physical sciences forum

MDPI

Proceeding Paper

Bayesian Inference and Deep Learning for Inverse Problems †

Ali Mohammad-Djafari [1,2,*], Ning Chu [2], Li Wang [3] and Liang Yu [4,5]

[1] International Science Consulting and Training (ISCT), 91440 Bures sur Yvette, France
[2] Zhejiang Shangfeng Special Blower Company, Shaoxing 312352, China; chuning1983@sina.com
[3] School of Mathematics and Statistics, Central South University, Changsha 410017, China; li.wang.csu@csu.edu.cn
[4] School of Civil Aviation, Northwestern Polytechnical University, Xi'an 710072, China
[5] State Key Laboratory of Airliner Integration Technology and Flight Simulation, Shanghai 200126, China; liang.yu@sjtu.edu.cn
* Correspondence: djafari@ieee.org
† Presented at the 42nd International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering, Garching, Germany, 3–7 July 2023.

Abstract: Inverse problems arise anywhere we have an indirect measurement. In general, they are ill-posed to obtain satisfactory solutions, which needs prior knowledge. Classically, different regularization methods and Bayesian inference-based methods have been proposed. As these methods need a great number of forward and backward computations, they become costly in computation, particularly when the forward or generative models are complex, and the evaluation of the likelihood becomes very costly. Using deep neural network surrogate models and approximate computation can become very helpful. However, in accounting for the uncertainties, we need first to understand Bayesian deep learning, and then we can see how we can use it for inverse problems. In this work, we focus on NN, DL, and, more specifically, the Bayesian DL particularly adapted for inverse problems. We first give details of Bayesian DL approximate computations with exponential families; then, we see how we can use them for inverse problems. We consider two cases: First, we consider the case where the forward operator is known and used as a physics constraint, and the second examines more general data-driven DL methods.

Keywords: bayesian inference; neural network; deep learning (DL); inverse problems; physics-based DL; infrared imaging

Citation: Mohammad-Djafari, A.; Chu, N.; Wang, L.; Yu, L. Bayesian Inference and Deep Learning for Inverse Problems. *Phys. Sci. Forum* **2023**, *9*, 14. https://doi.org/10.3390/psf2023009014

Academic Editors: Udo von Toussaint and Roland Preuss

Published: 1 December 2023

1. Introduction

Inverse problems arise almost everywhere in science and engineering. In fact, anytime and in any application when we have indirect measurements related to what we really want to measure through some mathematical relation, this is called the forward model. Then, we have to infer the desired unknown from the observed data using this forward model or a surrogate one. In general, many inverse problems are ill-posed, and many methods for finding well-posed solutions for them are mainly based either on regularization theory or Bayesian inference. We mention those in particular, which are based on the optimization of a criterion with two parts: a data model output matching criterion (likelihood part in the Bayesian) and a regularization term (prior model in the Bayesian). Different criteria for these two terms and a great number of standard and advanced optimization algorithms have been proposed and used with great success. When these two terms are distances, they can have a Bayesian maximum a posteriori (MAP) interpretation, where these two terms correspond, respectively, to the likelihood and prior probability models. The Bayesian approach gives more flexibility in choosing these terms via the likelihood and the prior probability distributions. This flexibility goes much further with the hierarchical models and appropriate hidden variables [1]. Also, the possibility of estimating the hyperparameters gives much more flexibility for semisupervised methods.

However, full Bayesian computations can become very heavy computationally. In particular, this occurs when the forward model is complex, and the evaluation of the likelihood has a high computational cost [2]. In those cases, using surrogate simpler models can become very helpful to reduce the computational costs, but then we have to account for the uncertainty quantification (UQ) of the obtained results [3]. Neural networks (NNs), with their diversity such as convolutional neural networks (CNNs), deep learning (DL), etc., have become tools for fast and low computational surrogate forward models.

Over the last decades, machine learning (ML) methods and algorithms have gained great success in many tasks, such as classification, clustering, segmentation, object detection, and many other areas. There are many different structures of neural networks (NNs), such as feed-forward, Convolutional NNs (CNNs), Deep NNs, etc. [4]. Using these methods directly for inverse problems, as intermediate preprocessing or as tools for performing fast approximate computation in different steps of regularization or Bayesian inference, has also been successful, but not as much as could be possible. Recently, physics-informed neural networks have gained great success in many inverse problems, thereby proposing interaction between the Bayesian formulation of forward models and optimization algorithms and ML-specific algorithms for intermediate hidden variables. These methods have become very helpful to obtain approximate practical solutions to inverse problems in real-world applications [5–11].

In this paper, first, in Section 2, some mathematical notations for dealing with NNs are given. In Section 3, a detailed presentation of the Bayesian inference and the approximate computation needed for BDL are given. Then, in Section 4, we consider a focus on the NN and DL methods for inverse problems. First, we present the same cases where we know the forward and its adjoint model. Then, we consider the case where we may not have this knowledge and want to propose directly data-driven DL methods [12,13].

2. Neural Networks, Deep Learning, and Bayesian DL

The main objective of the NN for a regression problem can be described as follows:

$$\mathbf{x}_i \rightarrow \boxed{\text{NN: } f(\mathbf{x}_i)} \rightarrow y_i.$$

The objective is to infer the function $f : \mathbb{R}^M \rightarrow \mathbb{R}$ from the observations $\mathbf{y} = (y_1, \ldots, y_N)^{\mathrm{T}}$ at locations given by $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$. The usual NN learning approach is to define a parametric family of functions $\phi_{\boldsymbol{\theta}} : \mathbb{R}^M \times \boldsymbol{\theta} \rightarrow \mathbb{R}$ that is flexible enough so that $\exists \boldsymbol{\theta}^\star$ such that $\phi(\cdot) \approx \phi_{\boldsymbol{\theta}^\star}(\cdot)$:

$$\{\mathbf{x}_i, y_i\} \rightarrow \boxed{\text{NN Learning: } y_i = \phi(\mathbf{x}_i) \approx \phi_{\boldsymbol{\theta}^\star}(\mathbf{x}_i)} \rightarrow \boldsymbol{\theta}^\star.$$

Deep learning focuses on learning the optimal parameters $\boldsymbol{\theta}^\star$, which can then be used for predicting the output $\hat{y}_j$ for any input $\mathbf{x}_j$:

$$\mathbf{x}_j \rightarrow \boxed{\text{NN: } \phi_{\boldsymbol{\theta}^\star}(\mathbf{x}_j)} \rightarrow \hat{y}_j.$$

In this approach, there is no uncertainty quantification.

The Bayesian deep learning approach can be summarized as follows:

$$\{\mathbf{x}_i, y_i\} \rightarrow \boxed{p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}} \rightarrow p(\boldsymbol{\theta}|\mathcal{D}),$$

$$\mathbf{x}_j \rightarrow \boxed{p(y_j|\mathbf{x}_j, \mathcal{D}) = \int p(y_j|\mathbf{x}_j, \boldsymbol{\theta}, \mathcal{D}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta}} \rightarrow \hat{y}_j.$$

As we can see, uncertainties are accounted for in both steps of the parameter estimation and prediction. However, as we will see, the computational costs are important. We need to find solutions to perform fast computation.

3. Bayesian Inference and Approximate Computation

In a general Bayesian framework for NNs and DL, the objective is to infer the parameters $\boldsymbol{\theta}$ from the data, $\mathcal{D} : \{\mathbf{x}_i, y_i\}$, using the Bayes rule:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}, \tag{1}$$

where $p(\boldsymbol{\theta})$ is the prior, $\ell(\mathcal{D}|\boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta})$ is the likelihood, $p(\boldsymbol{\theta}|\mathcal{D})$ is the posterior, and $p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}$ is called the evidence. We can also write $p(\boldsymbol{\theta}|\mathcal{D}) \propto \ell(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$, where the classical maximum likelihood estimation (MLE) is defined as $\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}}\{\ell(\mathcal{D}|\boldsymbol{\theta})\}$.

A particular point of the posterior is of high interest, because we may be interested in maximum a posterior (MAP) estimation: $\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}}\{\ell(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})\}$. We may also be interested in mean squared error (MSE) estimation, which is shown that it corresponds to $\widehat{\boldsymbol{\theta}} = \mathrm{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\{\boldsymbol{\theta}\} = \int \boldsymbol{\theta}\, p(\boldsymbol{\theta}|\mathcal{D})\,\mathrm{d}\boldsymbol{\theta}$.

The exact expression of the posterior and the computations of these integrals for great dimensional problems may be very computationally costly. For this reason, we need to perform approximate computation. In the following subsections, we review a few solutions.

3.1. Laplace Approximation

Rewriting the general Bayes rules slightly differently gives us the following:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{p(\mathcal{D})}p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) = \frac{1}{Z}\exp[\mathcal{L}(\boldsymbol{\theta}|\mathcal{D})], \tag{2}$$

the Laplace approximations use a second-order expansion of $\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \ln p(\mathcal{D}|\boldsymbol{\theta}) + \ln p(\boldsymbol{\theta})$ around $\widehat{\boldsymbol{\theta}}_{MAP}$ to construct a Gaussian approximation of $p(\boldsymbol{\theta}|\mathcal{D})$:

$$\mathcal{L}(\boldsymbol{\theta}) \approx \mathcal{L}(\widehat{\boldsymbol{\theta}}_{MAP}) + \frac{1}{2}(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_{MAP})'\left(\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})|_{\widehat{\boldsymbol{\theta}}_{MAP}}\right)(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_{MAP}), \tag{3}$$

where the first-order term vanishes at the $\widehat{\boldsymbol{\theta}}_{MAP}$. This is equivalent to calculating the following approximation:

$$\boxed{\text{Approximate} \quad p(\boldsymbol{\theta}|\mathcal{D}) \quad \text{by} \quad q(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\theta}|\widehat{\boldsymbol{\theta}}_{MAP}, \boldsymbol{\Sigma} = [\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})|_{\widehat{\boldsymbol{\theta}}_{MAP}}]\right).}$$

With this approximation, the evidence $p(\mathcal{D})$ is approximated by the following:

$$Z = p(\mathcal{D}) = (2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}\exp\left[-\mathcal{L}(\widehat{\boldsymbol{\theta}}_{MAP})\right]. \tag{4}$$

For great dimensional problems such as BDL, the full computation of $\boldsymbol{\Sigma}$ is very costly. We have still to do more approximations. The following are a few solutions for scalable approximations for BDL:

- Work with the subnetwork or last layer (transfer learning);
- Perform covariance matrix decomposition (low rank, Kronecker-factored approximate curvature (KFAC), Diag);
- Conduct the computation during the hyperparameter tuning using crossvalidation;
- Use approximate predictive computation.

3.2. Approximate Computation: Variational Inference

The main idea then is to perform approximate Bayesian computation (ABC) by approximating the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ using a simpler expression $q(\boldsymbol{\theta})$. When approximation is performed by minimizing

$$\mathrm{KL}[q(\boldsymbol{\theta}) : p(\boldsymbol{\theta}|\mathcal{D})] = \int q(\boldsymbol{\theta}) \log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})} \, \mathrm{d}\boldsymbol{\theta}, \tag{5}$$

the method is called the variational Bayesian approximation (VBA). When $q(\boldsymbol{\theta})$ is chosen to be separable in some components of the parameters $q(\boldsymbol{\theta}) = \prod_j q_j(\theta_j)$, the approximation is called *the mean field VBA (MFVBA)*.

Let us come back to the general VBA, $\mathrm{KL}[q : p] = \int q \log \frac{q}{p} = \mathrm{E}_q\{\log q\} - \mathrm{E}_q\{\log p\}$, and note using $L = \mathrm{E}_q\{\log p\}$ the expected likelihood and using $S = -\mathrm{E}_q\{\log q\}$ the entropy of q. Then, we have the following:

$$q* = \arg\min_q\{\mathrm{KL}[q : p]\} = \arg\max_q\{E = S + L\}. \tag{6}$$

E is also called the evidence lower bound (ELBO):

$$ELBO = -\mathrm{E}_q\{\log q\} + \mathrm{E}_q\{\log p\}. \tag{7}$$

At this point, it is important to note one main property of the VBA: When $p(\boldsymbol{\theta}|\mathcal{D})$, the posterior probability law p and the approximate probability law q are in the exponential family; then, $\mathrm{E}_p\{\boldsymbol{\theta}\} = \mathrm{E}_q\{\boldsymbol{\theta}\}$.

3.3. VBA and Natural Exponential Family

If q is chosen to be in a *natural exponential family*, $q(\boldsymbol{\theta}|\boldsymbol{\eta}) = \exp[\boldsymbol{\eta}'\boldsymbol{\theta} - A(\boldsymbol{\eta})]$, then it is entirely characterized by its mean $\mathbf{m} = \mathrm{E}_q\{\boldsymbol{\theta}\}$, and if q is conjugate to p, then $q^*(\boldsymbol{\theta}|\boldsymbol{\eta}) = \exp\left[\boldsymbol{\eta}^{*\prime}\boldsymbol{\theta} - A(\boldsymbol{\eta}^*)\right]$, which is entirely characterized by its mean $\mathbf{m}^* = \mathrm{E}_{q^*}\{\boldsymbol{\theta}\}$. We can then define the objective E as a function of $\mathbf{m}$, and the first order condition of the optimality is $\frac{\partial E}{\partial \mathbf{m}}\big|_{\mathbf{m}=\mathbf{m}*} = 0$. From this property, we can obtain a fixed-point algorithm to compute $\mathbf{m}^* = \mathrm{E}_{q^*}\{\boldsymbol{\theta}\}$:

$$\frac{\partial E}{\partial \mathbf{m}}\big|_{\mathbf{m}=\mathbf{m}*} = 0 \rightarrow \frac{\partial E}{\partial \mathbf{m}}\big|_{\mathbf{m}=\mathbf{m}*} + \mathbf{m} = \mathbf{m} \rightarrow \mathbf{M}(\mathbf{m}) = \mathbf{m}.$$

Iterating on this *fixed-point algorithm* gives us the following:

$$\mathbf{M}(\mathbf{m}) = \mathbf{m}^{(k-1)} \quad \text{with} \quad \mathbf{M}(\mathbf{m}) := \frac{\partial E}{\partial \mathbf{m}} + \mathbf{m} \tag{8}$$

which converges to $\mathbf{m}^* = \mathrm{E}_{q^*}\{\boldsymbol{\theta}^*\}$ and is also $\mathbf{m}^* = \mathrm{E}_p\{\boldsymbol{\theta}^*\}$. This algorithm can be summarized as follows:

- Having chosen the prior and likelihood, find the expression of $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$;
- Choose a family q and find the expressions of $L = \mathrm{E}_q\{\ln p\}$ and $S = \mathrm{E}_q\{\ln q\}$, which thus yield $E = L + S$ as a function of $\mathbf{m} = \mathrm{E}_q\{\boldsymbol{\theta}\}$;
- Find the expression of the vector operator $\mathbf{M} = \nabla_{\mathbf{m}} E + \mathbf{m}$ and update it $\mathbf{M}(\mathbf{m}) = \mathbf{m}^{(k-1)}$ until convergence, which results in $\mathbf{m}^* = \mathrm{E}_{q^*}\{\boldsymbol{\theta}^*\} = \mathrm{E}_{p^*}\{\boldsymbol{\theta}^*\}$.

At this point, it is important to note that, in this approach, even if the mean is well approximated, the variances or the covariance are underestimated. Some authors who are interested in this approach have proposed solutions to better estimate the covariance. See [14] for one of the solutions.

4. Deep Learning and Bayesian DL

As introduced before, in classical DL, the training and prediction steps can be summarized as follows:

$$\{\mathbf{x}_i, y_i\} \rightarrow \boxed{\text{NN Learning: } \phi_{\boldsymbol{\theta}^\star}(\cdot)} \rightarrow \boldsymbol{\theta}^\star \qquad \mathbf{x}_j \rightarrow \boxed{\text{NN: } \phi_{\boldsymbol{\theta}^\star}(\mathbf{x}_j)} \rightarrow \hat{y}_j$$

In this approach, there is no uncertainty quantification. The Bayesian deep learning can be summarized as follows:

$$\{\mathbf{x}_i, y_i\} \rightarrow \boxed{p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}} \rightarrow p(\boldsymbol{\theta}|\mathcal{D}) \text{ or } q(\boldsymbol{\theta}|\mathcal{D}),$$

$$\mathbf{x}_j \rightarrow \boxed{p(y_j|\mathbf{x}_j, \mathcal{D}) = \int p(y_j|\mathbf{x}_j, \boldsymbol{\theta}, \mathcal{D})p(\boldsymbol{\theta}|\mathcal{D})\, \mathrm{d}\boldsymbol{\theta}} \rightarrow p(y_i|\mathbf{x}_j, \mathcal{D}) \text{ or } q(y_i|\mathbf{x}_j, \mathcal{D}).$$

As we can see, uncertainties are accounted for in both steps of the parameter estimation and prediction. However, computational costs are important. We need to find solutions to perform fast computation. As mentioned before, the different possible tools are Laplace and Gaussian approximation, variational inference, and more controlled approximation to design new deep learning algorithms, which can scale up for practical situations.

Exponential Family Approximation

Let us consider the case of general exponential families $q(\boldsymbol{\theta}|\boldsymbol{\lambda}) = \exp\left[\boldsymbol{\lambda}^T \mathbf{t}(\boldsymbol{\theta}) - F(\boldsymbol{\lambda})\right]$, where $\boldsymbol{\theta}$ represents the original parameters, $\boldsymbol{\lambda}$ represents the natural parameters, $\mathbf{t}(\boldsymbol{\theta})$ represents sufficient statistics, $F(\boldsymbol{\lambda})$ represents the log partition function, and we define the expectations parameters as $\boldsymbol{\mu} := \mathrm{E}_q\{\mathbf{t}(\boldsymbol{\theta})\}$. Let us also define the dual function $F^*(\boldsymbol{\eta})$ and the dual parameters $\boldsymbol{\eta}$ via the Legendre transform:

$$G(\boldsymbol{\eta}) = F^*(\boldsymbol{\eta}) = \sup_{\boldsymbol{\lambda}} \{< \boldsymbol{\lambda}, \boldsymbol{\eta} > - F(\boldsymbol{\lambda})\}.$$

Then, we can show the triangular relation between $\boldsymbol{\theta} \in \Theta$, $\boldsymbol{\lambda} \in \Lambda$, and $\boldsymbol{\mu} \in \mathcal{M}$ shown in Figure 1.

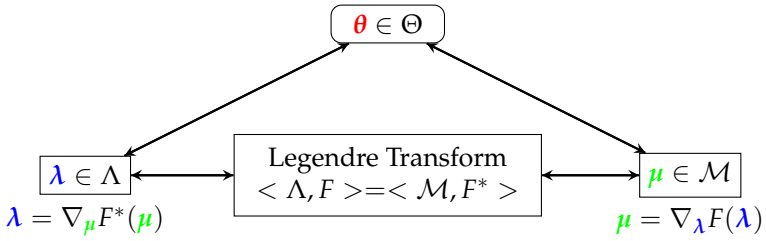

Figure 1. General exponential family, with original parameters $\boldsymbol{\theta}$, natural parameters $\boldsymbol{\lambda}$, expectations parameters $\boldsymbol{\mu} := \mathrm{E}_q\{\mathbf{t}(\boldsymbol{\theta})\}$, and their relations via the Legendre Transform.

With these notations, applying the VBA rule $\min_{q \in \mathcal{Q}} \mathrm{E}_q\{\mathcal{L}(\boldsymbol{\theta})\} - H(q)$ results in the following updating rule for the natural parameters:

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \rho \nabla_{\boldsymbol{\mu}} (\mathcal{L}(\boldsymbol{\theta}) - H(q)). \tag{9}$$

For example, by considering the Gaussian case

$$q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{m}, \mathbf{S}^{-1}) \propto \exp\left[-\frac{1}{2}(\boldsymbol{\theta} - \mathbf{m})^T \mathbf{S}(\boldsymbol{\theta} - \mathbf{m})\right] \propto \exp\left[(\mathbf{S}\mathbf{m})^T \boldsymbol{\theta} + \mathrm{Tr}(-\frac{\mathbf{S}}{2}\boldsymbol{\theta}\boldsymbol{\theta}^T)\right]. \tag{10}$$

we can identify the natural parameters as $\boldsymbol{\lambda} = [\mathbf{Sm}, -\mathbf{S}/2] \rightarrow \boldsymbol{\mu} := [\mathrm{E}_q\{\boldsymbol{\theta}\}, \mathrm{E}_q\{\boldsymbol{\theta}\boldsymbol{\theta}^T\}]$, and we easily obtain the following algorithm:

$$\begin{cases} \mathbf{Sm} \leftarrow (1-\rho)\mathbf{Sm} - \rho\nabla_{\mathrm{E}_q\{\boldsymbol{\theta}\}}\mathrm{E}_q\{\mathcal{L}(\boldsymbol{\theta})\}, \\ \mathbf{S} \leftarrow (1-\rho)\mathbf{S} - \frac{\rho}{2}\nabla_{\mathrm{E}_q\{\boldsymbol{\theta}\boldsymbol{\theta}^T\}}\mathrm{E}_q\{\mathcal{L}(\boldsymbol{\theta})\}. \end{cases} \tag{11}$$

where

$$\begin{cases} \nabla_{\mathrm{E}_q\{\boldsymbol{\theta}\}}\mathrm{E}_q\{\mathcal{L}(\boldsymbol{\theta})\} = \mathrm{E}_q\{\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})\} - 2\mathrm{E}_q\{\mathcal{L}(\boldsymbol{\theta})\}, \\ \nabla_{\mathrm{E}_q\{\boldsymbol{\theta}\boldsymbol{\theta}^T\}}\mathrm{E}_q\{\mathcal{L}(\boldsymbol{\theta})\} = \mathrm{E}_q\{\mathbf{H}(\boldsymbol{\theta})\}, \end{cases}$$

which results, explicitly, in

$$\begin{cases} \mathbf{m} \leftarrow \mathbf{m} - \rho\mathbf{S}^{-1}\nabla_{\mathbf{m}}\mathcal{L}(\mathbf{m}), \\ \mathbf{S} \leftarrow (1-\rho)\mathbf{S} + \rho\mathbf{H}_{\mathbf{m}}. \end{cases} \tag{12}$$

For a linear model $\mathbf{y} = \mathbf{X}\boldsymbol{\theta}$ and Gaussian priors, we have

$$\mathcal{L}(\boldsymbol{\theta}) = (\mathbf{y}-\mathbf{X}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}) + \gamma\boldsymbol{\theta}^T\boldsymbol{\theta} = -2\boldsymbol{\theta}^T(\mathbf{X}^T\mathbf{y}) + \mathrm{Tr}\Big[\boldsymbol{\theta}\boldsymbol{\theta}^T(\mathbf{X}^T\mathbf{X} + \gamma\mathbf{I})\Big], \tag{13}$$

and $\mathrm{E}_q\{\mathcal{L}(\boldsymbol{\theta})\} = \boldsymbol{\lambda}^T\boldsymbol{\mu}, \nabla_{\boldsymbol{\mu}}\mathrm{E}_q\{\mathcal{L}(\boldsymbol{\theta})\} = \boldsymbol{\lambda}$.

It is interesting to note that many classical algorithms for updating the parameters, such as forward–backward, sparse variational inference, and variational message passing, become special cases. A main remark here is that this linear generating function case will link us to the linear inverse problems $\mathbf{g} = \mathbf{H}\mathbf{f} + \boldsymbol{\epsilon}$ if we replace $\mathbf{y}$ with $\mathbf{g}$, $\mathbf{X}$ with $\mathbf{H}$, and $\boldsymbol{\theta}$ with $\mathbf{f}$.

5. NN, DL, and Bayesian Inference for Linear Inverse Problems

To show the possibilities of the interaction between inverse problems methods, neural networks and deep learning, the best way is to give a few examples.

5.1. First Example: A Known Linear Forward Model

The first and easiest example is the case of linear inverse problems $\mathbf{g} = \mathbf{H}\mathbf{f} + \boldsymbol{\epsilon}$, where we know the forward model $\mathbf{H}$ and we assume the Gaussian likelihood $p(\mathbf{g}|\mathbf{f}) = \mathcal{N}(\mathbf{g}|\mathbf{H}\mathbf{f}, \sigma_\epsilon^2\mathbf{I})$ and the Gaussian prior $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|0, \sigma_f^2\mathbf{I})$ are the easiest cases to consider, where we know that the posterior is Gaussian $p(\mathbf{f}|\mathbf{g}) = \mathcal{N}(\mathbf{f}|\widehat{\mathbf{f}}, \widehat{\boldsymbol{\Sigma}})$ with

$$\widehat{\mathbf{f}} = (\mathbf{H}^t\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^t\mathbf{g} = \mathbf{A}\,\mathbf{g} = \mathbf{B}\mathbf{H}^t\,\mathbf{g} \text{ or still } \widehat{\mathbf{f}} = \mathbf{H}^t(\frac{1}{\lambda}\mathbf{H}\mathbf{H}^t + \mathbf{I})^{-1}\mathbf{g} = \mathbf{H}^t\mathbf{C}\,\mathbf{g},$$

where $\lambda = \sigma_\epsilon^2/\sigma_f^2$, $\mathbf{A} = (\mathbf{H}^t\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^t$, $\mathbf{B} = (\mathbf{H}^t\mathbf{H} + \lambda\mathbf{I})^{-1}$, $\mathbf{C} = (\frac{1}{\lambda}\mathbf{H}\mathbf{H}^t + \mathbf{I})^{-1}$, and $\widehat{\boldsymbol{\Sigma}} = (\mathbf{H}^t\mathbf{H} + \lambda\mathbf{I})^{-1}$.

These relations can be presented schematically as

$$\mathbf{g} \rightarrow \boxed{\mathbf{A}} \rightarrow \widehat{\mathbf{f}}, \qquad \mathbf{g} \rightarrow \boxed{\mathbf{H}^t} \rightarrow \boxed{\mathbf{B}} \rightarrow \widehat{\mathbf{f}}, \qquad \mathbf{g} \rightarrow \boxed{\mathbf{C}} \rightarrow \boxed{\mathbf{H}^t} \rightarrow \widehat{\mathbf{f}}.$$

We can then consider replacing $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ with appropriate deep neural networks and apply all the previous BDL methods to them. As we can see, these relations directly induce a linear feed-forward NN structure. In particular, if $\mathbf{H}$ represents a convolution operator, then $\mathbf{H}^t$, $\mathbf{H}^t\mathbf{H}$, and $\mathbf{H}\mathbf{H}^t$ are too, as well as the operators $\mathbf{B}$ and $\mathbf{C}$. Thus, the whole inversion can be modeled using a CNN [15,16].

For the case of computed tomography (CT), the first operation is equivalent to an analytic inversion, the second corresponds to back projection that is first followed by 2D filtering in the image domain, and the third corresponds to the famous filtered back projection (FBP), which is implemented on classical CT scans. These cases are illustrated in Figure 2.

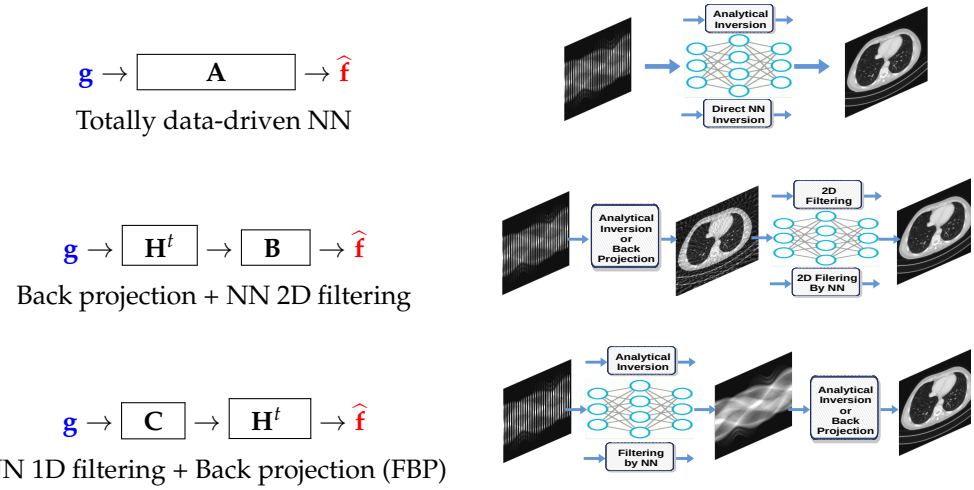

Figure 2. Three linear NN structures, which are derived directly from quadratic regularization inversion method. The right part of this figure is adapted from [16].

5.2. Second Example: A Deep Learning Equivalence of Iterative Gradient-Based Algorithms

One of the classical iterative methods in linear inverse problem algorithms is based on the gradient descent method to optimize $J(\mathbf{f}) = \|\mathbf{g} - \mathbf{H}\mathbf{f}\|^2$:

$$\mathbf{f}^{(k+1)} = \mathbf{f}^{(k)} + \alpha \mathbf{H}^t(\mathbf{g} - \mathbf{H}\mathbf{f}^{(k)}) = \alpha \mathbf{H}^t\mathbf{g} + (\mathbf{I} - \alpha \mathbf{H}^t\mathbf{H})\mathbf{f}^{(k)}, \tag{14}$$

where the solution to the problem is obtained recursively. Everybody knows that when the forward model operator $\mathbf{H}$ is singular or ill-conditioned, this iterative algorithm starts by converging, but it may diverge easily. One of the experimental methods to obtain an acceptable approximate solution is just to stop the iterations after K iterations. This idea can be translated to a deep learning NN by using K layers. Each layer represents one iteration of the algorithm. See Figure 3.

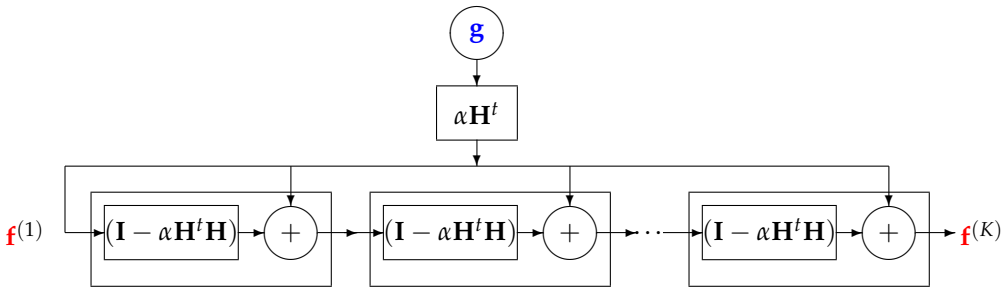

Figure 3. A K layers DL NN equivalent to K iterations of the basic optimization algorithm.

This DL structure can easily be extended to a regularized criterion $J(\mathbf{f}) = \frac{1}{2}\|\mathbf{g} - \mathbf{H}\mathbf{f}\|^2 + \lambda\|\mathbf{D}\mathbf{f}\|^2$, which can also be interpreted as the MAP or posterior mean solution with a Gaussian likelihood and prior. In this case, we have the following:

$$\mathbf{f}^{(k+1)} = \mathbf{f}^{(k)} + \alpha[\mathbf{H}^t(\mathbf{g} - \mathbf{H}\mathbf{f}^{(k)}) - \lambda \mathbf{D}^t\mathbf{D}] = \alpha \mathbf{H}^t\mathbf{g} + (\mathbf{I} - \alpha \mathbf{H}^t\mathbf{H} - \alpha\lambda \mathbf{D}^t\mathbf{D})\mathbf{f}^{(k)}. \tag{15}$$

We just need to replace $(\mathbf{I} - \alpha \mathbf{H}^t\mathbf{H})$ with $(\mathbf{I} - \alpha \mathbf{H}^t\mathbf{H} - \alpha\lambda \mathbf{D}^t\mathbf{D})$.

This structure can also be extended to all the sparsity-enforcing regularization terms such as ℓ_1 and the total variation (TV) using appropriate algorithms such as the ISTA (iterative soft threshold algorithm) or its fast version FISTA by replacing the update expression and by adding an NL operation, much like the ordinary NNs. A simple example is given in the following subsection.

5.3. Third Example: ℓ_1 Regularization and NN

The case of the MAP solution with a Gaussian likelihood and double exponential prior becomes equivalent to the ℓ_1 regularization criterion:

$$J(\mathbf{f}) = \|\mathbf{g} - \mathbf{H}\mathbf{f}\|_2^2 + \lambda\|\mathbf{f}\|_1\,, \tag{16}$$

where the solution can be obtained with an iterative optimization algorithm, such as ISTA:

$$\mathbf{f}^{(k+1)} = Prox_{\ell_1}\left(\mathbf{f}^{(k)}, \lambda\right) \stackrel{\triangle}{=} \mathcal{S}_{\lambda\alpha}\left(\alpha\mathbf{H}^t\mathbf{g} + (\mathbf{I} - \alpha\mathbf{H}^t\mathbf{H})\mathbf{f}^{(k)}\right), \tag{17}$$

where $\mathcal{S}_\theta$ is a soft threshold operator, and $\alpha \leq |eig(\mathbf{H}^t\mathbf{H})|$ is the Lipschitz constant of the normal operator. When $\mathbf{H}$ is a convolution operator, then

- $(\mathbf{I} - \alpha\mathbf{H}^t\mathbf{H})\mathbf{f}^{(k)}$ can also be approximated by a convolution and thus considered as a filtering operator;
- $\frac{1}{\alpha}\mathbf{H}^t\mathbf{g}$ can be considered as a bias term and is also a convolution operator;
- $\mathcal{S}_{\theta=\lambda\alpha}$ is a nonlinear pointwise operator. In particular, when $\mathbf{f}$ is a positive quantity, this soft threshold operator can be compared to the ReLU activation function of the NN. See Figure 4.

Using the iterative gradient-based algorithm with a fixed number of iterations for computing a GI or a regularized one, as explained in the previous section, can be used to propose a DL structure with K layers, with K being the number of iterations before stopping. Figure 5 shows this structure for a quadratic regularization, which results in a linear NN, and Figure 6 shows the case of ℓ_1 regularization.

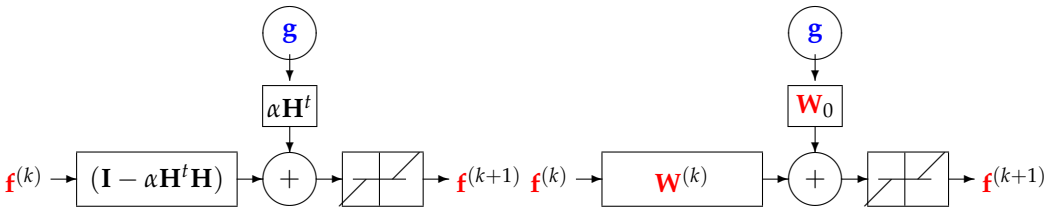

Figure 4. A K layers DL NN equivalent to K iterations of a basic gradient-based optimization algorithm. A quadratic regularization results in a linear NN, while a ℓ_1 regularization results in a classical NN with a nonlinear activation function. Left: supervised case. Right: unsupervised case. In both cases, all the K layers have the same structure.

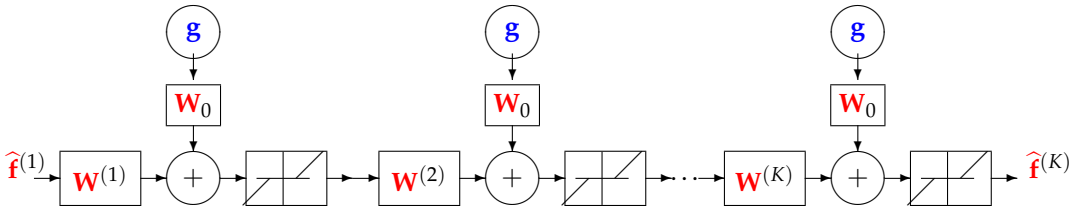

Figure 5. All the K layers of DL NN equivalent to K iterations of an iterative gradient-based optimization algorithm. The simplest solution is to choose $\mathbf{W}_0 = \alpha\mathbf{H}$ and $\mathbf{W}^{(k)} = \mathbf{W} = (\mathbf{I} - \alpha\mathbf{H}^t\mathbf{H})$, $k = 1, \cdots, K$. A more robust, but more costly approach, is to learn all the layers for $\mathbf{W}^{(k)} = (\mathbf{I} - \alpha^{(k)}\mathbf{H}^t\mathbf{H})$, $k = 1, \cdots, K$.

In all these examples, we could directly obtain the structure of the NN from the forward model and known parameters. However, in these approaches, there are some difficulties that consist of the determination of the structure of the NN. For example, in the first example, obtaining the structure of $\mathbf{B}$ depends on the regularization parameter λ. The same difficulty arises in determining the shape and the threshold level of the threshold bloc of the network in the second example. The same need for the regularization parameter, as well as many other hyperparameters, makes it necessary to create the NN structure and weights. In practice, we can decide, for example, on the number and structure of a DL network, but as their corresponding weights depend on many unknown or difficult-to-fix parameters, ML may become of help. In the following, we first consider the training part of a general ML method. Then, we will see how to include the physics-based knowledge of the forward model in the structure of learning [4,10,17,18].

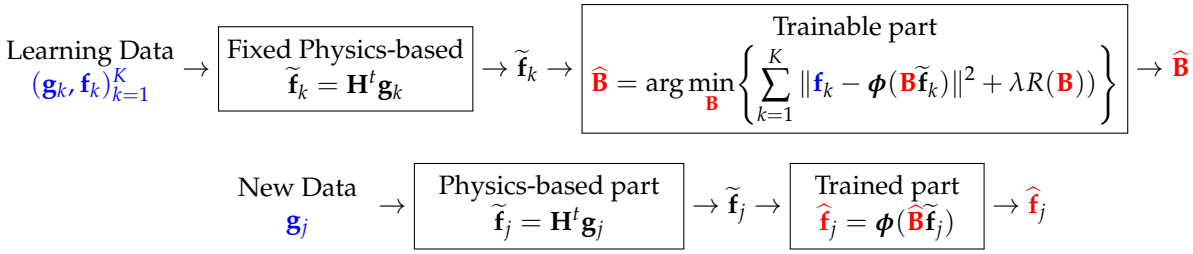

Figure 6. Training (**top**) and testing (**bottom**) steps in the first use of physics-based ML approach.

5.4. Decomposition of the NN Structure to Fixed and Trainable Parts

The first easiest and most understandable method consists of decomposing the structure of the network $\mathbf{W}$ in two parts: a fixed part and a learnable part. As the simplest example, we can consider the case of analytical expression of the quadratic regularization $\widehat{\mathbf{f}} = (\mathbf{H}\mathbf{H}^t + \lambda\mathbf{D}\mathbf{D}^t)^{-1}\mathbf{H}^t\mathbf{g} = \mathbf{B}\mathbf{H}^t\mathbf{g}$, which suggests having a two-layer network with a fixed part structure $\mathbf{H}^t$ and a trainable one $\mathbf{B} = (\mathbf{H}\mathbf{H}^t + \lambda\mathbf{D}\mathbf{D}^t)^{-1}$. See Figure 6.

It is interesting to note that in X-ray-computed tomography (CT), the forward operator $\mathbf{H}$ is called *projection*, the adjoint operator $\mathbf{H}^t$ is called *back projection (BP)*, and the $\mathbf{B}$ operator is assimilated to a 2D filtering (convolution).

6. DL Structure and Deterministic or Bayesian Computation

To be able to look at the DNN and analyze it either in a deterministic or Bayesian manner, let us come back to the general notations and consider the following NN with input $\mathbf{x}$, output $\mathbf{y}$, and intermediate hidden variables $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_L$:

$$\mathbf{x} \rightarrow \boxed{\text{first layer}} \rightarrow \mathbf{z}_1 \rightarrow \boxed{\text{second layer}} \rightarrow \mathbf{z}_2 \cdots \rightarrow \boxed{\text{last layer}} \rightarrow \mathbf{y}.$$

In the deterministic case, each layer is defined by its parameters (W_l, b_l):

$$\mathbf{x} \rightarrow \boxed{f_{W_0,b_0}(\mathbf{x})} \rightarrow \mathbf{z}_1 \rightarrow \boxed{f_{W_1,b_1}(\mathbf{z}_1)} \rightarrow \mathbf{z}_2 \cdots \rightarrow \boxed{f_{W_L,b_L}(\mathbf{z}_{L-1})} \rightarrow \mathbf{y},$$

and we can write:

$$\mathbf{y} = f_{\boldsymbol{\theta}}(\mathbf{x}) = (f_{(W_0,b_0)} \odot f_{(W_1,b_1)} \odot \cdots f_{(W_L,b_L)})(\mathbf{x})$$

with

$$\boldsymbol{\theta} = ((W_0, b_0), (W_1, b_1), \cdots, (W_L, b_L))$$

and

$$\mathbf{y} = f_L(W_L\mathbf{z}_{L-1} + b_L), \quad \mathbf{z}_l = f_l(W_l\mathbf{z}_{l-1} + b_l), \quad l = L, \cdots, 1, \quad \mathbf{z}_0 = \mathbf{x}. \tag{18}$$

6.1. Deterministic DL Computation

In general, during the training steps, the parameters $\boldsymbol{\theta}$ are estimated via

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left\{ \sum_i \ell(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \sum_k \lambda_k \phi_k(W_k, b_k) \right\}. \tag{19}$$

The main point here is to choose how to choose λ_k and $\phi(.)$, as well as which optimization algorithm to choose for better convergence.

When parameters are obtained (the model is trained), we can use it easily via

$$\mathbf{z}_0 = \mathbf{x}, \quad \mathbf{z}_l = f_l(W_l \mathbf{z}_{l-1} + b_l), \quad \mathbf{y} = f_L(W_L \mathbf{z}_{L-1} + b_L). \tag{20}$$

6.2. Bayesian Deep Neural Network

In Bayesian DL, the question of choosing λ_k and $\phi(.)$ in the previous case becomes the choice of the prior, $p(\boldsymbol{\theta})$, which can also be assumed to be a priori separable in the components of $\boldsymbol{\theta}$ or not. Then, we have to choose the expression of the likelihood (in general Gaussian) and find the expression of the posterior $p(\boldsymbol{\theta}|\mathcal{D})$. As explained extensively before, directly using this posterior is almost impossible. Hopefully, we have a great number of approximate computation methods, such as MCMC sampling, slice sampling, nested sampling, data augmentation, and variational inference, which can still be used in practical situations. However, the training step in the Bayesian DL still stays very costly, particularly if we want to quantify the uncertainties.

Prediction Step

In the prediction step, we again have to consider choosing a probability law $p(\mathbf{x})$ for the class of the possible inputs and for all the outputs $\mathbf{z}_l$ that are conditional to their inputs $\mathbf{z}_{l-1}$. Then, we can consider, for example, the Gibbs sampling scheme. A comparison between deterministic and Bayesian DL is shown here:

Deterministic:
$$\begin{cases} \mathbf{z}_0 = \mathbf{x}, \\ \mathbf{z}_l = f_l(W_l \mathbf{z}_{l-1} + b_l), \quad l = 1, \cdots, L, \\ \mathbf{y} = f_L(W_L \mathbf{z}_{L-1} + b_L). \end{cases}$$

$\rightarrow$ Bayesian:
$$\begin{cases} \mathbf{z}_0 \sim p(\mathbf{x}), \\ \mathbf{z}_l \sim p(\mathbf{z}_l|\mathbf{z}_{l-1}), \quad l = 1, \cdots, L, \\ \mathbf{y} \sim p(\mathbf{y}|\mathbf{z}_{L-1}). \end{cases}$$

If we consider Gaussian laws for the input and all the conditional variables, then we can write the following:

$$\mathbf{z}_0 \sim N(\mathbf{z}_0|\mathbf{x}, \tau_0 \mathbf{I}), \qquad \mathbf{z}_l \sim N(\mathbf{z}_l|f_l(\mathbf{z}_{l-1}), \tau_l \mathbf{I}), \; l = 1, \cdots, L, \qquad \mathbf{y} \sim p(\mathbf{y}|\mathbf{z}_{L-1}). \tag{21}$$

Here too, the main difficulty occurs when there are nonlinear activation functions, particularly in the last layer, where the Gaussian approximation may not be more valid.

7. Application: Infrared Imaging

Infrared (IR) imaging is used to diagnose and to survey the temperature field distribution of sensitive objects in many industrial applications. These images are, in general, low resolution and very noisy. The real values of the temperature also depend on many other parameters, such as emissivity, attenuation, and diffusion, due to the distance of the camera to the sources. To be really useful in practice, we need to reduce the noise, calibrate and increase the resolution, segment for detection of the hot area, and finally survey the temperature values of the different areas during the time to be able to conduct preventive diagnosis and possible maintenance.

Reducing the noise can be accomplished by filtering using the Fourier transform, wavelet transform, or other sparse representations of images. To increase the resolution, we may use deconvolution methods if we can obtain the point spread function (PSF) of the camera, or if not by using blind deconvolution techniques. The segmentation and detection of the hot area and the temperature value estimation at each area are also very important steps in real applications. Any of these steps can be performed separately, but trying to propose global processing using DL or BDL is necessary for real applications. As any of these steps are in fact different inverse problems, and it is difficult to fix the parameters in each step in a robust way, we propose a global process using the BDL. See the global scheme in Figure 7.

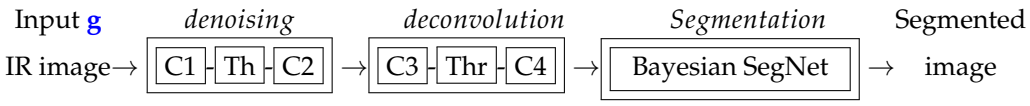

Figure 7. The proposed four groups of layers of NN for denoising, deconvolution, and segmentation of IR images.

In the first step, as the final objective is to segment the image to obtain different levels of temperature (for example, four levels: background, normal, high, and very high), we propose to design an NN that obtains as input a low resolution and noisy image and outputs a segmented image with those four levels and, at the same time, a good estimate of the temperature at each segment. See Figure 8.

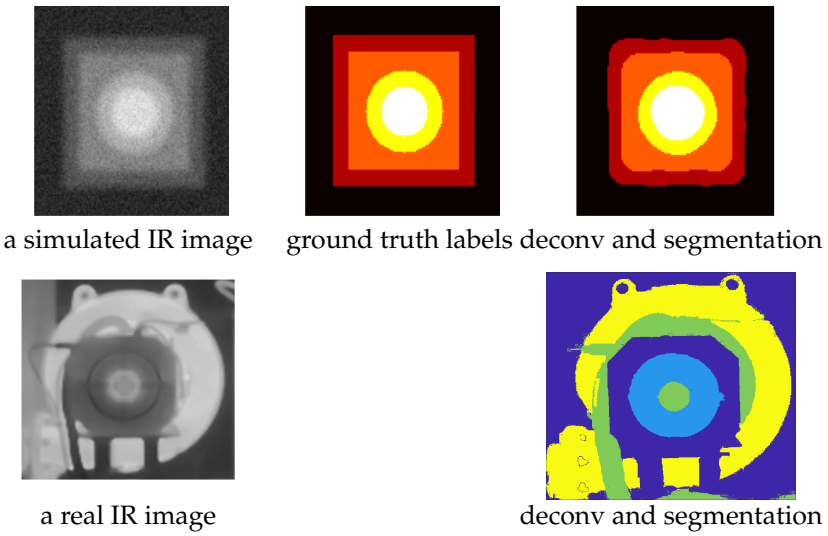

Figure 8. Example of expected results in deterministic methods. First row: a simulated IR image (**left**), its ground truth labels (**middle**), and the result of the deconvolution and segmentation (**right**). Second row: a real IR image (**left**) and the result of its deconvolution and segmentation (**right**).

To train this NN, we can generate different known shaped images to consider as the ground truth and simulate the blurring effects of temperature diffusions via the convolution of different appropriate PSFs. We can also add some noise to generate realistic images. We can also use black body thermal sources and acquire different images at different conditions. All these images can be used for the training of the network. See an example of the obtained result in Figure 9.

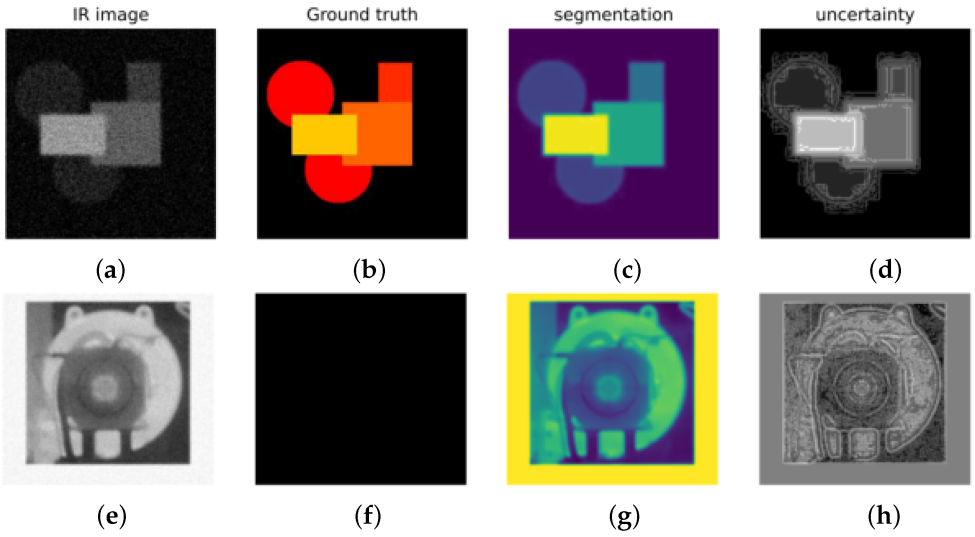

Figure 9. Example of expected results in Bayesian methods. First row from left: (**a**) simulated IR image, (**b**) its ground truth labels, (**c**) the result of the deconvolution and segmentation, and (**d**) uncertainties. Second row: (**e**) a real IR image, (**f**) no ground truth, (**g**) the result of its deconvolution and segmentation, and (**h**) uncertainties.

Author Contributions: Conceptualization and methodology, A.M.-D., N.C., L.W. and L.Y.; software, A.M.-D.; validation, A.M.-D., N.C., L.W. and L.Y.; formal analysis, A.M.-D.; investigation, resources, and data curation, N.C.; writing—original draft preparation, writing—review and editing, visualization and supervision, A.M.-D.; project administration and funding acquisition, N.C. and A.M.-D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable to this article.

Conflicts of Interest: The authors Ali Mohammad-Djafari and Ning Chu are the scientific researchers of Zhejiang Shangfeng Special Blower Company. The authors declare no conflict of interest.

References

1. Ayasso, H.; Mohammad-Djafari, A. Joint NDT Image Restoration and Segmentation Using Gauss-Markov-Potts Prior Models and Variational Bayesian Computation. *IEEE Trans. Image Process.* **2010**, *19*, 2265–2277. [CrossRef] [PubMed]
2. Tang, J.; Egiazarian, K.; Golbabaee, M.; Davies, M. The Practicality of Stochastic Optimization in Imaging Inverse Problems. *IEEE Trans. Comput. Imaging* **2020**, *6*, 1471–1485. [CrossRef]
3. Zhu, Y.; Zabaras, N. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *J. Comput. Phys.* **2018**, *366*, 415–447. [CrossRef]
4. McCann, M.T.; Jin, K.H.; Unser, M. A Review of Convolutional Neural Networks for Inverse Problems in Imaging. *Image Video Process.* **2017**, *34*, 85–95. [CrossRef]
5. Fang, Z. A High-Efficient Hybrid Physics-Informed Neural Networks Based on Convolutional Neural Network. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 5514–5526. [CrossRef] [PubMed]
6. Gilton, D.; Ongie, G.; Willett, R. Neumann Networks for Linear Inverse Problems in Imaging. *IEEE Trans. Comput. Imaging* **2020**, *6*, 328–343. [CrossRef]
7. Gong, D.; Zhang, Z.; Shi, Q.; van den Hengel, A.; Shen, C.; Zhang, Y. Learning Deep Gradient Descent Optimization for Image Deconvolution. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 5468–5482. [CrossRef] [PubMed]
8. De Haan, K.; Rivenson, Y.; Wu, Y.; Ozcan, A. Deep-Learning-Based Image Reconstruction and Enhancement in Optical Microscopy. *Proc. IEEE* **2020**, *108*, 30–50. [CrossRef]
9. Aggarwal, H.K.; Mani, M.P.; Jacob, M. MoDL: Model-Based Deep Learning Architecture for Inverse Problems. *IEEE Trans. Med. Imaging* **2019**, *38*, 394–405. [CrossRef] [PubMed]
10. Chen, Y.; Lu, L.; Karniadakis, G.E.; Negro, L.D. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Opt. Express* **2020**, *28*, 11618–11633. [CrossRef] [PubMed]

11. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
12. Mohammad-Djafari, A. Hierarchical Markov modeling for fusion of X-ray radiographic data and anatomical data in computed tomography. In Proceedings of the IEEE International Symposium on Biomedical Imaging, Washington, DC, USA, 7–10 July 2002; pp. 401–404. [CrossRef]
13. Mohammad-djafari, A. Regularization, Bayesian Inference and Machine Learning methods for Inverse Problems. *Entropy* **2021**, *23*, 1673. [CrossRef] [PubMed]
14. Giordano, R.; Broderick, T.; Jordan, M. Linear response methods for accurate covariance estimates from mean field variational Bayes. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015.
15. Chun, I.Y.; Huang, Z.; Lim, H.; Fessler, J. Momentum-Net: Fast and convergent iterative neural network for inverse problems. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *45*, 4915–4931. [CrossRef] [PubMed]
16. Lucas, A.; Iliadis, M.; Molina, R.; Katsaggelos, A.K. Using Deep Neural Networks for Inverse Problems in Imaging: Beyond Analytical Methods. *IEEE Signal Process. Mag.* **2018**, *35*, 20–36. [CrossRef]
17. Chang, J.H.R.; Li, C.L.; Poczos, B.; Kumar, B.V.K.V.; Sankaranarayanan, A.C. One Network to Solve Them All—Solving Linear Inverse Problems using Deep Projection Models. In Proceedings of the IEEE International Conference on Computer Vision 2017, Venice, Italy, 22–29 October 2017.
18. Liang, D.; Cheng, J.; Ke, Z.; Ying, L. Deep Magnetic Resonance Image Reconstruction: Inverse Problems Meet Neural Networks. *IEEE Signal Process. Mag.* **2020**, *37*, 141–151. [CrossRef] [PubMed]