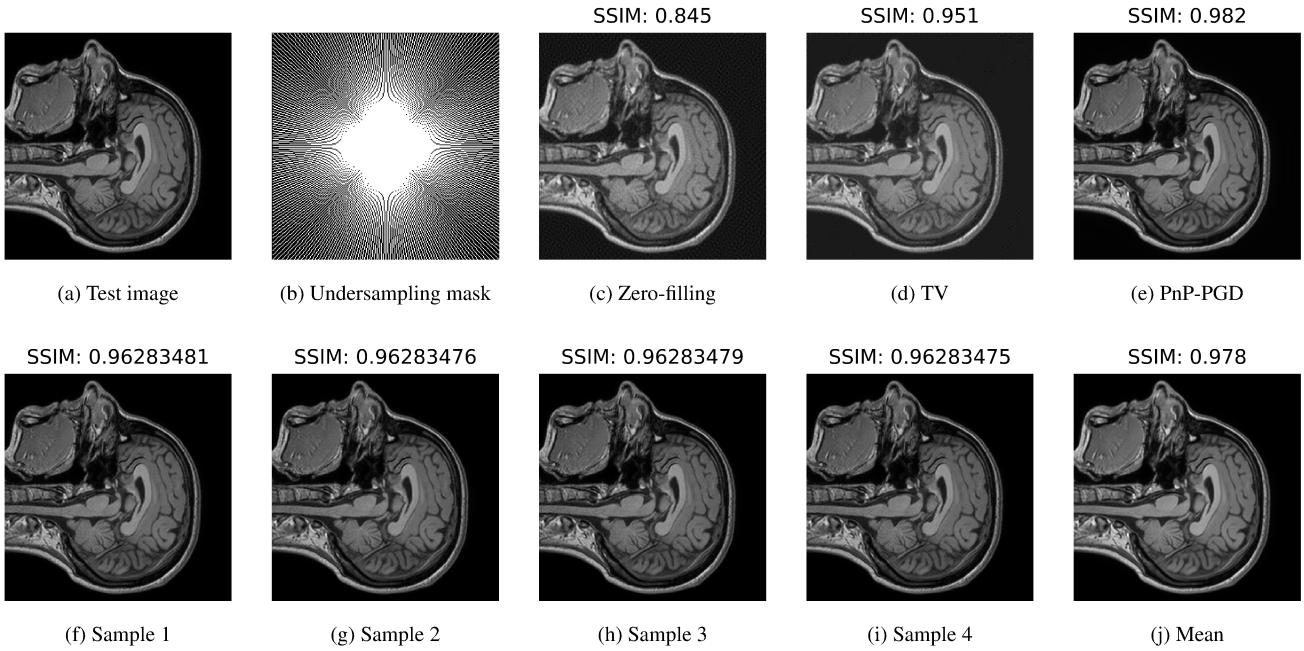


(a) Uncertainty map 1 (b) Uncertainty map 2 (c) Uncertainty map 3

(d) Test image 1 (e) Test image 2

Figure 3. **Output of the proposed PnP model for two test images, one containing an abnormal feature, and the other not.** At the variational inference stage, we used the training dataset containing MR images from the IXI Dataset. At the inference stage, we obtained the uncertainty maps (a) and (b) for the test images (d) and (e), respectively. Next, we added images containing white squares to the training dataset and repeated the variational inference stage. At the inference stage, we obtained the uncertainty map (c) for the test image (e). Best viewed in color.

For the TV method, the number of iterations was set to 100, and the regularization parameter was chosen from the set $\{0.0001, 0.001, 0.01, 0.1, 1, 10\}$ to achieve the highest SSIM. For PnP-PGD method, the denoiser was trained by minimizing the mean squared error with the Adam algorithm. The noisy images were obtained by adding Gaussian noise, whose standard deviation was randomly chosen from $[0, 50/255]$, to the clean images. The learning rate was set to 0.001, and we used a mini-batch size of 16. The step size of the PnP algorithm and noise level parameter of the denoiser were chosen from the sets $\{0.01, 0.05, 0.1, 0.5\}$ and $\{0.001, 0.005, 0.01, 0.05, 0.1\}$, respectively, to achieve the highest SSIM. The number of iterations of the PnP-PGD algorithm was fixed to 100. For the proposed PnP method, number of iterations, K, was set to 100, and we fixed $\{T_k\}_{k=1}^{K} = 10$. The step size, α, and noise level parameter of the denoiser were chosen from the sets $\{0.01, 0.05, 0.1, 0.5\}$ and $\{0.001, 0.005, 0.01, 0.05, 0.1\}$, respectively, to achieve the highest SSIM.



(a) Test image (b) Undersampling mask (c) Zero-filling (d) TV (e) PnP-PGD

(f) Sample 1 (g) Sample 2 (h) Sample 3 (i) Sample 4 (j) Mean

Figure 4. **Reconstructed test images using zero-filling (ZF), total variation regularized reconstruction (TV), PnP-PGD algorithm with DRUNet (PnP-PGD), and the proposed PnP model.** (a) is the ground truth test image, and (b) is the undersampling mask. (c), (d), and (e) are the outputs of ZF, TV and PnP-PGD methods. (f-i) are the samples from the approximation of the predictive distribution, and (j) is the mean of the predictive distribution, *i.e.*, the reconstructed image provided by the proposed PnP model.

UR	SNR	ZF	TV	PnP-PGD	Proposed
30%	10	0.473	0.670	**0.855**	<u>0.850</u>
	70	0.711	0.832	**0.923**	<u>0.912</u>
50%	10	0.469	0.545	**0.871**	<u>0.862</u>
	70	0.871	0.908	**0.977**	<u>0.971</u>
70%	10	0.438	0.546	**0.880**	<u>0.856</u>
	70	0.956	0.960	**0.990**	<u>0.986</u>

Table 1. **Comparison of average SSIM between different reconstruction methods under various undersampling rates (URs) and signal-to-noise ratios (SNRs).** SNR is expressed in decibels, and the best and the second best results are highlighted by bold font and underline, respectively.

Figure 4 shows the results of the methods for a test image, and Table 1 compares the reconstruction performance of the methods under different settings. Our experiments showed that both PnP-PGD method and the proposed PnP model outperformed the TV method and that the proposed PnP model could achieve comparable performance to the standard PnP-PGD method. However, we observed that the run time of the proposed PnP model is longer than the standard PnP-PGD method, mainly due to the Monte Carlo sampling operation in (11b). Luckily, the sampling operation in (11b) is parallelizable, *i.e.*, each summand in (11b) can be calculated in parallel, so we can shorten the run time of the proposed PnP method at the expense of using multiple processing units at the inference stage.

5. Conclusion

In this paper, we proposed a novel PnP model combining the PnP-PGD algorithm and the MC Dropout method to quantify model uncertainty. The proposed PnP model can be applied to a diverse set of image recovery problems and can be implemented in a deep learning framework straightforwardly. We tested the proposed PnP model on a magnetic resonance imaging problem using real data from the IXI Dataset. We demonstrated that the proposed PnP model could provide information about how certain the model is for a given test sample and training data, and that it could potentially be used to guide the training dataset construction process. It may also be of interest to characterize the aleatoric uncertainty [19] by learning the covariance matrix of the distribution in (4). Characterization of epistemic and aleatoric uncertainties together may bridge the performance gap between the standard PnP-PGD method and the proposed PnP model. We leave this extension for future work.

Acknowledgment

This work was partially supported by the National Science Foundation (NSF) under Grant CCF-1934962.

References

[1] Rizwan Ahmad, Charles A. Bouman, Gregery T. Buzzard, Stanley Chan, Sizhuo Liu, Edward T. Reehorst, and Philip Schniter. Plug-and-Play methods for magnetic resonance imaging: Using denoisers for image recovery. *IEEE Signal Processing Magazine*, 37(1):105–116, 2020.

[2] Muhammed Burak Alver, Ammar Saleem, and Mujdat Cetin. Plug-and-Play synthetic aperture radar image formation using deep priors. *IEEE Transactions on Computational Imaging*, 7:43–57, 2021.

[3] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, 2015.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[5] Alon Brifman, Yaniv Romano, and Michael Elad. Turning a denoiser into a super-resolver using Plug and Play priors. In *IEEE International Conference on Image Processing*, 2016.

[6] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[7] Stanley H. Chan, Xiran Wang, and Omar A. Elgendy. Plug-and-Play ADMM for image restoration: Fixed-point convergence and applications. *IEEE Transactions on Computational Imaging*, 3(1):84–98, 2017.

[8] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, 2014.

[9] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007.

[10] Canberk Ekmekci and Mujdat Cetin. Model-based Bayesian deep learning architecture for linear inverse problems in computational imaging. In *IS&T International Symposium on Electronic Imaging*, 2021.

[11] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *International Conference on Learning Representations*, 2016.

[12] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.

[13] Tobias Glasmachers. Limits of end-to-end learning. In *Asian Conference on Machine Learning*, 2017.

[14] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, 2011.

[15] Shuhang Gu, Radu Timofte, and Luc Van Gool. Integrating local and non-local denoiser priors for image restoration. In *International Conference on Pattern Recognition*, 2018.

[16] IXI Dataset. `https://brain-development.org/ixi-dataset/`.

[17] Ulugbek S. Kamilov, Hassan Mansour, and Brendt Wohlberg. A Plug-and-Play priors approach for solving nonlinear imaging inverse problems. *IEEE Signal Processing Letters*, 24(12):1872–1876, 2017.

[18] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian SegNet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *British Machine Vision Conference*, 2017.

[19] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, 2017.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[21] Yi-An Ma, Tianqi Chen, and Emily B. Fox. A complete recipe for stochastic gradient MCMC. In *International Conference on Neural Information Processing Systems*, 2015.

[22] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in Neural Information Processing Systems*, 2016.

[23] Tim Meinhardt, Michael Moeller, Caner Hazirbas, and Daniel Cremers. Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In *IEEE International Conference on Computer Vision*, 2017.

[24] Radford M. Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

[25] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[26] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. In *Advances in Neural Information Processing Systems*, 2018.

[27] Arie Rond, Raja Giryes, and Michael Elad. Poisson inverse problems by the Plug-and-Play scheme. *Journal of Visual Communication and Image Representation*, 41:96–108, 2016.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.

[29] Jo Schlemper, Jose Caballero, Joseph V. Hajnal, Anthony N. Price, and Daniel Rueckert. A deep cascade of convolutional neural networks for dynamic MR image reconstruction. *IEEE Transactions on Medical Imaging*, 37(2):491–503, 2018.

[30] Jo Schlemper, Daniel C. Castro, Wenjia Bai, Chen Qin, Ozan Oktay, Jinming Duan, Anthony N. Price, Jo Hajnal, and Daniel Rueckert. Bayesian deep learning for accelerated MR image reconstruction. In *International Workshop on Machine Learning for Medical Image Reconstruction*, 2018.

[31] Suhas Sreehari, S. V. Venkatakrishnan, Brendt Wohlberg, Gregery T. Buzzard, Lawrence F. Drummy, Jeffrey P. Simmons, and Charles A. Bouman. Plug-and-Play priors for bright field electron tomography and sparse interpolation. *IEEE Transactions on Computational Imaging*, 2(4):408–423, 2016.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[33] Yu Sun, Shiqi Xu, Yunzhe Li, Lei Tian, Brendt Wohlberg, and Ulugbek S. Kamilov. Regularized Fourier ptychography using an online Plug-and-Play algorithm. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

[34] Tom Tirer and Raja Giryes. Image restoration by iterative denoising and backward projections. *IEEE Transactions on Image Processing*, 28(3):1220–1234, 2019.

[35] Singanallur V. Venkatakrishnan, Charles A. Bouman, and Brendt Wohlberg. Plug-and-Play priors for model based reconstruction. In *IEEE Global Conference on Signal and Information Processing*, 2013.

[36] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *International Conference on Machine Learning*, 2011.

[37] Yujia Xue, Shiyi Cheng, Yunzhe Li, and Lei Tian. Reliable deep-learning-based phase imaging with uncertainty quantification. *Optica*, 6(5):618–629, 2019.

[38] Dong Hye Ye, Somesh Srivastava, Jean-Baptiste Thibault, Ken Sauer, and Charles Bouman. Deep residual learning for model-based iterative CT reconstruction using Plug-and-Play framework. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018.

[39] Kai Zhang. Deep Plug-and-Play image restoration. `https://github.com/cszn/DPIR`.

[40] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. Plug-and-Play image restoration with deep denoiser prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[41] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[42] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep CNN denoiser prior for image restoration. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[43] Kai Zhang, Wangmeng Zuo, and Lei Zhang. FFDNet: Toward a fast and flexible solution for CNN-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[44] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient MCMC for Bayesian deep learning. In *International Conference on Learning Representations*, 2020.