remote sensing

MDPI

Article

An Overview of Neural Network Methods for Predicting Uncertainty in Atmospheric Remote Sensing

Adrian Doicu [1,*], Alexandru Doicu [2], Dmitry S. Efremenko [1], Diego Loyola [1] and Thomas Trautmann [1]

[1] Remote Sensing Technology Institute, German Aerospace Center (DLR), 82234 Oberpfaffenhofen, Germany; dmitry.efremenko@dlr.de (D.S.E.); diego.loyola@dlr.de (D.L.); thomas.trautmann@dlr.de (T.T.)
[2] Institute of Mathematics, University of Augsburg, 86159 Augsburg, Germany; alexandru.doicu@math.uni-augsburg.de
* Correspondence: Adrian.Doicu@dlr.de

Abstract: In this paper, we present neural network methods for predicting uncertainty in atmospheric remote sensing. These include methods for solving the direct and the inverse problem in a Bayesian framework. In the first case, a method based on a neural network for simulating the radiative transfer model and a Bayesian approach for solving the inverse problem is proposed. In the second case, (i) a neural network, in which the output is the convolution of the output for a noise-free input with the input noise distribution; and (ii) a Bayesian deep learning framework that predicts input aleatoric and model uncertainties, are designed. In addition, a neural network that uses assumed density filtering and interval arithmetic to compute uncertainty is employed for testing purposes. The accuracy and the precision of the methods are analyzed by considering the retrieval of cloud parameters from radiances measured by the Earth Polychromatic Imaging Camera (EPIC) onboard the Deep Space Climate Observatory (DSCOVR).

Keywords: neural networks; interval arithmetic; radiative transfer

Citation: Doicu, A.; Doicu, A.; Efremenko, D.S.; Loyola, D.; Trautmann, T. An Overview of Neural Network Methods for Predicting Uncertainty in Atmospheric Remote Sensing. *Remote Sens.* **2021**, *13*, 5061. https://doi.org/10.3390/rs13245061

Academic Editors: Xiaoli Li, Zhenghua Chen, Min Wu and Jianfei Yang

Received: 4 November 2021
Accepted: 10 December 2021
Published: 13 December 2021

1. Introduction

In atmospheric remote sensing, the retrieval of atmospheric parameters is an inverse problem that is usually ill posed. Due to its ill posedness, measurement errors can lead to large errors in the retrieved quantities. It is therefore desirable to characterize the retrieved value by an estimate of uncertainty describing a range of values that probably produce a measurement [1].

The retrieval algorithms are mostly based on deterministic or stochastic optimization methods. From the first category, the method of Tikhonov regularization and iterative regularization methods deserve to be mentioned, while from the second category, the Bayesian methods are the most representative. The Bayesian framework [2,3] provides an efficient way to deal with the ill-posedness of the inverse problem and its uncertainties. In this case, the solution of the inverse problem is given by the a posteriori distribution (the conditional distribution of the retrieval quantity given the measurement) that accounts for all assumed retrieval uncertainties. Under the assumptions that (i) the a priori knowledge and the measurement uncertainty are both Gaussian distributions, and (ii) the forward model is moderately nonlinear, the a posteriori distribution is approximately Gaussian with a covariance matrix that can be analytically calculated. However, even neglecting the validity of these assumptions, the method is not efficient for the operational processing of large data volumes. The reason is that the computations of the forward model and its Jacobian are expensive computational processes.

Compared to Bayesian methods, neural networks are powerful tools for the design of efficient retrieval algorithms. Their capability to approximate any continuous function on a compact set to an arbitrary accuracy makes them well suited to approximate the input–output function represented by a radiative transfer model. An additional feature is

that the derivatives of a neural network model with respect to its inputs can be analytically computed. Thus, a neural network algorithm can produce, in addition to the approximation of the radiative quantities of interest, also an estimation of their derivatives with respect to the model inputs. While the training of a neural network may require a significant amount of time, a trained neural network may deliver accurate predictions of the forward model and its Jacobian, typically in a fraction of a millisecond.

Neural networks were initially developed for the emulation of radiative transfer models [4–8] and subsequently for atmospheric remote sensing. The latter include:

1. Neural networks that are used to emulate the forward model and are applied in conjunction with a Bayesian approach to solve the inverse problem [9–12];
2. Neural networks that are developed to directly learn the retrieval mappings from data [13–20]; and
3. Neural networks that are designed to directly invert the atmospheric parameters of interest which are then used as initial values in an optimization algorithm, e.g., the method of Tikhonov regularization [21,22].

In the first case, the total uncertainty sums up the contributions of the uncertainties in the data and in the neural network model, whereby the model uncertainty is computed from the statistics of the errors over the data set. However, the retrieval algorithm is still based on the assumption that the forward model is nearly linear, which is generally not true. In the second case, the probabilistic character of the inverse problem is neglected and uncertainty estimates are provided as mean errors computed over the data set; this is a disadvantage compared to Bayesian methods. Remarkable exceptions to the approaches listed above are the works of Aires [23,24], in which a Bayesian framework in conjunction with the Laplace approximation was used to model the retrieval errors (estimated from the error covariance matrix observed on the data set), and of Pfreundschuh et al. [25], in which a quantile regression neural network was designed. Unfortunately, the Laplace approximation is only suitable for small training sets and simple networks, while quantile regression is suitable for the retrieval of scalar quantities (it is unclear whether a reasonable approximation of the quantile contours of the joint a posteriori distribution can be obtained).

This paper is devoted to an overview of neural networks methods for predicting uncertainty in atmospheric remote sensing. In addition to the method based on a neural network for simulating the radiative transfer model and a Bayesian approach for solving the inverse problem (Case 1 above), methods relying on Bayesian networks are described. These methods, in which the network activations and weights can be modeled by parametric probability distributions, are standard tools for uncertainty predictions in deep neural networks, and can be applied to nonlinear retrieval problems, and to the best knowledge of the authors, have not yet been used in atmospheric remote sensing. The paper, which is merely of pedagogical nature, is organized as follows. In Section 1, we present the theoretical background of this study. This is then used in Section 2 to develop several neural network methods and apply them to a specific cloud parameters retrieval problem. Section 3 contains a few concluding remarks.

2. Theoretical Background

We consider a generic model $\mathbf{y} = \mathbf{F}(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^{N_x}$ is the input vector, $\mathbf{F}$ is some deterministic function and $\mathbf{y} \in \mathbb{R}^{N_y}$ is the output vector. For an atmosphere characterized by a set of state parameters, the signal measured by an instrument at different wavelengths can be computed by a radiative transfer model $\mathbf{R}$. Specifically, we will refer to the measurement signals as data, and split the vector of state parameters in (i) the vector of atmospheric parameters that are intended to be retrieved, and (ii) the vector of atmospheric parameters that are known to have some accuracy, but are not included in the retrieval (forward model parameters). In this study, we will use neural networks to model the radiative transfer function $\mathbf{R}$, as well as its inverse $\mathbf{R}^{-1}$. In this regard and in order to simplify the notation, we will consider two cases. In the first case, referred to as the direct problem, the input $\mathbf{x}$ is the set of atmospheric parameters, the output $\mathbf{y}$ is the set of data and the forward model $\mathbf{F}$

coincides with the radiative transfer model $\mathbf{R}$, while in the second case, referred to as the inverse problem, the situation is reversed: the input $\mathbf{x}$ includes the sets of data and forward model parameters, while the output $\mathbf{y}$ includes the set of atmospheric parameters to be retrieved (in the absence of forward model parameters, the forward model $\mathbf{F}$ reproduces the inverse of the radiative transfer model $\mathbf{R}^{-1}$).

In machine learning, the task is to approximate $\mathbf{F}(\mathbf{x})$ by a neural network model $\mathbf{f}(\mathbf{x}, \boldsymbol{\omega})$ characterized by a set of parameters $\boldsymbol{\omega}$ [26,27]. For doing this, we consider a set of inputs $\mathbf{X} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ and a corresponding set of outputs $\mathbf{Y} = \{\mathbf{y}^{(n)}\}_{n=1}^{N}$, given by $\mathbf{y}^{(n)} = \mathbf{F}(\mathbf{x}^{(n)})$, where N is the number of samples. In a regression problem, $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$ forms a data set—or more precisely, a training set—from which the neural network model $\mathbf{f}(\mathbf{x}, \boldsymbol{\omega})$ can be inferred. Traditional neural networks are comprised of units or nodes arranged in an input layer, an output layer, and a number of hidden layers situated between the input and output layers. Let $L+1$ be the number of layers, and N_l be the number of units in layer l, where $l = 0, \ldots, L$. The input layer corresponds to $l = 0$, the output layer to $l = L$, so that $N_{\mathrm{x}} = N_0$ and $N_{\mathrm{y}} = N_L$. In feed-forward networks, the signals $y_{i,l-1}$ from units $i = 1, \ldots, N_{l-1}$ in layer $l-1$ are multiplied by a set of weights $w_{ji,l}$, $j = 1, \ldots, N_l$, $i = 1, \ldots, N_{l-1}$, and then summed and combined with a bias $b_{j,l}$, $j = 1, \ldots, N_l$. This calculation forms the pre-activation signal $u_{j,l} = \sum_{i=1}^{N_{l-1}} w_{ji,l} y_{i,l-1} + b_{j,l}$ which is transformed by the layer activation function g_l to form the activation signal $y_{j,l}$ of unit $j = 1, \ldots, N_l$ in layer l. Defining the matrix of weights $\mathsf{W}_l \in \mathbb{R}^{N_l \times N_{l-1}}$ and the vector of biases $\mathbf{b}_l \in \mathbb{R}^{N_l}$ by $[\mathsf{W}_l]_{ji} = w_{ji,l}$ and $[\mathbf{b}_l]_j = b_{j,l}$, respectively, and letting $\boldsymbol{\omega} = \{\mathsf{W}_l, \mathbf{b}_l\}_{l=1}^{L}$ be the set of network parameters, the feed-forward operations can be written in matrix form as

$$\mathbf{y}_0 = \mathbf{x}, \tag{1}$$

$$\mathbf{u}_l = \mathsf{W}_l \mathbf{y}_{l-1} + \mathbf{b}_l, \tag{2}$$

$$\mathbf{y}_l = g_l(\mathbf{u}_l), \quad l = 1, \ldots, L, \tag{3}$$

$$\mathbf{f}(\mathbf{x}, \boldsymbol{\omega}) = \mathbf{y}_L, \tag{4}$$

where $[\mathbf{y}_l]_i = y_{i,l}$ and $[\mathbf{u}_l]_j = u_{j,l}$. Deep learning is the process of regressing the network parameters $\boldsymbol{\omega}$ on the data $\mathcal{D}$. The standard procedure is to compute a point estimate $\widehat{\boldsymbol{\omega}}$ as the minimizer of some loss function by using the back-propagation algorithm [28]. In a stochastic framework, the loss function is usually defined as the log likelihood of the data set, with an eventual regularization term to penalize the network parameters. From a statistical point of view, this procedure is equivalent to a maximum a posteriori (MAP) estimation when regularization is used, and maximum likelihood estimation (MLE) when this is not the case.

In this section, we review the basic theory which serves as a basis for the development of different neural network architectures. In particular, we describe (i) the methodology for computing point estimates; (ii) the different types of uncertainty; and (iii) Bayesian networks.

2.1. Point Estimates

A data model with output noise is given by

$$\mathbf{y} = \mathbf{f}(\mathbf{x}, \boldsymbol{\omega}) + \boldsymbol{\delta}_{\mathbf{y}}, \tag{5}$$

$$\boldsymbol{\delta}_{\mathbf{y}} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_{\mathbf{y}}^{\delta}), \tag{6}$$

where here and in the following, the notation $\mathcal{N}(\mathbf{x}; \overline{\mathbf{x}}, \mathsf{C}_{\mathbf{x}})$, or more simply and when no confusion arises, $\mathcal{N}(\overline{\mathbf{x}}, \mathsf{C}_{\mathbf{x}})$ stands for a Gaussian distribution with the mean $\overline{\mathbf{x}}$ and covariance matrix $\mathsf{C}_{\mathbf{x}}$. When the true input $\mathbf{x}$ is hidden (so that it cannot be observed) but

samples from a random vector $\mathbf{z} = \mathbf{x} + \boldsymbol{\delta}_\mathbf{x}$ with $\boldsymbol{\delta}_\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_\mathbf{x}^\delta)$, i.e., $p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mathbf{x}, \mathsf{C}_\mathbf{x}^\delta)$ are available, the pertinent model is the data model with input and output noise, that is:

$$\mathbf{y} = \mathbf{f}(\mathbf{z}, \boldsymbol{\omega}) + \boldsymbol{\Delta}_\mathbf{y}, \tag{7}$$

$$\boldsymbol{\Delta}_\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathscr{C}_\mathbf{y}^\delta). \tag{8}$$

The error $\boldsymbol{\Delta}_\mathbf{y}$ sums up the contributions of the output error and of the input error that propagates through the network in the output space. Specifically, when the noise process in the input space is small and the linearization:

$$\mathbf{f}(\mathbf{x}, \boldsymbol{\omega}) = \mathbf{f}(\mathbf{z}, \boldsymbol{\omega}) + \mathsf{K}_\mathbf{x}(\mathbf{z}, \boldsymbol{\omega})(\mathbf{x} - \mathbf{z}), \tag{9}$$

$$\mathsf{K}_\mathbf{x}(\mathbf{z}, \boldsymbol{\omega}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{z}, \boldsymbol{\omega}), \tag{10}$$

is assumed, we find (cf. Equations (5), (7), and (9)) $\boldsymbol{\Delta}_\mathbf{y} = \boldsymbol{\delta}_\mathbf{y} + \mathsf{K}_\mathbf{x}(\mathbf{z}, \boldsymbol{\omega})(\mathbf{x} - \mathbf{z})$, and further:

$$\mathscr{C}_\mathbf{y}^\delta(\mathbf{z}, \boldsymbol{\omega}) = \mathsf{C}_\mathbf{y}^\delta + \mathsf{K}_\mathbf{x}(\mathbf{z}, \boldsymbol{\omega})\mathsf{C}_\mathbf{x}^\delta\mathsf{K}_\mathbf{x}^T(\mathbf{z}, \boldsymbol{\omega}). \tag{11}$$

To design a neural network, we consider a data set $\mathcal{D}$ associated to each data model, meaning that:

1. An exact data set $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$, where $\mathbf{y}^{(n)} = \mathbf{F}(\mathbf{x}^{(n)})$;
2. A data set with output noise $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$, where $\mathbf{y}^{(n)} = \mathbf{F}(\mathbf{x}^{(n)}) + \boldsymbol{\delta}_\mathbf{y}$; and
3. A data set with input and output noise $\mathcal{D} = \{(\mathbf{z}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$, where $\mathbf{z}^{(n)} = \mathbf{x}^{(n)} + \boldsymbol{\delta}_\mathbf{x}$ and $\mathbf{y}^{(n)} = \mathbf{F}(\mathbf{x}^{(n)}) + \boldsymbol{\delta}_\mathbf{y}$.

In a stochastic framework, a neural network can be regarded as a probabilistic model $p(\mathbf{y}|\mathbf{z}, \boldsymbol{\omega})$; given an observable input $\mathbf{z}$ and a set of parameters $\boldsymbol{\omega}$, a neural network assigns a probability to each possible output $\mathbf{y}$. In view of Equations (7) and (8), the a priori confidence in the predictive power of the model is given by

$$p(\mathbf{y}|\mathbf{z}, \boldsymbol{\omega}) = \mathcal{N}(\mathbf{y}; \mathbf{f}(\mathbf{z}, \boldsymbol{\omega}), \mathscr{C}_\mathbf{y}^\delta(\mathbf{z}, \boldsymbol{\omega})). \tag{12}$$

The process of learning from the data $\mathcal{D}$ can be described by the posterior $p(\boldsymbol{\omega}|\mathcal{D}) = p(\boldsymbol{\omega}|\mathbf{Z}, \mathbf{Y})$, which represents the Bayes plausibility for the parameters $\boldsymbol{\omega}$ given the data $\mathcal{D}$. This can be estimated by using the Bayesian rule:

$$p(\boldsymbol{\omega}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\omega})p(\boldsymbol{\omega})}{p(\mathcal{D})} \propto p(\mathcal{D}|\boldsymbol{\omega})p(\boldsymbol{\omega}) \propto \exp[-E(\boldsymbol{\omega})], \tag{13}$$

where $p(\mathcal{D}|\boldsymbol{\omega})$ is the likelihood or the probability of the data, $p(\boldsymbol{\omega})$ is the prior over the network parameters, $p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\omega})p(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega}$ the evidence, and:

$$E(\boldsymbol{\omega}) = E_\mathrm{D}(\boldsymbol{\omega}) + E_\mathrm{R}(\boldsymbol{\omega}) \tag{14}$$

is the loss function. The first term $E_\mathrm{D}(\boldsymbol{\omega})$ in the expression of the loss function $E(\boldsymbol{\omega})$ is the contribution from the likelihood $p(\mathcal{D}|\boldsymbol{\omega})$, written as the product (cf. Equation (12)):

$$p(\mathcal{D}|\boldsymbol{\omega}) = p(\mathbf{Y}|\mathbf{Z}, \boldsymbol{\omega}) = \prod_{n=1}^N p(\mathbf{y}^{(n)}|\mathbf{z}^{(n)}, \boldsymbol{\omega}) \propto \exp[-E_\mathrm{D}(\boldsymbol{\omega})], \tag{15}$$

$$E_\mathrm{D}(\boldsymbol{\omega}) = \frac{1}{2}\sum_{n=1}^N [\mathbf{y}^{(n)} - \mathbf{f}(\mathbf{z}^{(n)}, \boldsymbol{\omega})]^T [\mathscr{C}_\mathbf{y}^\delta(\mathbf{z}^{(n)}, \boldsymbol{\omega})]^{-1} [\mathbf{y}^{(n)} - \mathbf{f}(\mathbf{z}^{(n)}, \boldsymbol{\omega})], \tag{16}$$

while the second term $E_R(\boldsymbol{\omega})$ is the contribution from the prior $p(\boldsymbol{\omega})$, chosen for example, as the Gaussian distribution:

$$p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}; \mathbf{0}, \mathsf{C}_{\boldsymbol{\omega}}) \propto \exp[-E_R(\boldsymbol{\omega})], \tag{17}$$

$$E_R(\boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{\omega}^T \mathsf{C}_{\boldsymbol{\omega}}^{-1}\boldsymbol{\omega}. \tag{18}$$

In this regard, point estimates with regularization are computed by maximizing the posterior $p(\boldsymbol{\omega}|\mathcal{D})$:

$$\widehat{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\text{MAP}} = \arg\max_{\boldsymbol{\omega}} \log p(\boldsymbol{\omega}|\mathcal{D}) = \arg\min_{\boldsymbol{\omega}} E(\boldsymbol{\omega}), \tag{19}$$

while point estimates without regularization are computed by maximizing the likelihood $p(\mathcal{D}|\boldsymbol{\omega})$:

$$\widehat{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\text{MLE}} = \arg\max_{\boldsymbol{\omega}} \log p(\mathcal{D}|\boldsymbol{\omega}) = \arg\min_{\boldsymbol{\omega}} E_D(\boldsymbol{\omega}). \tag{20}$$

Some comments are in order.

1. For a data model with output noise, we have $\mathbf{z} = \mathbf{x}$, $\mathscr{C}_{\mathbf{y}}^{\delta} = \mathsf{C}_{\mathbf{y}}^{\delta}$, and $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$ with $\mathbf{y}^{(n)} = \mathbf{F}(\mathbf{x}^{(n)}) + \boldsymbol{\delta}_{\mathbf{y}}$. Moreover, for the covariance matrix $\mathsf{C}_{\mathbf{y}}^{\delta} = \sigma_{\mathbf{y}}^2\mathsf{I}$, where I is the identity matrix, we find:

$$E_D(\boldsymbol{\omega}) = \sum_{n=1}^{N} \frac{1}{2\sigma_{\mathbf{y}}^2}||\mathbf{y}^{(n)} - \mathbf{f}(\mathbf{z}^{(n)}, \boldsymbol{\omega})||^2, \tag{21}$$

or more precisely:

$$E_D(\boldsymbol{\omega}) = \sum_{n=1}^{N} \Big[\frac{1}{2\sigma_{\mathbf{y}}^2}||\mathbf{y}^{(n)} - \mathbf{f}(\mathbf{z}^{(n)}, \boldsymbol{\omega})||^2 + \frac{N_{\mathbf{y}}}{2}\log\sigma_{\mathbf{y}}^2\Big], \tag{22}$$

Assuming $\mathsf{C}_{\boldsymbol{\omega}} = \sigma_{\boldsymbol{\omega}}^2\mathsf{I}$ and using Equation (21), we infer that the point estimate $\widehat{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\text{MAP}}$ is the minimizer of the Tikhonov function:

$$E(\boldsymbol{\omega}) = \frac{1}{2}\sum_{n=1}^{N} ||\mathbf{y}^{(n)} - \mathbf{f}(\mathbf{x}^{(n)}, \boldsymbol{\omega})||^2 + \alpha||\boldsymbol{\omega}||^2, \tag{23}$$

where $\alpha = \sigma_{\mathbf{y}}^2/(2\sigma_{\boldsymbol{\omega}}^2)$ is the regularization parameter.

2. A model with exact data can be handled by considering the data model with output noise and by making $\sigma_{\mathbf{y}}^2 \approx 0$ in the representation of the data error covariance matrix $\mathsf{C}_{\mathbf{y}}^{\delta} = \sigma_{\mathbf{y}}^2\mathsf{I}$. For $\sigma_{\mathbf{y}}^2 \approx 0$, the relation $\mathbf{y}^{(n)} = \mathbf{F}(\mathbf{x}^{(n)}) + \boldsymbol{\delta}_{\mathbf{y}}$ yields $\mathbf{y}^{(n)} \approx \mathbf{F}(\mathbf{x}^{(n)})$, while Equation (23) and the relation $\alpha = \sigma_{\mathbf{y}}^2/(2\sigma_{\boldsymbol{\omega}}^2) \approx 0$ gives:

$$\widehat{\boldsymbol{\omega}} \approx \boldsymbol{\omega}_{\text{MLE}} = \arg\min_{\boldsymbol{\omega}} E(\boldsymbol{\omega}),$$

$$E(\boldsymbol{\omega}) = \frac{1}{2}\sum_{n=1}^{N} ||\mathbf{y}^{(n)} - \mathbf{f}(\mathbf{x}^{(n)}, \boldsymbol{\omega})]||^2.$$

Thus, when learning a neural network with the exact data, the maximum likelihood estimate minimizes the sum of square errors. Note that in this case, regularization is not absolutely required, because the output data are exact.

3. For a data model with input and output noise, the computation of the estimate $\widehat{\boldsymbol{\omega}}$ is not a trivial task, because the covariance matrix $\mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z}, \boldsymbol{\omega})$, which enters in the expression of $E_D(\boldsymbol{\omega})$, depends on $\boldsymbol{\omega}$. Moreover, in Equation (11), $\mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z}, \boldsymbol{\omega})$ corresponds to a linearization of the neural network function under the assumption that the noise

process in the input space is small. This problem can be solved by implicitly learning the covariance matrix $\mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z}, \boldsymbol{\omega})$ from the loss function [29]. Specifically, we assume that $\mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z}, \boldsymbol{\omega})$ is a diagonal matrix with entries $\sigma_j^2(\mathbf{z}, \boldsymbol{\omega})$, that is:

$$\mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z}, \boldsymbol{\omega}) = \mathrm{diag}[\sigma_j^2(\mathbf{z}, \boldsymbol{\omega})]_{j=1}^{N_\mathrm{y}}, \tag{24}$$

implying:

$$\log p(\mathbf{y}|\mathbf{z}, \boldsymbol{\omega}) \propto -\sum_{j=1}^{N_\mathrm{y}} \Big\{ \frac{1}{2\sigma_j^2(\mathbf{z}, \boldsymbol{\omega})} [y_j - \mu_j(\mathbf{z}, \boldsymbol{\omega})]^2 + \frac{1}{2} \log \sigma_j^2(\mathbf{z}, \boldsymbol{\omega}) \Big\}. \tag{25}$$

Here, we identified $\boldsymbol{\mu} \equiv \mathbf{f}$, and set $\mu_j = [\boldsymbol{\mu}]_j$ and $y_j = [\mathbf{y}]_j$. To learn the variances $\sigma_j^2(\mathbf{z}, \boldsymbol{\omega})$ from the loss function, we use a single network with input $\mathbf{z}$, and output $[\mu_j(\mathbf{z}, \boldsymbol{\omega}), \sigma_j^2(\mathbf{z}, \boldsymbol{\omega})] \in \mathbb{R}^{2N_\mathrm{y}}$; thus, in the output layer, N_y units are used to predict μ_j and N_y units to predict σ_j^2. In practice, to increase numerical stability, we train the network to predict the log variance $\rho_j = \log \sigma_j^2$, in which case, the likelihood loss function is:

$$E_\mathrm{D}(\boldsymbol{\omega}) = \sum_{n=1}^{N} E_\mathrm{D}^{(n)}(\boldsymbol{\omega}), \tag{26}$$

with:

$$E_\mathrm{D}^{(n)}(\boldsymbol{\omega}) = \frac{1}{2} \sum_{j=1}^{N_\mathrm{y}} \{ \exp[-\rho_j(\mathbf{z}^{(n)}, \boldsymbol{\omega})] [y_j^{(n)} - \mu_j(\mathbf{z}^{(n)}, \boldsymbol{\omega})]^2 + \rho_j(\mathbf{z}^{(n)}, \boldsymbol{\omega}) \}. \tag{27}$$

It should be pointed out that from Equation (25) that it is apparent that the model is stopped from predicting high uncertainty through the $\log \sigma_j^2$ term, but also low uncertainty for points with high residual error, as low σ_j^2 will increase the contribution of the residual. On the other hand, it should also be noted that a basic assumption of the model is that the covariance matrix $\mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z}, \boldsymbol{\omega})$ is diagonal, which unfortunately, is contradicted by Equation (11).

4. In order to generate a data set with input and output noise, that is, $\mathcal{D} = \{(\mathbf{z}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$, where $\mathbf{z}^{(n)} = \mathbf{x}^{(n)} + \boldsymbol{\delta}_\mathbf{x}$, $\boldsymbol{\delta}_\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathrm{C}_\mathbf{x}^{\delta})$, and $\mathbf{y}^{(n)} = \mathbf{F}(\mathbf{x}^{(n)}) + \boldsymbol{\delta}_\mathbf{y}$, we used the jitter approach. According to this approach, at each forward pass through the network, a new random noise $\boldsymbol{\delta}_\mathbf{x}$ is added to the original input vector $\mathbf{x}^{(n)}$. In other words, each time a training sample is passed through the network, random noise is added to the input variables, making them different every time it is passed through the network. By this approach, which is a simple form of data augmentation, the dimension of the data set is reduced (actually, the data set is $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$ with $\mathbf{y}^{(n)} = \mathbf{F}(\mathbf{x}^{(n)}) + \boldsymbol{\delta}_\mathbf{y}$).

2.2. Uncertainties

In our analysis, we are interested in estimating the uncertainty associated with the underlying processes. The quantity which exactly quantifies the model's uncertainty is the predictive distribution of an unknown output $\mathbf{y}$ given an observable input $\mathbf{z}$, defined by

$$p(\mathbf{y}|\mathbf{z}, \mathcal{D}) = \int p(\mathbf{y}|\mathbf{z}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathcal{D}) \mathrm{d}\boldsymbol{\omega}. \tag{28}$$

If $p(\mathbf{y}|\mathbf{z}, \mathcal{D})$ is known, the first two moments of the output $\mathbf{y}$ can be computed as

$$\mathbb{E}(\mathbf{y}) = \int \mathbf{y} p(\mathbf{y}|\mathbf{z}, \mathcal{D}) \mathrm{d}\mathbf{y}, \tag{29}$$

$$\mathbb{E}(\mathbf{y}\mathbf{y}^T) = \int \mathbf{y}\mathbf{y}^T p(\mathbf{y}|\mathbf{z}, \mathcal{D}) \mathrm{d}\mathbf{y}, \tag{30}$$

and the covariance matrix as

$$\mathrm{Cov}(\mathbf{y}) = \mathbb{E}(\mathbf{y}\mathbf{y}^T) - \mathbb{E}(\mathbf{y})\mathbb{E}(\mathbf{y})^T. \tag{31}$$

From Equation (28), we see that the predictive distribution $p(\mathbf{y}|\mathbf{z}, \mathcal{D})$ can be computed if the Bayesian posterior $p(\boldsymbol{\omega}|\mathcal{D})$ is known. Unfortunately, computing the distribution $p(\boldsymbol{\omega}|\mathcal{D})$ by means of Equation (13) is usually an intractable problem, because computing the evidence $p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\omega}) p(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}$ is not a trivial task. To address this problem, either:

1. The Laplace approximation, which yields an approximate representation for the posterior $p(\boldsymbol{\omega}|\mathcal{D})$; or
2. A variational inference approach, which learns a variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ to approximate the posterior $p(\boldsymbol{\omega}|\mathcal{D})$,

can be used. In the first case, we are dealing with deterministic (point estimate) neural networks, in which a single realization of the network parameters $\boldsymbol{\omega}$ is learned, while in the second case, we are dealing with stochastic neural networks, in which a distribution over the network parameters $\boldsymbol{\omega}$ is learned.

The Laplace approximation is of theoretical interest because it provides explicit representations for the different types of uncertainty. In Appendix A it is shown that under the Laplace approximation, the predictive distribution is given by [23,24,30]

$$p(\mathbf{y}|\mathbf{z}, \mathcal{D}) \propto \exp\Big\{-\frac{1}{2}[\mathbf{y} - \mathbf{f}(\mathbf{z}, \widehat{\boldsymbol{\omega}})]^T \mathscr{C}_{\mathbf{y}}^{-1}(\mathbf{z}, \widehat{\boldsymbol{\omega}})[\mathbf{z} - \mathbf{f}(\mathbf{z}, \widehat{\boldsymbol{\omega}})]\Big\}, \tag{32}$$

$$\mathscr{C}_{\mathbf{y}}(\mathbf{z}, \widehat{\boldsymbol{\omega}}) = \mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z}, \widehat{\boldsymbol{\omega}}) + \mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{z}, \widehat{\boldsymbol{\omega}}), \tag{33}$$

$$\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{z}, \widehat{\boldsymbol{\omega}}) = \mathsf{K}_{\omega}(\mathbf{z}, \widehat{\boldsymbol{\omega}}) \mathsf{H}^{-1}(\widehat{\boldsymbol{\omega}}) \mathsf{K}_{\omega}^T(\mathbf{z}, \widehat{\boldsymbol{\omega}}), \tag{34}$$

where:

$$\mathsf{K}_{\omega}(\mathbf{z}, \boldsymbol{\omega}) = \frac{\partial \mathbf{f}}{\partial \boldsymbol{\omega}}(\mathbf{z}, \boldsymbol{\omega}) \tag{35}$$

is the Jacobian of $\mathbf{f}$ with respect to $\boldsymbol{\omega}$ and:

$$[\mathsf{H}(\widehat{\boldsymbol{\omega}})]_{ij} = \frac{\partial E}{\partial \omega_i \partial \omega_j}(\widehat{\boldsymbol{\omega}}), \tag{36}$$

is the Hessian matrix of the loss function. Equation (32) provides a Gaussian approximation to the predictive distribution $p(\mathbf{y}|\mathbf{z}, \mathcal{D})$ with mean $\mathbf{f}(\mathbf{z}, \widehat{\boldsymbol{\omega}})$ and covariance matrix $\mathscr{C}_{\mathbf{y}}(\mathbf{z}, \widehat{\boldsymbol{\omega}})$. From Equation (33), we see that the covariance matrix $\mathscr{C}_{\mathrm{y}}(\mathbf{z}, \widehat{\boldsymbol{\omega}})$ has two components.

1. The first component $\mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z}, \widehat{\boldsymbol{\omega}})$ is the covariance matrix in the distribution over the error in the output $\mathbf{y}$. This term, which is input dependent, describes the so-called aleatoric heteroscedastic uncertainty measured by $p(\mathbf{y}|\mathbf{z}, \boldsymbol{\omega})$. For a data model with output noise, we have (cf. Equation (11)) $\mathscr{C}_{\mathbf{y}}^{\delta} = \mathsf{C}_{\mathbf{y}}^{\delta}$, and this term, which is input independent, describes the so-called aleatoric homoscedastic uncertainty measured by $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})$.
2. The second component $\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{z}, \widehat{\boldsymbol{\omega}})$ reflects the uncertainty induced in the weights $\boldsymbol{\omega}$, also called epistemic uncertainty or model uncertainty. The sources of this uncertainty measured by $p(\boldsymbol{\omega}|\mathcal{D})$ are for example: (i) non-optimal hyperparameters (the number of hidden layers, the number of units per layer, the type of activation function); (ii) non-optimal training parameters (the minimum learning rate at which the training is stopped, the learning rate decay factor, the batch size used for the mini-batch gradient

descent); and (iii) a non-optimal optimization algorithm (ADAGRAD, ADADELTA, ADAM, ADAMAX, NADAM).

or model uncertainty, which refers to the fact that we do not know the model that best explains the given data. For NNs, this is uncertainty from not knowing the *best values of weights* in all the trainable layers. This is often referred to as reducible uncertainty, because we can theoretically reduce this type of uncertainty by acquiring more data.

Some comments can be made here.

1. The heteroscedastic covariance matrix $\mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z},\boldsymbol{\omega}) = \mathrm{diag}[\sigma_j^2(\mathbf{z},\boldsymbol{\omega})]_{j=1}^{N_{\mathrm{y}}}$ can be learned from the data by minimizing the loss functions (26) and (27).
2. The computation of the epistemic covariance matrix $\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{z},\widehat{\boldsymbol{\omega}})$ from Equation (34) requires the knowledge of the Hessian matrix $\mathsf{H}(\widehat{\boldsymbol{\omega}})$. In general, this problem is practically insoluble because the matrix H is very huge, and so, is very difficult to compute. However, the problem can be evaded by using the diagonal Hessian approximation
$$[\mathsf{H}(\widehat{\boldsymbol{\omega}})]_{ij} = \delta_{ij}\frac{\partial^2 E}{\partial \omega_i^2}(\widehat{\boldsymbol{\omega}}),$$
where δ_{ij} is the Kronecker delta. The diagonal matrix elements can be very efficiently computed by using a procedure which is similar to the back-propagation algorithm used for computing the first derivatives [31].
3. The covariance matrix $\mathscr{C}_{\mathbf{y}}(\mathbf{z},\widehat{\boldsymbol{\omega}})$ can be approximated by the conditional average covariance matrix $\mathbb{E}(\mathscr{C}_{\mathbf{y}}|\mathcal{D})$ of all network errors $\boldsymbol{\varepsilon} = \mathbf{y} - \mathbf{f}(\mathbf{z},\widehat{\boldsymbol{\omega}})$ over the data set $\mathcal{D}$, meaning that:
$$\mathbb{E}(\mathscr{C}_{\mathrm{y}}|\mathcal{D}) = \frac{1}{N}\sum_{n=1}^{N}[\boldsymbol{\varepsilon}^{(n)} - \mathbb{E}(\boldsymbol{\varepsilon})][\boldsymbol{\varepsilon}^{(n)} - \mathbb{E}(\boldsymbol{\varepsilon})]^T,$$
where $\boldsymbol{\varepsilon}^{(n)} = \mathbf{y}^{(n)} - \mathbf{f}(\mathbf{z}^{(n)},\widehat{\boldsymbol{\omega}})$ and $\mathbb{E}(\boldsymbol{\varepsilon}) = (1/N)\sum_{n=1}^{N}\boldsymbol{\varepsilon}^{(n)}$. Note that this is a very rough approximation, because in contrast to $\mathscr{C}_{\mathbf{y}}(\mathbf{z},\widehat{\boldsymbol{\omega}})$, $\mathbb{E}(\mathscr{C}_{\mathbf{y}}|\mathcal{D})$ does not depend on $\mathbf{z}$.
4. For exact data, we have $\mathscr{C}_{\mathbf{y}}(\mathbf{x},\widehat{\boldsymbol{\omega}}) = B\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{x},\widehat{\boldsymbol{\omega}})$. Thus, when learning a neural network with exact data, the predictive distribution $p(\mathbf{y}|\mathbf{x},\mathcal{D})$ is Gaussian with mean $\mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})$ and (epistemic) covariance matrix $\mathcal{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{x},\widehat{\boldsymbol{\omega}})$.

2.3. Bayesian Networks

Stochastic neural networks are a type of neural networks built by introducing stochastic components into the network (activation or weights). A Bayesian neural network can be regarded as a stochastic neural network trained by using Bayesian inference [32]. This type of neural network provides a natural approach to quantify uncertainty in deep learning and allows to distinguish between epistemic and aleatoric uncertainties.

Bayesian neural networks equipped with variational inference bypass the computation of the evidence $p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\omega})p(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega}$, which determines the Bayesian posterior $p(\boldsymbol{\omega}|\mathcal{D})$ via Equation (13), by learning a variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ to approximate $p(\boldsymbol{\omega}|\mathcal{D})$, that is:

$$q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \approx p(\boldsymbol{\omega}|\mathcal{D}), \tag{37}$$

where $\boldsymbol{\theta}$ are some variational parameters. These parameters are computed by minimizing the Kullback–Leibler (KL) divergence:

$$\mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\omega})|p(\boldsymbol{\omega}|\mathcal{D})) = \int q_{\boldsymbol{\theta}}(\boldsymbol{\omega})\log\left[\frac{q_{\boldsymbol{\theta}}(\boldsymbol{\omega})}{p(\boldsymbol{\omega}|\mathcal{D})}\right]\mathrm{d}\boldsymbol{\omega}, \tag{38}$$

with respect to $\boldsymbol{\theta}$. In fact, the KL divergence is a measure of similarity between the approximate distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ and the posterior distribution obtained from the full

Gaussian process $p(\boldsymbol{\omega}|\mathcal{D})$, and minimizing the KL divergence to be the same as minimizing the variational free energy defined by

$$\begin{aligned} F(\boldsymbol{\theta}, \mathcal{D}) &= -\int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \log p(\mathcal{D}|\boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega} + \mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\omega})|p(\boldsymbol{\omega})) \\ &= \int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \Big[\log q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) - \log p(\boldsymbol{\omega}) - \log p(\mathcal{D}|\boldsymbol{\omega})\Big] \mathrm{d}\boldsymbol{\omega}. \end{aligned} \tag{39}$$

Considering the data model with output noise, assuming $\mathsf{C}_{\mathbf{y}}^{\delta} = \sigma_{\mathbf{y}}^2 \mathsf{I}$, and replacing $p(\boldsymbol{\omega}|\mathcal{D})$ by $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ in Equation (28), which gives an approximate predictive distribution:

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}) \approx q_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}) q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \tag{40}$$

which can be approximated at test time by

$$q_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \approx \frac{1}{T} \sum_{t=1}^{T} p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}_t), \tag{41}$$

where $\boldsymbol{\omega}_t$ is sampled from the distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$. As a result, for $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}) = \mathcal{N}(\mathbf{y}, \mathbf{f}(\mathbf{x}, \boldsymbol{\omega}), \mathsf{C}_{\mathbf{y}}^{\delta} = \sigma_{\mathbf{y}}^2 \mathsf{I})$, the predictive mean and the covariance matrix of the output $\mathbf{y}$ given the input $\mathbf{x}$, can be approximated, respectively, by (Appendix B):

$$\mathbb{E}(\mathbf{y}) \approx \frac{1}{T} \sum_{t=1}^{T} \mathbf{f}(\mathbf{x}, \boldsymbol{\omega}_t), \tag{42}$$

$$\mathrm{Cov}(\mathbf{y}) \approx \sigma_{\mathbf{y}}^2 \mathsf{I} + \frac{1}{T} \sum_{t=1}^{T} \mathbf{f}(\mathbf{x}, \boldsymbol{\omega}_t) \mathbf{f}(\mathbf{x}, \boldsymbol{\omega}_t)^T - \mathbb{E}(\mathbf{y}) \mathbb{E}(\mathbf{y})^T. \tag{43}$$

The first term $\sigma_{\mathbf{y}}^2 \mathsf{I}$ in Equation (43) corresponds to the homoscedastic uncertainty (the amount of noise in the data), while the second part corresponds to the epistemic uncertainty (how much the model is uncertain in its prediction). The predictive mean (42) is known as model averaging, and is equivalent to performing T stochastic passes through the network and averaging the results. Note that for exact data, i.e., $\sigma_{\mathbf{y}}^2 \approx 0$, the covariance matrix simplifies to:

$$\mathrm{Cov}(\mathbf{y}) \approx \frac{1}{T} \sum_{t=1}^{T} \mathbf{f}(\mathbf{x}, \boldsymbol{\omega}_t) \mathbf{f}(\mathbf{x}, \boldsymbol{\omega}_t)^T - \mathbb{E}(\mathbf{y}) \mathbb{E}(\mathbf{y})^T \tag{44}$$

and describes the epistemic uncertainty.

In this section, we present the most relevant algorithm that is used for Bayesian inference (Bayes-by-backprop), as well as two Bayesian approximate methods. For this purpose, we consider a data model with output noise and assume $\mathsf{C}_{\mathbf{y}}^{\delta} = \sigma_{\mathbf{y}}^2 \mathsf{I}$; in this case, $p(\mathcal{D}|\boldsymbol{\omega})$ is given by Equation (15) in conjunction with Equations (21) or (22).

2.3.1. Bayes by Backpropagation

Bayes-by-backprop is a practical implementation of variational inference combined with a reparameterization trick [33]. The idea of the parameterization trick is to introduce a random variable $\boldsymbol{\epsilon}$, and to determine $\boldsymbol{\omega}$ by a deterministic transformation $t(\boldsymbol{\epsilon}, \boldsymbol{\theta})$, such that $\boldsymbol{\omega} = t(\boldsymbol{\epsilon}, \boldsymbol{\theta})$ follows the distributions $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$. If the variational posterior $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ is a Gaussian distribution with a diagonal covariance matrix, i.e., $q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}; \boldsymbol{\mu}_{\omega}, \mathrm{diag}[\sigma_{\omega j}^2]_{j=1}^{W})$, where $\sigma_{\omega j} = [\boldsymbol{\sigma}_{\omega}]_j$ and $W = \dim(\boldsymbol{\omega}) = \dim(\boldsymbol{\mu}_{\omega}) = \dim(\boldsymbol{\sigma}_{\omega})$, a sample of the weight $\boldsymbol{\omega}$ can be obtained by sampling a unit Gaussian $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathsf{I})$, scaling it by a standard deviation $\boldsymbol{\sigma}_{\omega}$ and by shifting a mean $\boldsymbol{\mu}_{\omega}$. Actually, to guarantee that $\boldsymbol{\sigma}_{\omega}$ is always non-negative, the standard deviation can be parametrized pointwise as $\boldsymbol{\sigma}_{\omega} = \boldsymbol{\sigma}_{\omega}(\boldsymbol{\rho}_{\omega}) = \log(1 + \exp(\boldsymbol{\rho}_{\omega}))$ or as $\boldsymbol{\sigma}_{\omega} = \boldsymbol{\sigma}_{\omega}(\boldsymbol{\rho}_{\omega}) = \exp(\boldsymbol{\rho}_{\omega}/2)$ (i.e., $\rho_{\omega j} = \log \sigma_{\omega j}^2$). Thus, the variational posterior

parameters are $\boldsymbol{\theta} = (\boldsymbol{\mu}_\omega, \boldsymbol{\rho}_\omega)$. By using a Monte Carlo sampling with one sample, we compute the variational free energy (39) by using the relations:

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \mathcal{N}(\mathbf{0}, \mathsf{I}), \\ \boldsymbol{\omega} &= \boldsymbol{\mu}_\omega + \boldsymbol{\sigma}_\omega(\boldsymbol{\rho}_\omega) \circ \boldsymbol{\epsilon}, \\ F(\boldsymbol{\theta}, \mathcal{D}) &= \log q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) - \log p(\boldsymbol{\omega}) - \log p(\mathcal{D}|\boldsymbol{\omega}), \end{aligned}$$

where $\circ$ denotes point-wise multiplication. In Ref. [33], the prior over the weights $p(\boldsymbol{\omega})$ is chosen as a mixture of two Gaussian with zero mean but differing variances, meaning that:

$$p(\boldsymbol{\omega}) = \beta\mathcal{N}(\boldsymbol{\omega}; 0, \sigma_1^2\mathsf{I}) + (1-\beta)\mathcal{N}(\boldsymbol{\omega}; \mathbf{0}, \sigma_2^2\mathsf{I}), \tag{45}$$

while in [34], it is shown that, for $q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}; \boldsymbol{\mu}_\omega, \mathrm{diag}[\sigma_{\omega j}^2])$ and $p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}; \mathbf{0}, \mathsf{I})$, the KL divergence $\mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\omega})|p(\boldsymbol{\omega}))$ can be analytically computed; the result is:

$$\begin{aligned} \mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\omega})|p(\boldsymbol{\omega})) &= \int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \Big[\log q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) - \log p(\boldsymbol{\omega})\Big] \mathrm{d}\boldsymbol{\omega} \\ &= \frac{1}{2}\sum_{j=1}^{W} [\mu_{\omega j}^2 + \sigma_{\omega j}^2 - \log(\sigma_{\omega j}^2) - 1], \end{aligned} \tag{46}$$

where $\mu_{\omega j} = [\boldsymbol{\mu}_\omega]_j$, $\sigma_{\omega j} = [\boldsymbol{\sigma}_\omega]_j$. Thus:

$$F(\boldsymbol{\theta}, \mathcal{D}) = -\log p(\mathcal{D}|\boldsymbol{\omega}) + \frac{1}{2}\sum_{j=1}^{W} [\mu_{\omega j}^2 + \sigma_{\omega j}^2 - \log(\sigma_{\omega j}^2) - 1]. \tag{47}$$

After the training stage, i.e., after the variational posterior parameters $\boldsymbol{\theta} = (\boldsymbol{\mu}_\omega, \boldsymbol{\rho}_\omega)$ have been learned, we consider the set of samples $\{\boldsymbol{\omega}_t\}_{t=1}^T$, where $\boldsymbol{\omega}_t = \boldsymbol{\mu}_\omega + \boldsymbol{\sigma}_\omega(\boldsymbol{\rho}_\omega) \circ \boldsymbol{\epsilon}_t$ and $\boldsymbol{\epsilon}_t$ is sampled from the Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathsf{I})$, and compute the predictive mean and covariance matrix according to Equations (42) and (43).

2.3.2. Dropout

Dropout, which was initially proposed as a regularization technique during the training [35,36], is equivalent to a Bayesian approximation. This result was proved in [37] by showing that the variational free energy $F(\boldsymbol{\theta}, \mathcal{D})$ has the standard form representation of the dropout loss function (as the sum of a square loss function and a L_2 regularization term). A simplified proof of this assertion is given in Appendix C. The term "dropout" refers to removing a unit along with all its connections. The choice of a dropped unit is random. In the case of dropout, the feed-forward operation of a standard neural network (2) and (3) becomes:

$$\mathbf{u}_l = \mathsf{W}_l \mathsf{Z}_{l-1} \mathbf{y}_{l-1} + \mathbf{b}_l, \tag{48}$$

$$\mathbf{y}_l = g_l(\mathbf{u}_l), \tag{49}$$

where:

$$\mathsf{Z}_{l-1} = \mathrm{diag}[z_{k,l-1}]_{k=1}^{N_{l-1}}, \tag{50}$$

$$z_{k,l-1} \sim \mathrm{Bernoulli}(p). \tag{51}$$

Essentially, the output of the unit k in layer $l-1$, $y_{k,l-1}$ is multiplied by the binary variable $z_{k,l-1}$ to create the new output $z_{k,l-1}y_{k,l-1}$. The binary variable $z_{k,l-1}$ takes the value 1 with probability p and the value 0 with probability $1-p$; thus, if $z_{k,l-1} = 0$, the new output is zero and the unit k is dropped out as an input to the next layer l. The same values of the binary variable are used in the backward pass when propagating the derivatives of the loss function. In the test time, the weights W_l are scaled as $p\mathsf{W}_l$. Thus, we retain units

with probability p at training time, and multiply the weights by p at test time. Alternatively, in order to maintain constant outputs at training time, we can scale the weights by $1/p$, and do not modify the weights at test time. The model parameters $\boldsymbol{\omega} = \{\mathsf{W}_l, \mathbf{b}_l\}_{l=1}^{L}$ are obtained by minimizing the loss function (21) with possibly an L_2 regularization term.

Uncertainty can be obtained from a dropout neural network [37]. For the set of samples $\{\boldsymbol{\omega}_t\}_{t=1}^{T}$, where $\boldsymbol{\omega}_t$ corresponds to a realization from the Bernoulli distribution $\mathsf{Z}_{l-1}^{(t)}$ and $\mathsf{W}_l^{(t)} = \mathsf{W}_l \mathsf{Z}_{l-1}^{(t)}$ for all $l = 1, \ldots, L$, the predictive mean and covariance matrix are computed by means of Equations (42) and (43).

2.3.3. Batch Normalization

Ioffe and Szegedy [38] introduced batch normalization as a technique for training deep neural networks that normalizes (standardizes) the inputs to a layer for each mini-batch. This allows for higher learning rates, regularizes the model, and reduces the number of training epochs. Moreover, Teye et al. [39] showed that a batch normalized network can be regarded as an approximate Bayesian model, and it can thus be used for modeling uncertainty.

Let us split the data set $\mathcal{D}$ into N_b mini-batches, and let the nth mini-batch $\mathcal{B}^{(n)}$ contain M pair samples $(\mathbf{x}^{(n,m)}, \mathbf{y}^{(n,m)})$, meaning that:

$$\begin{aligned} \mathcal{D} &= \cup_{n=1}^{N_\mathrm{b}} \mathcal{B}^{(n)}, \\ \mathcal{B}^{(n)} &= \{(\mathbf{x}^{(n,m)}, \mathbf{y}^{(n,m)})\}_{m=1}^{M}. \end{aligned}$$

In batch normalization, the input $u_{j,l}^{(n,m)} = \sum_{i=1}^{N_{l-1}} w_{ji,l} y_{i,l-1}^{(n,m)}$ of the activation function g_l corresponding to unit j, layer l, sample m, and mini-batch n, which is first normalized:

$$\widetilde{u}_{j,l}^{(n,m)} = \frac{u_{j,l}^{(n,m)} - \mu_{j,l}^{(n)}}{\sqrt{v_{j,l}^{(n)} + \varepsilon}}, \tag{52}$$

and then scaled and shifted:

$$\widehat{u}_{j,l}^{(n,m)} = \alpha_{j,l} \widetilde{u}_{j,l}^{(n,m)} + \beta_{j,l}, \tag{53}$$

where:

$$\mu_{j,l}^{(n)} = \frac{1}{M} \sum_{m=1}^{M} u_{j,l}^{(n,m)} \text{ and } v_{j,l}^{(n)} = \frac{1}{M} \sum_{m=1}^{M} (u_{j,l}^{(n,m)} - \mu_{j,l}^{(n)})^2 \tag{54}$$

are the mean and variance of the activation inputs over the M samples (in unit j, layer l and mini-batch n), respectively, $\alpha_{j,l}$ and $\beta_{j,l}$ are learnable model parameters, and ε is a small number added to the mini-batch variance to prevent division by zero. By normalization (or whitening), $\widetilde{u}_{j,l}^{(n,m)}$ becomes a random variable with zero mean and unit variance, while by scaling and shifting, we guarantee that the transformation (53) can represent the identity transform. In a stochastic framework, we interpret the mean $\mu_{j,l}^{(n)}$ and variance $v_{j,l}^{(n)}$, corresponding to the nth mini-batch $\mathcal{B}^{(n)}$, as realizations of the random variables $\mu_{j,l}$ and $v_{j,l}$, corresponding to the data set $\mathcal{D}$. The model parameters include the learnable parameters:

$$\boldsymbol{\theta} = \{w_{ji,l}, \alpha_{j,l}, \beta_{j,l} | j = 1, \ldots, N_l,\ i = 1, \ldots, N_{l-1},\ l = 1, \ldots, L\},$$

and the stochastic parameters:

$$\boldsymbol{\omega} = (\boldsymbol{\mu}, \boldsymbol{v}) = \{(\mu_{j,l}, v_{j,l}) | j = 1, \ldots, N_l\ l = 1, \ldots, L\},$$

where for the mini-batch $\mathcal{B}^{(n)}$:

$$\boldsymbol{\omega}^{(n)} = (\boldsymbol{\mu}^{(n)}, \boldsymbol{v}^{(n)}) = \{(\mu_{j,l}^{(n)}, v_{j,l}^{(n)}) | j = 1, \ldots, N_l \, l = 1, \ldots, L\}$$

is a realization of $\boldsymbol{\omega} = (\boldsymbol{\mu}, \boldsymbol{v})$. Optimizing over mini-batches of size M instead on the full training set, the objective function for the mini-batch $\mathcal{B}^{(n)}$ becomes [39]:

$$F^{(n)}(\boldsymbol{\theta}) = \frac{1}{2M} \sum_{m=1}^{M} ||\mathbf{y}^{(n,m)} - \mathbf{f}_{\boldsymbol{\omega}^{(n)}}(\mathbf{x}^{(n,m)}, \boldsymbol{\theta})||^2 + \Omega(\boldsymbol{\theta}), \tag{55}$$

where $\Omega(\boldsymbol{\theta}) = \alpha \sum_{l=1}^{L} \|\mathrm{W}_l\|_2^2$ with $[\mathrm{W}_l]_{ji} = w_{ji,l}$ is the regularization term and the notation $\mathbf{f}_{\boldsymbol{\omega}^{(n)}}(\mathbf{x}, \boldsymbol{\theta})$ indicates that the mean and variance $\boldsymbol{\omega}^{(n)} = (\boldsymbol{\mu}^{(n)}, \boldsymbol{v}^{(n)})$ are used in the normalization step (52). At the end of the training stage, we obtain:

1. The maximum a posteriori parameters $\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}_{\mathrm{MAP}}$;
2. The mean and variance realizations $\{\boldsymbol{\omega}^{(n)} = (\boldsymbol{\mu}^{(n)}, \boldsymbol{v}^{(n)})\}_{n=1}^{N_\mathrm{b}}$ of the stochastic parameters $\boldsymbol{\omega} = (\boldsymbol{\mu}, \boldsymbol{v})$; and
3. The moving averages of the mean and variance over the training set $\overline{\boldsymbol{\omega}} = (\overline{\boldsymbol{\mu}} = \mathbb{E}(\boldsymbol{\mu}), \overline{\boldsymbol{v}} = \mathbb{E}(\boldsymbol{v}))$.

Some comments can be made here.

1. A batch-normalized network samples the stochastic parameters $\boldsymbol{\omega}$ once per training step (mini-batch). For a large number of epochs, the $\boldsymbol{\omega}^{(n)}$ become independent and identical distributed random variables for each training example;
2. The variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ corresponds to the joint distribution of the weights induced by the stochastic parameters $\boldsymbol{\omega}$;
3. The equivalence between a batch-normalized network and a Bayesian approximation was proven in [39] by showing that (cf. Equations (39) and (55)) $\partial \mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) | p(\boldsymbol{\omega}))/\partial \boldsymbol{\theta} = (N/\sigma_{\mathbf{y}}^2)\, \partial \Omega(\boldsymbol{\theta})/\partial \boldsymbol{\theta}$. The proof relies on the following simplified assumptions: (i) no scale and shift transformations; (ii) batch normalization applied to each layer; (iii) independent input features in each layer; and (iv) large N and M.

In the inference, the output for a given input $\mathbf{x}$ is $\mathbf{f}_{\overline{\boldsymbol{\omega}}}(\mathbf{x}, \widehat{\boldsymbol{\theta}})$. To estimate the predictive mean and covariance matrix, we proceed as follows. For each $t = 1 \ldots, T$, we sample a mini-batch $\mathcal{B}^{(t)}$ from the data set $\mathcal{D} = \cup_{n=1}^{N_\mathrm{b}} \mathcal{B}^{(n)}$, and for the corresponding mean and variance realization $\boldsymbol{\omega}^{(t)} \in \{\boldsymbol{\omega}^{(n)}\}_{n=1}^{N_\mathrm{b}}$, compute $\mathbf{f}_{\boldsymbol{\omega}^{(t)}}(\mathbf{x}, \widehat{\boldsymbol{\theta}})$ and then the predictive mean and covariance matrix from Equations (42) and (43) with $\mathbf{f}_{\boldsymbol{\omega}^{(t)}}(\mathbf{x}, \widehat{\boldsymbol{\theta}})$ in place of $\mathbf{f}(\mathbf{x}, \boldsymbol{\omega}_t)$.

3. Neural Networks for Atmospheric Remote Sensing

In this section, we design several neural networks for atmospheric remote sensing. To test the neural networks, we considered a specific problem, namely the retrieval of cloud optical thickness and cloud top height from radiances measured by the Earth Polychromatic Imaging Camera (EPIC) onboard the Deep Space Climate Observatory (DSCOVR). DSCOVR is placed in a Lissajous orbit about the Lagrange-1 point, and provides a unique angular perspective in an almost backward direction with scattering angles between 168° and 176°. The EPIC instrument has 10 spectral channels ranging from the UV to the near-IR, which include four channels around the oxygen A- and B-bands; two absorption channels are centered at 688 nm and 764 nm with bandwiths of 0.8 nm and 1.0 nm, respectively, while two continuum bands are centered at 680 nm and 780 nm with bandwiths of 2.0 nm. These four channels are used for inferring the cloud parameters. To generate the database, we use a radiative transfer model based on the discrete ordinate method with matrix exponential [40,41] that uses several acceleration techniques, as for example, the telescoping technique, the method of false discrete ordinate [42], the correlated k-distribution method [43], and the principal component analysis [44–46]. The atmospheric parameters to be retrieved are the cloud optical thickness τ and the cloud top height H, while the forward model

parameters are the solar and viewing zenith angles and the surface albedo. Specifically, we consider the fact that the cloud optical thickness varies in the range $4.0, \ldots, 16.0$, the cloud top height in the range $2.0, \ldots 10.0$ km, the solar and viewing zenith angles in the range $0^{\circ}, \ldots, 60^{\circ}$, and the surface albedo in the range $0.02, \ldots, 0.2$ (only snow/ice free scenes are considered). The simulations are performed for a water-cloud model with a Gamma size distribution $p(a) \propto a^{\alpha} \exp(-\alpha a / a_{\text{mod}})$ with parameters $a_{\text{mod}} = 8$ μm and $\alpha = 6$. The droplet size ranges between 0.02 and 50.0 μm, the cloud geometrical thickness is 1 km and the relative azimuth angle between the solar and viewing directions is 176°. The O_2 absorption cross-sections are computed using LBL calculations [47] with optimized rational approximations for the Voigt line profile [48]. The wavenumber grid point spacing is chosen as a fraction (e.g., 1/4) of the minimum half-width of the Voigt lines taken from HITRAN database [49]. The Rayleigh cross-section and depolarization ratios are computed as in [50], while the pressure and temperature profiles correspond to the US standard model atmosphere [51]. The radiances are solar-flux normalized and are computed by means of the delta-M approximation in conjunction with the TMS correction. We generate $N = 20{,}000$ samples by employing the smart sampling technique [52]. Among this data set, 18,000 samples were used for training and the other 2000 for prediction. The noisy spectra are generated by using the measurement noise $\boldsymbol{\delta}_{\text{mes}} \sim \mathcal{N}(\mathbf{0}, \text{diag}[\sigma^2_{\text{mes}j}]_{j=1}^4)$, where for the jth channel, we use $\sigma_{\text{mes}j} = 0.1\overline{I}_j$ with $\overline{I}_j$ being the average of the simulated radiance over the N samples.

The neural network algorithms are implemented in FORTRAN by using a feed-forward multilayer perceptron architecture. The tool contains a variety of optimization algorithms, activation functions, regularization terms, dynamic learning rates, and stopping rules. For the present application, the number of hidden layers and the number of units in each layer are optimized by using 2000 samples from the training set for validation. To estimate the performances of different hyperparameter configurations, we used the holdout cross-validation together with a grid search over a set of three values for the number of hidden layers, i.e., $\{1, 2, 3\}$, and a set of eight values for the number of units, i.e., {25, 50, 75, 100, 125, 150, 175, 200}. The mini-batch gradient descent in conjunction with Adaptive Moment Estimation (ADAM) [53] is used as optimization tool, a mini batch of 100 samples is chosen, and a ReLU activation function is considered.

3.1. Neural Networks for Solving the Direct Problem

For a direct problem, the input $\mathbf{x}$ is the set of atmospheric parameters, the output $\mathbf{y}$ is the set of data, and the forward model $\mathbf{F}$ reproduces the radiative transfer model $\mathbf{R}$.

We consider a neural network trained with exact data. For the predictive distribution $p(\mathbf{y}|\mathbf{x}, \mathcal{D})$ given by Equation (32), we assume that the epistemic covariance matrix $\mathscr{C}^{\text{e}}_{\mathbf{y}}(\mathbf{x}, \widehat{\boldsymbol{\omega}})$ $(= \mathscr{C}_{\mathbf{y}}(\mathbf{x}, \widehat{\boldsymbol{\omega}}))$ is computed from the statistics of $\boldsymbol{\varepsilon} = \mathbf{y} - \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}})$, that is, $\mathscr{C}^{\text{e}}_{\mathbf{y}} \approx \mathbb{E}(\mathscr{C}_{\mathbf{y}}|\mathcal{D})$, where $\mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}})$ is the network output. Furthermore, let $\mathbf{y}^{\delta} = \mathbf{y} + \boldsymbol{\delta}_{\mathbf{y}}$ with $\boldsymbol{\delta}_{\mathbf{y}} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}^{\delta}_{\mathbf{y}})$, be the noisy data vector. Using the result:

$$p(\mathbf{y}^{\delta}|\mathbf{y}) \propto \exp\Big[-\frac{1}{2}(\mathbf{y}^{\delta} - \mathbf{y})^T (\mathsf{C}^{\delta}_{\mathbf{y}})^{-1} (\mathbf{y}^{\delta} - \mathbf{y})\Big], \tag{56}$$

we compute the predictive distribution for the noisy data $p(\mathbf{y}^{\delta}|\mathbf{x}, \mathcal{D})$ by marginalization, that is:

$$\begin{aligned} p(\mathbf{y}^{\delta}|\mathbf{x}, \mathcal{D}) &= \int p(\mathbf{y}^{\delta}|\mathbf{y}) p(\mathbf{y}|\mathbf{x}, \mathcal{D}) \mathrm{d}\mathbf{y} \\ &= \int \exp\Big\{-\frac{1}{2}[\mathbf{y}^{\delta} - \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}}) + \Delta\mathbf{y}]^T [\mathscr{C}^{\text{e}}_{\mathbf{y}}(\widehat{\boldsymbol{\omega}})]^{-1} [\mathbf{y}^{\delta} - \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}}) + \Delta\mathbf{y}]\Big\} \\ &\times \exp\Big[-\frac{1}{2}\Delta\mathbf{y}^T (\mathsf{C}^{\delta}_{\mathbf{y}})^{-1} \Delta\mathbf{y}\Big] \mathrm{d}\Delta\mathbf{y} \\ &\propto \exp\Big\{-\frac{1}{2}[\mathbf{y}^{\delta} - \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}})]^T \mathscr{C}^{-1}_{\mathbf{y}}(\widehat{\boldsymbol{\omega}}) [\mathbf{y}^{\delta} - \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}})]\Big\}, \end{aligned} \tag{57}$$

where (cf. Equation (33)) $\mathscr{C}_{\mathbf{y}}(\widehat{\boldsymbol{\omega}}) = \mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\widehat{\boldsymbol{\omega}}) + \mathsf{C}_{\mathbf{y}}^{\delta}$ and $\Delta\mathbf{y} = \mathbf{y} - \mathbf{y}^{\delta}$.

In the next step, we solve the inverse problem $\mathbf{y}^{\delta} = \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}})$ by means of a Bayesian approach [54,55]. In this case, the posteriori density $p(\mathbf{x}|\mathbf{y}^{\delta}, \mathcal{D})$ is given by

$$p(\mathbf{x}|\mathbf{y}^{\delta}, \mathcal{D}) = \frac{p(\mathbf{y}^{\delta}|\mathbf{x}, \mathcal{D})p(\mathbf{x})}{p(\mathbf{y})}; \tag{58}$$

whence, by assuming that the state vector $\mathbf{x}$ is a Gaussian random vector with mean $\mathbf{x}_{\mathrm{a}}$ and the covariance matrix $\mathsf{C}_{\mathbf{x}}$, meaning that:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{x}_{\mathrm{a}}, \mathsf{C}_{\mathbf{x}}) \propto \exp\Big[-\frac{1}{2}(\mathbf{x} - \mathbf{x}_{\mathrm{a}})\mathsf{C}_{\mathbf{x}}^{-1}(\mathbf{x} - \mathbf{x}_{\mathrm{a}})^{T}\Big], \tag{59}$$

we obtain:

$$\log p(\mathbf{x}|\mathbf{y}^{\delta}, \mathcal{D}) = -\frac{1}{2}V(\mathbf{x}|\mathbf{y}^{\delta}) + C, \tag{60}$$

where:

$$V(\mathbf{x}|\mathbf{y}^{\delta}) = [\mathbf{y}^{\delta} - \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}})]\mathscr{C}_{\mathbf{y}}^{-1}(\widehat{\boldsymbol{\omega}})[\mathbf{y}^{\delta} - \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}})]^{T} + (\mathbf{x} - \mathbf{x}_{\mathrm{a}})\mathsf{C}_{\mathbf{x}}^{-1}(\mathbf{x} - \mathbf{x}_{\mathrm{a}})^{T} \tag{61}$$

is the a posteriori potential and C is constant. The maximum a posteriori estimator, defined by

$$\widehat{\mathbf{x}} = \mathbf{x}_{\mathrm{MAP}} = \arg\max_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y}^{\delta}, \mathcal{D}) = \arg\min_{\mathbf{x}} V(\mathbf{x}|\mathbf{y}^{\delta}), \tag{62}$$

can be computed by any optimization method, as for example, the Gauss–Newton method. If the problem is almost linear, the a posteriori density $p(\mathbf{x}|\mathbf{y}^{\delta}, \mathcal{D})$ is Gaussian with the mean $\widehat{\mathbf{x}}$ and covariance matrix [54]:

$$\mathscr{C}_{\mathbf{x}}(\widehat{\mathbf{x}}, \widehat{\boldsymbol{\omega}}) = [\mathsf{K}_{\mathbf{x}}^{T}(\widehat{\mathbf{x}})\mathscr{C}_{\mathbf{y}}^{-1}(\widehat{\boldsymbol{\omega}})\mathsf{K}_{\mathbf{x}}(\widehat{\mathbf{x}}) + \mathsf{C}_{\mathbf{x}}^{-1}]^{-1}. \tag{63}$$

It should be pointed out that we can train the neural network for a data set with output noise $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{\delta(n)})\}_{n=1}^{N}$, in which case, the covariance matrix $\mathscr{C}_{\mathbf{y}}(\widehat{\boldsymbol{\omega}})$ can be directly computed from the statistics of $\boldsymbol{\varepsilon} = \mathbf{y}^{\delta} - \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}})$.

Essentially, the method involves the following steps:

1. Train a neural network with exact data for simulating the radiative transfer model;
2. Compute the epistemic covariance matrix from the statistics of all network errors over the data set;
3. Solve the inverse problem by a Bayesian approach under the assumption that the a priori knowledge and the measurement uncertainty are both Gaussian;
4. Determine the uncertainty in the retrieval by assuming that the forward model is nearly linear.

In Figure 1, we plot the histograms of the relative error over the prediction set:

$$\varepsilon_x = \frac{x_{\mathrm{pred}} - x}{x},$$

where x stands for τ and H, and x_{pred} and x are the predicted and true values, respectively. Also shown here are the plots of x_{pred} and $x_{\mathrm{pred}} \pm 3\sigma_x$ versus x. The results demonstrate that the cloud optical thickness is retrieved with better accuracy than the cloud top height, and that for the cloud top height, the predicted uncertainty are especially unrealistic, because the condition:

$$x_{\mathrm{pred}} - 3\sigma_x \leq x \leq x_{\mathrm{pred}} + 3\sigma_x$$

is not satisfied. The reason for this failure might be that the forward model is not nearly linear.

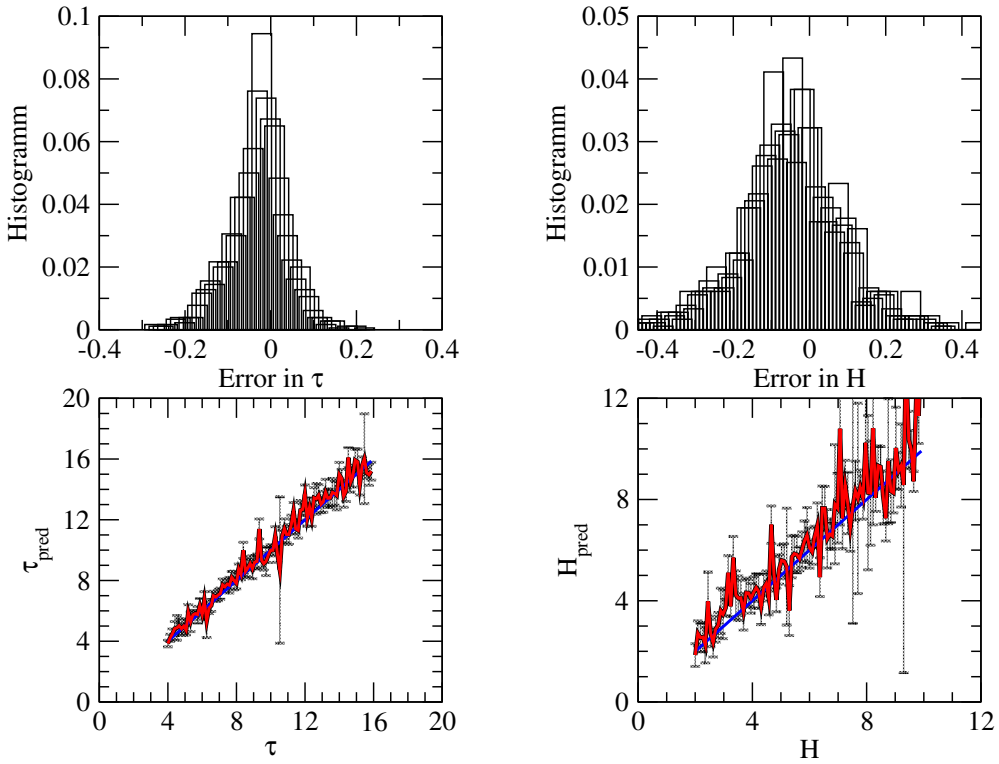

Figure 1. Retrieval results for the direct problem. The plots in the upper panels show the histograms of the relative error over the prediction set, while the lower plots show the predicted values (red) and the uncertainty intervals (gray) versus the true values.

3.2. Neural Networks for Solving the Inverse Problem

For an inverse problem, the input $\mathbf{x}$ includes the sets of data and forward model parameters (solar and viewing zenith angles, and surface albedo), while the output $\mathbf{y}$ includes the set of atmospheric parameters to be retrieved (cloud optical thickness and cloud top height).

3.2.1. Method 1

Let $\mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}})$ be the output of a neural network trained with exact data, and assume that the predictive distribution $p(\mathbf{y}|\mathbf{x}, \mathcal{D})$ given by Equation (32), and in particular, the epistemic covariance matrix $\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{x}, \widehat{\boldsymbol{\omega}})(= \mathscr{C}_{\mathbf{y}}(\mathbf{x}, \widehat{\boldsymbol{\omega}}))$ are available. For the noisy input $\mathbf{z} = \mathbf{x} + \boldsymbol{\delta}_{\mathbf{x}}$ with $p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mathbf{x}, \mathsf{C}_{\mathbf{x}}^{\delta} = \sigma_{\mathbf{x}}^2 \mathsf{I})$ and under the assumption that the prior $p(\mathbf{x})$ is a slowly varying function as compared to $p(\mathbf{z}|\mathbf{x})$, the predictive distribution of the network output can be approximated by [56]

$$\begin{aligned} p(\mathbf{y}|\mathbf{z}, \mathcal{D}) &= \int p(\mathbf{y}|\mathbf{x}, \mathcal{D}) p(\mathbf{x}|\mathbf{z}) \mathrm{d}\mathbf{x} \\ &= \frac{1}{p(\mathbf{z})} \int p(\mathbf{y}|\mathbf{x}, \mathcal{D}) p(\mathbf{z}|\mathbf{x}) p(\mathbf{x}) \mathrm{d}\mathbf{x} \\ &\approx \int p(\mathbf{y}|\mathbf{x}, \mathcal{D}) p(\mathbf{z}|\mathbf{x}) \mathrm{d}\mathbf{x}. \end{aligned} \tag{64}$$

Thus, if $p(\mathbf{x})$ varies much more slowly than $p(\mathbf{z}|\mathbf{x}) = p(\mathbf{z} - \mathbf{x})$, we assume that $p(\mathbf{y}|\mathbf{z}, \mathcal{D})$ is the convolution of the predictive distribution for an uncorrupted input $p(\mathbf{y}|\mathbf{x}, \mathcal{D})$ with the input noise distribution $p(\mathbf{z} - \mathbf{x})$. In the noise-free case, that is, if $\sigma_{\mathbf{x}} \to 0$, we have $\lim_{\sigma_{\mathbf{x}} \to 0} p(\mathbf{z}|\mathbf{x}) = \delta(\mathbf{z} - \mathbf{x})$, yielding $p(\mathbf{y}|\mathbf{z}, \mathcal{D}) = p(\mathbf{y}|\mathbf{x}, \mathcal{D})$. Using Equation (64), we obtain:

$$\begin{aligned}\mathbb{E}(\mathbf{y}|\mathbf{z}) &= \int \mathbf{y}p(\mathbf{y}|\mathbf{z},\mathcal{D})\mathrm{d}\mathbf{y}\\ &= \int\left(\int \mathbf{y}p(\mathbf{y}|\mathbf{x},\mathcal{D})\mathrm{d}\mathbf{y}\right)p(\mathbf{z}|\mathbf{x})\mathrm{d}\mathbf{x}\\ &= \int \mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})p(\mathbf{z}|\mathbf{x})\mathrm{d}\mathbf{x}\end{aligned} \tag{65}$$

and:

$$\begin{aligned}\mathbb{E}(\mathbf{y}\mathbf{y}^T|\mathbf{z}) &= \int \mathbf{y}\mathbf{y}^T p(\mathbf{y}|\mathbf{z},\mathcal{D})\mathrm{d}\mathbf{y}\\ &= \int\left(\int \mathbf{y}\mathbf{y}^T p(\mathbf{y}|\mathbf{x},\mathcal{D})\mathrm{d}\mathbf{y}\right)p(\mathbf{z}|\mathbf{x})\mathrm{d}\mathbf{x}\\ &= \int[\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{x},\widehat{\boldsymbol{\omega}}) + \mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})\mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})^T]p(\mathbf{z}|\mathbf{x})\mathrm{d}\mathbf{x}.\end{aligned} \tag{66}$$

Equations (65) and (66) show that in general $\mathbb{E}(\mathbf{y}|\mathbf{z}) \neq \mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})$, and a noise process blurs or smooths the original mappings.

To compute the predictive mean $\mathbb{E}(\mathbf{y}|\mathbf{z})$ and the covariance matrix $\mathrm{Cov}(\mathbf{y}|\mathbf{z})$, the integrals in Equations (65) and (66) must be calculated. This can be done by Monte Carlo integration (by sampling the Gaussian distribution $p(\mathbf{z}|\mathbf{x})$) or by a quadrature method. In the latter case, we consider a uniform grid $\{\mathbf{x}_i\}_{i=1}^{N_{\mathrm{int}}}$ in the interval, say $[\mathbf{z}-2\sigma_{\mathbf{x}}, \mathbf{z}+2\sigma_{\mathbf{x}}]$, define the weights:

$$v_i = \frac{\exp\left(-\frac{1}{2\sigma_{\mathbf{x}}^2}||\mathbf{z}-\mathbf{x}_i||^2\right)}{\sum_{i=1}^{N_{\mathrm{int}}}\exp\left(-\frac{1}{2\sigma_{\mathbf{x}}^2}||\mathbf{z}-\mathbf{x}_i||^2\right)}, \tag{67}$$

and use the computational formulas:

$$\mathbb{E}(\mathbf{y}|\mathbf{z}) = \sum_{i=1}^{N_{\mathrm{int}}} v_i\mathbf{f}(\mathbf{x}_i,\widehat{\boldsymbol{\omega}}), \tag{68}$$

and:

$$\mathrm{Cov}(\mathbf{y}|\mathbf{z}) = \sum_{i=1}^{N_{\mathrm{int}}} v_i\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{x}_i,\widehat{\boldsymbol{\omega}}) + \sum_{i=1}^{N_{\mathrm{int}}} v_i\mathbf{f}(\mathbf{x}_i,\widehat{\boldsymbol{\omega}})\mathbf{f}(\mathbf{x}_i,\widehat{\boldsymbol{\omega}})^T - \mathbb{E}(\mathbf{y}|\mathbf{z})\mathbb{E}(\mathbf{y}|\mathbf{z})^T. \tag{69}$$

The neural network trained with exact data can be deterministic or stochastic, as for example, Bayes-by-backprop, dropout, and batch normalization. In this regard, for each noisy input $\mathbf{z}$, we consider a uniform grid $\{\mathbf{x}_i\}_{i=1}^{N_{\mathrm{int}}}$ around $\mathbf{z}$, calculate the weights v_i by means of Equation (67), and for each $\mathbf{x}_i$, compute $\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{x}_i,\widehat{\boldsymbol{\omega}})$ as follows:

1. For a deterministic network, we approximate $\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{x}_i,\widehat{\boldsymbol{\omega}})$ by the conditional average covariance matrix $\mathbb{E}(\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}|\mathcal{D})$ of all network errors over, that is, $\mathscr{C}_{\mathbf{y}}^{\mathrm{e}} \approx \mathbb{E}(\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}|\mathcal{D})$, yielding $\sum_{i=1}^{N_{\mathrm{int}}} v_i\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{x}_i,\widehat{\boldsymbol{\omega}}) = \mathbb{E}(\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}|\mathcal{D})$;
2. For a Bayes-by-backprop and dropout networks, we compute $\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{x}_i,\widehat{\boldsymbol{\omega}})$ by means of Equation (44) with $\mathbb{E}(\mathbf{y})$ as in Equation (42);
3. For a batch normalized network, we compute $\mathscr{C}_{\mathbf{y}}^{\mathrm{e}}(\mathbf{x}_i,\widehat{\boldsymbol{\omega}})$ as

$$\mathrm{Cov}(\mathbf{y}) = \frac{1}{T}\sum_{t=1}^{T}\mathbf{f}_{\boldsymbol{\omega}^{(t)}}(\mathbf{x}_i,\widehat{\boldsymbol{\theta}})\mathbf{f}_{\boldsymbol{\omega}^{(t)}}(\mathbf{x}_i,\widehat{\boldsymbol{\theta}})^T - \mathbb{E}(\mathbf{y})\mathbb{E}(\mathbf{y})^T$$

with $\mathbb{E}(\mathbf{y}) = (1/T)\sum_{t=1}^{T}\mathbf{f}_{\boldsymbol{\omega}^{(t)}}(\mathbf{x}_i,\widehat{\boldsymbol{\theta}})$.

Note that for a Bayes-by-backprop network, the output is $\mathbf{f}(\mathbf{x}_i, \widehat{\omega} = \widehat{\mu}_\omega)$, for a dropout network, the output $\mathbf{f}(\mathbf{x}_i, \widehat{\omega})$ is computed without dropout, and for a batch normalized network, the output is $\mathbf{f}_{\overline{\omega}}(\mathbf{x}_i, \widehat{\theta})$.

In summary, the method uses:

1. Deterministic and stochastic networks trained with exact data to compute the epistemic covariance matrix; and
2. The assumption that the predictive distribution of the network output is the convolution of the predictive distribution for an uncorrupted input with the input noise distribution to estimate the covariance matrix.

Under the assumption that the noise process is Gaussian, the convolution integrals are computed by a quadrature method with a uniform grid around the noisy input.

The results for Method 1 with deterministic and stochastic networks are illustrated in Figures 2–5.

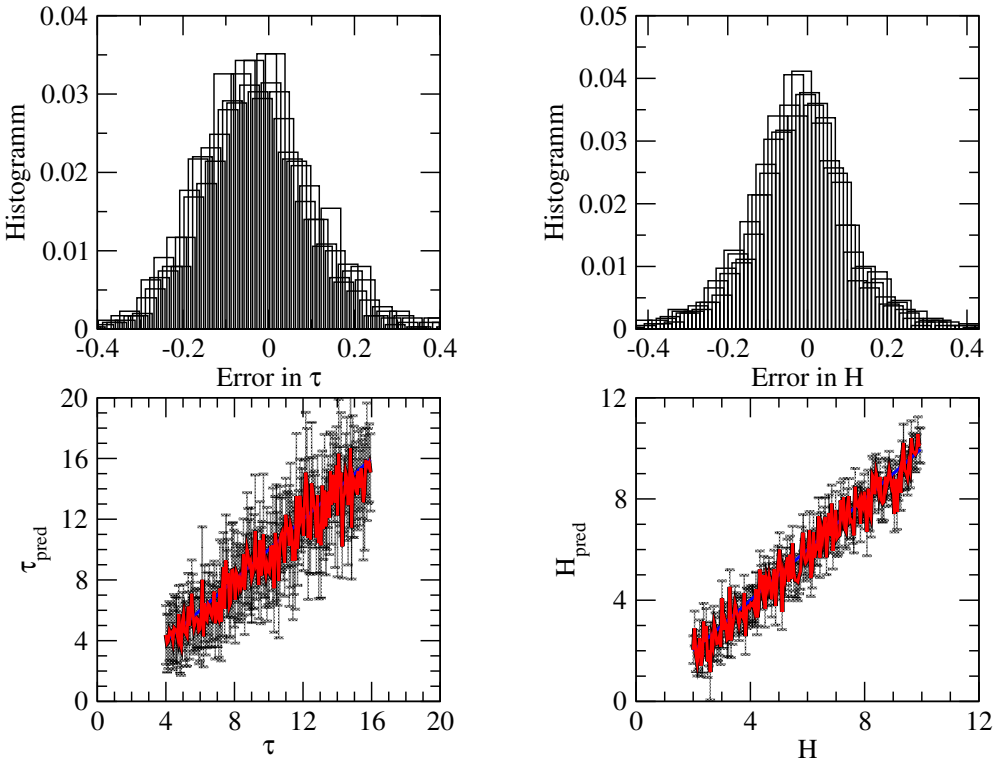

Figure 2. Retrieval results obtained with Method 1 using a deterministic network.

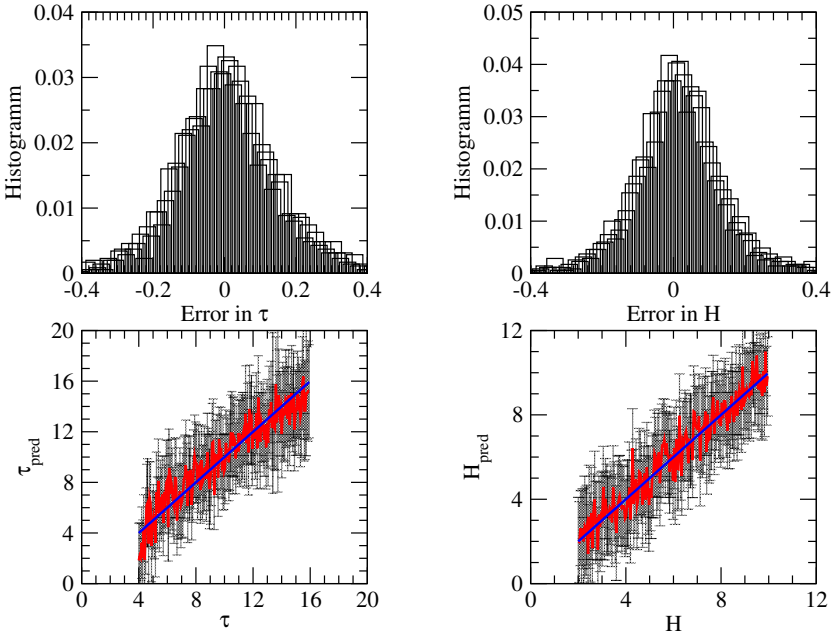

Figure 3. Retrieval results obtained with Method 1 using a Bayes-by-backprop network.

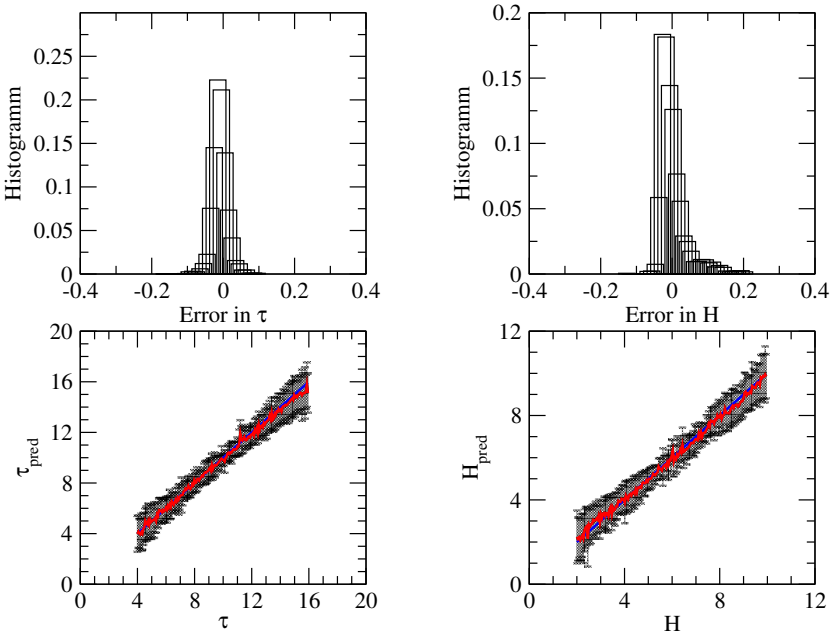

Figure 4. Retrieval results obtained with Method 1 using a dropout network.

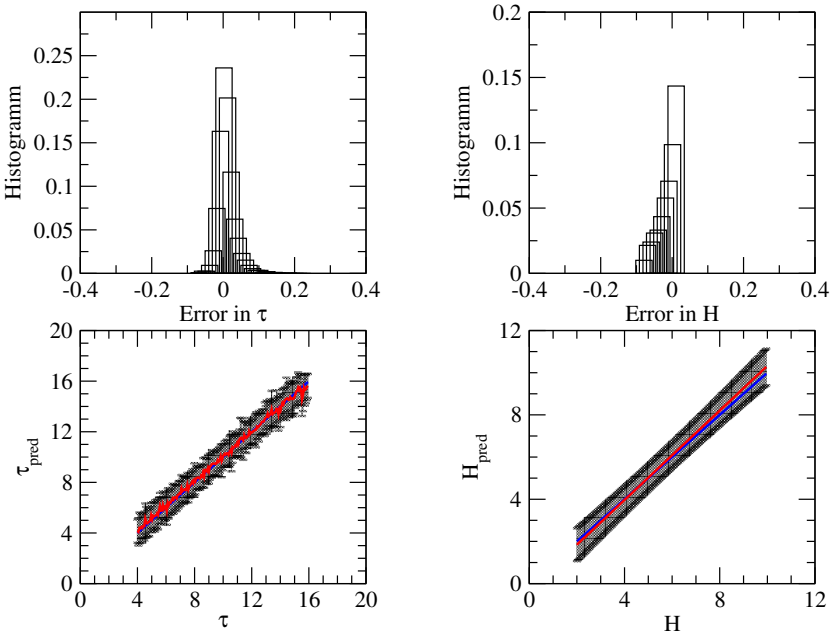

Figure 5. Retrieval results obtained with Method 1 using a batch normalized network.

3.2.2. Method 2

In order to model both heteroscedastic and epistemic uncertainties, we used the approach described in Ref. [57]. More precisely, we considered the data model with input and output noise, and used dropout to learn the heteroscedastic covariance matrix $\mathscr{C}_{\mathrm{y}}^{\delta}(\mathbf{z},\boldsymbol{\omega}) = \mathrm{diag}[\sigma_j^2(\mathbf{z},\boldsymbol{\omega})]_{j=1}^{N_{\mathrm{y}}}$ from the data (see Section 2.1). The network has the output $[\mu_j(\mathbf{z},\boldsymbol{\omega}), \sigma_j^2(\mathbf{z},\boldsymbol{\omega})] \in \mathbb{R}^{2N_{\mathrm{y}}}$ and is trained to predict the log variance $\rho_j = \log\sigma_j^2$, in which case, the likelihood loss function is given by Equations (26) and (27). Considering the set of samples $\{\boldsymbol{\omega}_t\}_{t=1}^{T}$, where $\boldsymbol{\omega}_t$ corresponds to a realization of the Bernoulli distribution $\mathsf{Z}_{l-1}^{(t)}$ and $\mathsf{W}_l^{(t)} = \mathsf{W}_l\mathsf{Z}_{l-1}^{(t)}$ for all $l = 1,\ldots,L$, we compute the predictive mean and covariance matrix as [57]

$$\mathbb{E}(\mathbf{y}) \approx \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\mu}(\mathbf{z},\boldsymbol{\omega}_t), \tag{70}$$

$$\mathrm{Cov}(\mathbf{y}) \approx \frac{1}{T}\sum_{t=1}^{T}\mathrm{diag}[\sigma_j^2(\mathbf{z},\boldsymbol{\omega}_t)]_{j=1}^{N_{\mathrm{y}}} + \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\mu}(\mathbf{z},\boldsymbol{\omega}_t)\boldsymbol{\mu}(\mathbf{z},\boldsymbol{\omega}_t)^T - \mathbb{E}(\mathbf{y})\mathbb{E}(\mathbf{y})^T, \tag{71}$$

for each noisy input $\mathbf{z}$. The first term in Equation (71) reproduces the heteroscedastic uncertainty, while the second and third terms reproduce the epistemic uncertainty.

Instead of a dropout network, a Bayes-by-backprop network can also be used to learn the heteroscedastic covariance matrix from the data. In this case, $F(\boldsymbol{\theta},\mathcal{D})$ is given by Equation (47) with $\log p(\mathcal{D}|\boldsymbol{\omega}) = -E_{\mathrm{D}}(\boldsymbol{\omega}) = -\sum_{n=1}^{N} E_{\mathrm{D}}^{(n)}(\boldsymbol{\omega})$ and $E_{\mathrm{D}}^{(n)}(\boldsymbol{\omega})$ as in Equation (27).

The results for Method 2 with a dropout and a Bayes-by-backprop network are illustrated in Figures 6 and 7.

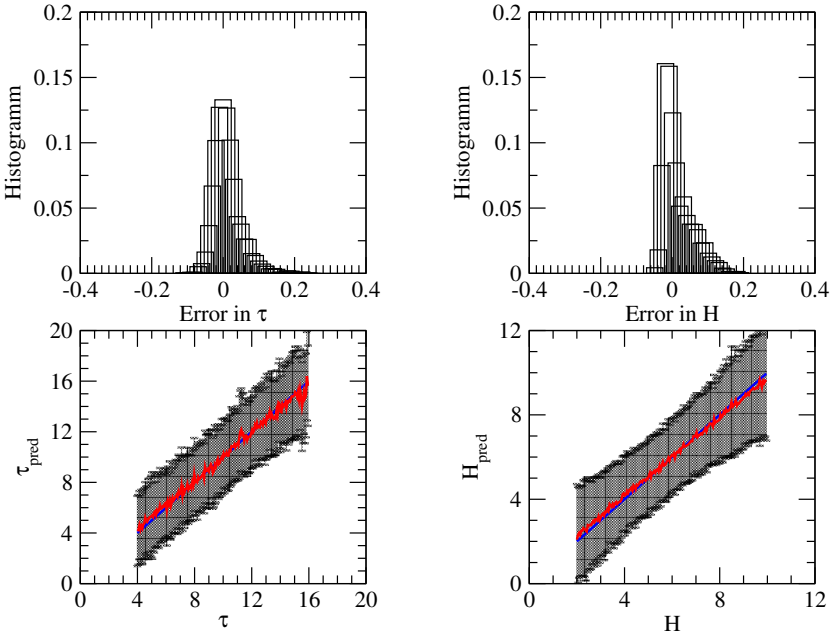

Figure 6. Retrieval results obtained with Method 2 using a dropout network.

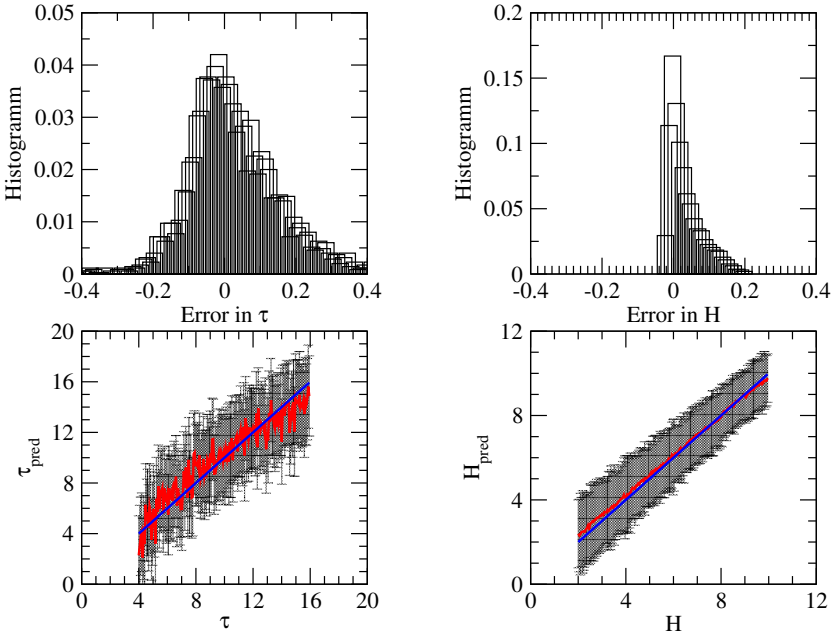

Figure 7. Retrieval results obtained with Method 2 using a Bayes-by-backprop network.

3.2.3. Method 3

Let $\widehat{\omega} = \{\mathsf{W}_l, \mathbf{b}_l\}_{l=1}^{L}$ be the parameters of a dropout network trained with exact data. Further, assume that the input data are noisy, i.e., $\mathbf{z} = \mathbf{x} + \boldsymbol{\delta}_\mathbf{x}$ with $p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mathbf{x}, \mathsf{C}_\mathbf{x}^\delta = \mathrm{diag}[\sigma_{\mathrm{x}k}^2]_{k=1}^{N_\mathrm{x}})$. In order to compute the heteroscedastic uncertainty, we forward propagate the input noise through the network. This is done by using assumed density filtering and interval arithmetic.

1. **Assumed density filtering (ADF)**. This approach was applied to neural networks by Gast and Roth [58] to replace each network activation by probability distributions. In the following, we provide a simplified justification of this approach, while for a

more pertinent analysis, we refer to Appendix D. For a linear layer ($g_l(x) = x$), the feed-forward operation (without dropout) is:

$$y_{k,l} = \sum_{j=1}^{N_{l-1}} w_{kj,l} y_{j,l-1} + b_{k,l}, \quad k = 1, \ldots, N_l, \tag{72}$$

By straightforward calculation, we find:

$$\mu_{k,l} = \mathbb{E}(y_{k,l}) = \sum_{j=1}^{N_{l-1}} w_{kj,l} \mathbb{E}(y_{j,l-1}) + b_{k,l} = \sum_{j=1}^{N_{l-1}} w_{kj,l} \mu_{j,l-1} + b_{k,l}, \tag{73}$$

and:

$$\begin{aligned} \mathbb{E}(y_{k,l})\mathbb{E}(y_{k_1,l}) = &\sum_{j=1}^{N_{l-1}} \sum_{j_1=1}^{N_{l-1}} w_{kj,l} w_{k_1 j_1,l} \mu_{j,l-1} \mu_{j_1,l-1} \\ &+ b_{k,l} \mu_{k_1,l} + b_{k_1,l} \mu_{k,l} - b_{k,l} b_{k_1,l}, \end{aligned} \tag{74}$$

yielding:

$$\begin{aligned} &\mathbb{E}(y_{k,l} y_{k_1,l}) - \mathbb{E}(y_{k,l})\mathbb{E}(y_{k_1,l}) \\ &= \sum_{j=1}^{N_{l-1}} \sum_{j_1=1}^{N_{l-1}} w_{kj,l} w_{k_1 j_1,l} [\mathbb{E}(y_{j,l-1} y_{j_1,l-1}) - \mathbb{E}(y_{j,l-1})\mathbb{E}(y_{j_1,l-1})]. \end{aligned} \tag{75}$$

Assuming that the $y_{k,l}$ are independent random variables, in which case, the covariance matrix corresponding to the column vector $[y_{1.l}, \ldots, y_{N_l l}]^T$ is diagonal, meaning that:

$$\mathbb{E}(y_{k,l} y_{k_1,l}) - \mathbb{E}(y_{k,l})\mathbb{E}(y_{k_1,l}) = \delta_{kk_1} \left[\mathbb{E}(y_{k,l}^2) - \mathbb{E}^2(y_{k,l}) \right] = \delta_{kk_1} v_{k,l}, \tag{76}$$

we obtain $v_{k,l} = \sum_{j=1}^{N_{l-1}} w_{kj,l}^2 v_{j,l-1}$. In summary, the iterative process for a linear layer is:

$$\mu_{k,0} = x_k, \quad v_{k,0} = \sigma_{xk}^2. \tag{77}$$

$$\mu_{k,l} = \sum_{j=1}^{N_{l-1}} w_{kj,l} \mu_{j,l-1} + b_{k,l}, \tag{78}$$

$$v_{k,l} = \sum_{j=1}^{N_{l-1}} w_{kj,l}^2 v_{j,l-1}, \quad l = 1, \ldots, L. \tag{79}$$

For a ReLU activation function $\mathrm{ReLU}(x) = \max(0, x)$, it was shown that:

$$\mu_{\mathrm{ReLU}k,l} = \sqrt{v_{k,l}} \phi(\alpha) + \mu_{k,l} \Phi(\alpha), \tag{80}$$

$$v_{\mathrm{ReLU}k,l} = \mu_{k,l} \sqrt{v_{k,l}} \phi(\alpha) + (\mu_{k,l}^2 + v_{k,l}) \Phi(\alpha) - \mu_{\mathrm{ReLU}k,l}^2, \tag{81}$$

where $\mu_{k,l}$ and $v_{k,l}$ are given by Equations (78) and (79), respectively, and:

$$\begin{aligned} \alpha &= \frac{\mu_{k,l}}{\sqrt{v_{k,l}}}, \\ \phi(x) &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} x^2 \right), \\ \Phi(x) &= \int_{-\infty}^{x} \phi(y) \boldsymbol{dy}. \end{aligned}$$

2. **Interval Arithmetic (IA)**. Interval arithmetic is based on an extension of the real number system to a system of closed intervals on the real axis [59]. For the intervals

X and Y, the elementary arithmetic operations are defined by the rule $X \oplus Y = \{x \oplus y | x \in X, y \in Y\}$, where the binary operation $\oplus$ can stand for addition, subtraction, multiplication, or division. This definition guarantees that $x \oplus y \in X \oplus Y$. Functions of interval arguments are defined in terms of standard set mapping, that is, the image of an interval X under a function f is the set $f(X) = \{f(x) | x \in X\}$. This is not the same as an interval function obtained from a real function f by replacing the real argument by an interval argument and the real arithmetic operations by the corresponding interval operations. The latter is called an interval extension of the real function f and is denoted by $F(X)$. As a corollary of the fundamental theorem of interval analysis, it can be shown that $f(X) \subseteq F(X)$. Interval analysis provides a simple and accessible way to assess error propagation. The iterative process for error propagation is (compared with Equations (77)–(79)):

$$Y_{k,0} = [x_k - \sigma_{xk}, x_k + \sigma_{xk}], \tag{82}$$

$$U_{k,l} = \sum_{i=1}^{N_{l-1}} w_{kj,l} Y_{j,l-1} + b_{k,l}, \tag{83}$$

$$Y_{k,l} = G_l(U_{k,l}), \ \ l = 1, \ldots, L, \tag{84}$$

while the output predictions $\mu_{k,L}$ and their standard deviations $\sigma_{k,L} = \sqrt{v_{k,L}}$ are computed as

$$\mu_{k,L} = \frac{1}{2}[\underline{Y}_{k,L} + \overline{Y}_{k,L}], \tag{85}$$

$$\sigma_{k,L} = \frac{1}{2}[\overline{Y}_{k,L} - \underline{Y}_{k,L}], \ \ k = 1, \ldots, N_L, \tag{86}$$

where $G_l(U)$ is the interval extension of the activation function g_l, and $\underline{X}$ and $\overline{X}$ are the left and right endpoints of the interval X, respectively, that is, $X = [\underline{X}, \overline{X}]$.

By assumed density filtering and interval arithmetic, the forward pass of a neural network generates not only the output predictions $\boldsymbol{\mu}_L = [\mu_{1,L}, \ldots, \mu_{N_L,L}]^T$ but also their variances $\boldsymbol{v}_L = [v_{1,L}, \ldots, v_{N_L,L}]^T$. Following Ref. [2], we consider now a network with dropout, that is, in Equations (78), (79) and (83), we replace $w_{kj,l}$ by $w_{kj,l} z_{j,l-1}$, where $z_{j,l-1} \sim \text{Bernoulli}(p)$ and the dropout probability p is the same as that used for training. For the set of samples $\{\boldsymbol{\omega}_t\}_{t=1}^T$, where $\boldsymbol{\omega}_t$ corresponds to a realization from the Bernoulli distribution $\mathsf{Z}_{l-1}^{(t)}$ and $\mathsf{W}_l^{(t)} = \mathsf{W}_l \mathsf{Z}_{l-1}^{(t)}$ for all $l = 1, \ldots, L$, we denote by $\boldsymbol{\mu}_L(\mathbf{x}, \boldsymbol{\omega}_t)$ and $\boldsymbol{v}_L(\mathbf{x}, \boldsymbol{\omega}_t)$ the outputs of the network for an input $\mathbf{x}$ corrupted by the noise $\boldsymbol{\delta}_\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_\mathbf{x}^\delta = \text{diag}[\sigma_{xk}^2]_{k=1}^{N_x})$, and compute the predictive mean and covariance matrix as (Appendix B)

$$\mathbb{E}(\mathbf{y}|\mathbf{x}) \approx \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\mu}_L(\mathbf{x}, \boldsymbol{\omega}_t), \tag{87}$$

$$\text{Cov}(\mathbf{y}|\mathbf{x}) \approx \frac{1}{T} \sum_{t=1}^{T} \text{diag}[v_{k,L}(\mathbf{x}, \boldsymbol{\omega}_t)]_{k=1}^{N_y} + \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\mu}_L(\mathbf{x}, \boldsymbol{\omega}_t) \boldsymbol{\mu}_L(\mathbf{x}, \boldsymbol{\omega}_t)^T - \mathbb{E}(\mathbf{y}|\mathbf{x}) \mathbb{E}(\mathbf{y}|\mathbf{x})^T. \tag{88}$$

From Equations (87) and (88), it is obvious that the prediction ensemble is not generated by the output of the network $\mathbf{f}(\mathbf{x}, \boldsymbol{\omega}_t)$, but by the prediction $\boldsymbol{\mu}_L(\mathbf{x}, \boldsymbol{\omega}_t)$. In summary, the algorithm involves the following steps:

1. Transform a dropout network into its assumed density filtering or interval arithmetic versions (which does not require retraining);
2. Propagate $(\mathbf{x}, v_0)$ through the dropout network and collect T output predictions and variances;
3. Compute the predictive mean and covariance matrix by means of Equations (87) and (88).

The results for Method 3 using assumed density filtering and interval arithmetic are illustrated in Figures 8 and 9, respectively. The main drawback of this method is that it requires the knowledge of the exact input data **x**. Because in atmospheric remote sensing that is not the case, Method 3 will be used for a comparison with the other two methods.

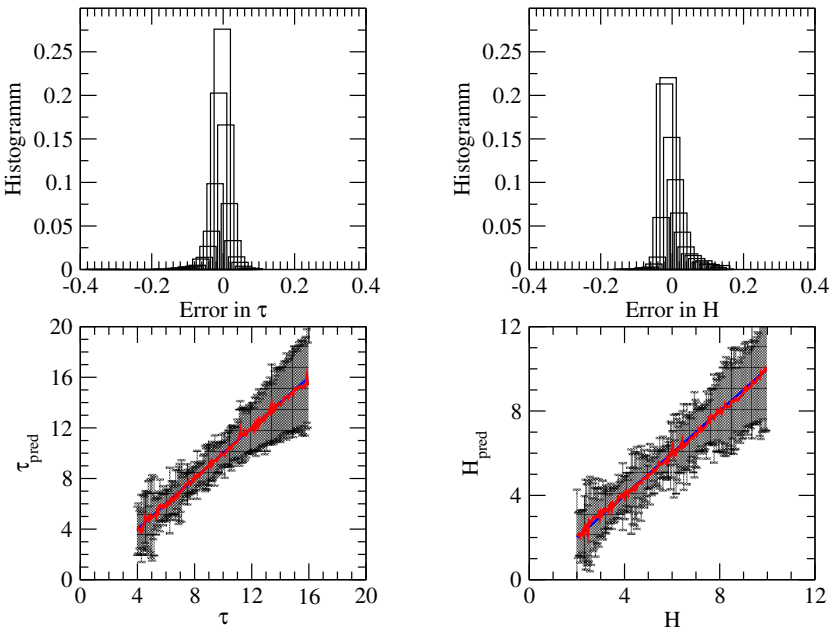

Figure 8. Retrieval results obtained with Method 3 using assumed density filtering.

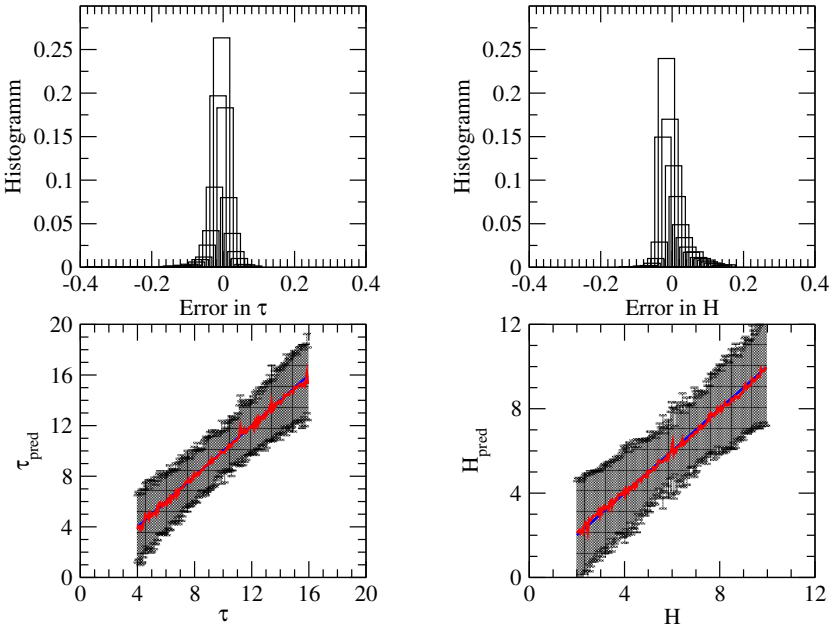

Figure 9. Retrieval results obtained with Method 3 using interval arithmetic. For a practical implementation of the algorithm, we use the interval arithmetic library INTLIB [60].

3.3. Summary of Numerical Analysis

In Table 1, we summarize the results of our numerical simulations by illustrating the average relative error and the standard deviation over the prediction set, $\mathbb{E}(\varepsilon_x) \pm \sqrt{\mathbb{E}([\varepsilon_x - \mathbb{E}(\varepsilon_x)]^2)}$ and $\mathbb{E}(\sigma_x)$, respectively, where x stands for the cloud optical thickness τ and the cloud top height H. The accuracy of a method is reflected by the bias of the error $\mathbb{E}(\varepsilon_x)$ and the interval about the mean with length $\sqrt{\mathbb{E}([\varepsilon_x - \mathbb{E}(\varepsilon_x)]^2)}$, while the precision is reflected by the standard deviation $\mathbb{E}(\sigma_x)$ (which determines the length of the

uncertainty interval). Note that (i) $\sqrt{\mathbb{E}([\varepsilon_x - \mathbb{E}(\varepsilon_x)]^2)}$ reproduces the square root of the diagonal elements of the conditional average covariance matrix $\mathbb{E}(\mathscr{C}_{\mathbf{y}}|\mathcal{D}_{\text{test}})$ of all network errors over the test set $\mathcal{D}_{\text{test}}$; and (ii) roughly speaking, the epistemic (model) uncertainties are large if there are large variations around the mean.

Table 1. Average relative error $\mathbb{E}(\varepsilon_x) \pm \sqrt{\mathbb{E}([\varepsilon_x - \mathbb{E}(\varepsilon_x)]^2)}$ and standard deviation $\mathbb{E}(\sigma_x)$ over the prediction set for the methods used to solve direct and inverse problems. The parameter x stands for the cloud optical thickness τ and cloud top height H. In the case of the inverse problem, the results correspond to Method 1 with a deterministic network (1a), Bayes-by-backprop (1b), dropout (1c), and batch normalization (1d); Method 2 with dropout (2a) and Bayes-by-backprop (2b); and Method 3 with assumed density filtering (3a) and interval arithmetic (3b).

Method	x	Error	Std. Deviation
Direct Problem	τ	$-2.94 \times 10^{-2} \pm 6.67 \times 10^{-2}$	2.04×10^{-1}
	H	$-5.87 \times 10^{-2} \pm 1.72 \times 10^{-1}$	3.66×10^{-1}
1a	τ	$-2.78 \times 10^{-2} \pm 1.30 \times 10^{-1}$	8.92×10^{-1}
	H	$-1.96 \times 10^{-2} \pm 1.29 \times 10^{-1}$	4.36×10^{-1}
1b	τ	$3.39 \times 10^{-2} \pm 1.66 \times 10^{-1}$	9.23×10^{-1}
	H	$1.83 \times 10^{-2} \pm 1.33 \times 10^{-1}$	8.11×10^{-1}
1c	τ	$-8.75 \times 10^{-3} \pm 2.25 \times 10^{-2}$	3.23×10^{-1}
	H	$3.45 \times 10^{-3} \pm 4.11 \times 10^{-2}$	2.31×10^{-1}
1d	τ	$1.01 \times 10^{-2} \pm 2.41 \times 10^{-2}$	3.54×10^{-1}
	H	$4.37 \times 10^{-3} \pm 2.73 \times 10^{-2}$	2.29×10^{-1}
2a	τ	$1.16 \times 10^{-2} \pm 4.05 \times 10^{-2}$	8.24×10^{-1}
	H	$1.63 \times 10^{-2} \pm 4.21 \times 10^{-2}$	6.72×10^{-1}
2b	τ	$3.10 \times 10^{-2} \pm 1.38 \times 10^{-1}$	9.63×10^{-1}
	H	$3.31 \times 10^{-2} \pm 4.57 \times 10^{-2}$	4.88×10^{-1}
3a	τ	$-6.67 \times 10^{-3} \pm 2.48 \times 10^{-2}$	6.55×10^{-1}
	H	$8.18 \times 10^{-4} \pm 3.18 \times 10^{-2}$	4.76×10^{-1}
3b	τ	$-7.08 \times 10^{-3} \pm 2.53 \times 10^{-2}$	7.82×10^{-1}
	H	$1.56 \times 10^{-3} \pm 3.33 \times 10^{-2}$	6.53×10^{-1}

The results in Figures 1–9 and Table 1 can be summarized as follows:

1. For the direct problem, the neural network method used in conjunction with a Bayesian inversion method provides satisfactory accuracy, but does not correctly predict the uncertainty. The reason for this is that the forward model is not nearly linear, which is the main assumption for computing the uncertainty in the retrieval.
2. For the inverse problem, the following features are apparent.
   (a) Method 1 using a deterministic and Bayes-by-backprop network yields a low accuracy, while the method using dropout and batch-normalized networks provides high accuracy;
   (b) Method 2 using a dropout network has an acceptable accuracy. For cloud top height retrieval, the method using a Bayes-by-backprop network has a similar accuracy, but for cloud optical thickness retrieval, the accuracy is low. Possible reasons for the loss of accuracy of a Bayes-by-backprop network can be (i) a non-optimal training and/or the use of the prior $p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}; \mathbf{0}, \mathsf{I})$ instead of that given by Equation (45) (recall that for $p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}; \mathbf{0}, \mathsf{I})$, the KL divergence $\mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\omega})|p(\boldsymbol{\omega}))$ can be computed analytically).
   (c) Method 3 with assumed density filtering and interval arithmetic is comparable with Method 2 using a dropout network, that is, the method has a high accuracy.

(d) From the methods with reasonable accuracy, by using a dropout network, Method 2 predicts similar uncertainties to Method 3 with assumed density filtering and interval arithmetic. In contrast, using dropout and batch normalized networks, Method 1 provides lower uncertainties, dominated by epistemic uncertainties. In general, it seems that Method 1 predicts a too small heteroscedastic uncertainty. Possibly, this is due to the fact that we trained a neural network with exact data, and for this retrieval example, the predictive distribution, represented as a convolution integral over the input noise distribution, does not correctly reproduce the aleatoric uncertainty.

(e) Because $\sqrt{\mathbb{E}([\varepsilon_x - \mathbb{E}(\varepsilon_x)]^2)} < \mathbb{E}(\sigma_x)$, we deduce that the conditional average covariance matrix $\mathbb{E}(\mathscr{C}_{\mathbf{y}}|\mathcal{D}_{\text{test}})$ does not coincide with the predictive covariance matrix $\text{Cov}(\mathbf{y})$, which reflects the uncertainty.

4. Conclusions

We presented several neural networks for predicting uncertainty in an atmospheric remote sensing. The neural networks are designed in a Bayesian framework and are devoted to the solution of direct and inverse problems.

1. For solving the direct problem, we considered a neural network for simulating the radiative transfer model, computed of the epistemic covariance matrix from the statistics of all network errors over the data set, solving the inverse problem by a Bayesian approach, and determined the uncertainty in the retrieval by assuming that the forward model is nearly linear.
2. For solving the inverse problem, two neural network methods, relying on different assumptions, were implemented:
   (a) The first method uses deterministic and stochastic (Bayes-by-backprop, dropout, and batch normalization) networks to compute the epistemic covariance matrix and under the assumption that the predictive distribution of the network output is the convolution of the predictive distribution for a noise-free input with the input noise distribution, estimates the covariance matrix;
   (b) the second method uses dropout and Bayes-by-backprop to learn the heteroscedastic covariance matrix from the data.

In addition, for solving the inverse problem, a third method that uses a dropout network and forward propagates the input noise through the network by using assumed density filtering and interval arithmetic was designed. Because this method requires the knowledge of the exact input data, it was used only for testing purposes.

Our numerical analysis has shown that a dropout network that is used to learn the heteroscedastic covariance matrix from the data is appropriate for predicting the uncertainty associated with the retrieval of cloud parameters from EPIC measurements. In fact, the strengths of a dropout network are (i) its capability to avoid overfitting and (ii) its stochastic character (the method is equivalent to a Bayesian approximation).

All neural network algorithms are implemented in FORTRAN and incorporated in a common tool. In the future, we intend to implement the algorithms in the high-level programming language Python and use the deep learning library PyTorch. This software library has a variety of network architectures that provide auto-differentiation and support GPUs to enable fast and efficient computation. The Python tool will be released through a public repository to make the methods available to the scientific community.

Author Contributions: Conceptualization, A.D. (Adrian Doicu) and A.D. (Alexandru Doicu); software, A.D. (Adrian Doicu), A.D. (Alexandru Doicu) and D.S.E.; formal analysis, D.L. and T.T.; writing—original draft preparation, A.D. (Adrian Doicu) and A.D. (Alexandru Doicu); writing—review and editing, D.S.E., D.L. and T.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

According to the Laplace approximation, we expand the loss function:

$$E(\boldsymbol{\omega}) = \frac{1}{2}\sum_{n=1}^{N}[\mathbf{y}^{(n)} - \mathbf{f}(\mathbf{z}^{(n)}, \boldsymbol{\omega})]^T[\mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z}^{(n)}, \boldsymbol{\omega})]^{-1}[\mathbf{y}^{(n)} - \mathbf{f}(\mathbf{z}^{(n)}, \boldsymbol{\omega})] + \frac{1}{2}\boldsymbol{\omega}^T\mathsf{C}_{\boldsymbol{\omega}}^{-1}\boldsymbol{\omega}, \tag{A1}$$

around the point estimate $\widehat{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\mathrm{MAP}}$ and use the optimality condition $\nabla E(\widehat{\boldsymbol{\omega}}) = \mathbf{0}$, to obtain:

$$E(\boldsymbol{\omega}) = E(\widehat{\boldsymbol{\omega}}) + \frac{1}{2}\Delta\boldsymbol{\omega}^T\mathsf{H}(\widehat{\boldsymbol{\omega}})\Delta\boldsymbol{\omega}, \tag{A2}$$

$$\Delta\boldsymbol{\omega} = \boldsymbol{\omega} - \widehat{\boldsymbol{\omega}}, \tag{A3}$$

$$[\mathsf{H}(\widehat{\boldsymbol{\omega}})]_{ij} = \frac{\partial E}{\partial\omega_i\partial\omega_j}(\widehat{\boldsymbol{\omega}}), \tag{A4}$$

and further (cf. Equation (13)):

$$p(\boldsymbol{\omega}|\mathcal{D}) \propto \exp[-E(\boldsymbol{\omega})] \propto \exp\Big[-\frac{1}{2}(\boldsymbol{\omega} - \widehat{\boldsymbol{\omega}})^T\mathsf{H}(\widehat{\boldsymbol{\omega}})(\boldsymbol{\omega} - \widehat{\boldsymbol{\omega}})\Big]. \tag{A5}$$

Inserting Equations (12) and (A5) in the expression of the predictive distribution as given by Equation (28) we find:

$$p(\mathbf{y}|\mathbf{z}, \mathcal{D}) = \int p(\mathbf{y}|\mathbf{z}, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathcal{D})\mathrm{d}\boldsymbol{\omega} \propto \int \exp\Big\{-\frac{1}{2}[\mathbf{y} - \mathbf{f}(\mathbf{z}, \boldsymbol{\omega})]^T[\mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z}, \boldsymbol{\omega})]^{-1}[\mathbf{y} - \mathbf{f}(\mathbf{z}, \boldsymbol{\omega})]\Big\} \times \exp\Big[-\frac{1}{2}(\boldsymbol{\omega} - \widehat{\boldsymbol{\omega}})^T\mathsf{H}(\widehat{\boldsymbol{\omega}})(\boldsymbol{\omega} - \widehat{\boldsymbol{\omega}})\Big]\mathrm{d}\boldsymbol{\omega}. \tag{A6}$$

In Equation (A6), the model function $\mathbf{f}(\mathbf{x}, \boldsymbol{\omega})$ can be approximated by a linear Taylor expansion around $\widehat{\boldsymbol{\omega}}$, that is:

$$\mathbf{f}(\mathbf{x}, \boldsymbol{\omega}) \approx \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}}) + \mathsf{K}_{\boldsymbol{\omega}}(\mathbf{x}, \widehat{\boldsymbol{\omega}})(\boldsymbol{\omega} - \widehat{\boldsymbol{\omega}}), \tag{A7}$$

where:

$$\mathsf{K}_{\boldsymbol{\omega}}(\mathbf{x}, \boldsymbol{\omega}) = \frac{\partial\mathbf{f}}{\partial\boldsymbol{\omega}}(\mathbf{x}, \boldsymbol{\omega}) \tag{A8}$$

is the Jacobian of $\mathbf{f}$ with respect to $\boldsymbol{\omega}$. Substituting Equation (A7) into Equation (A6), approximating $\mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z}, \boldsymbol{\omega}) \approx \mathscr{C}_{\mathbf{y}}^{\delta}(\mathbf{z}, \widehat{\boldsymbol{\omega}})$, and computing the integral over $\boldsymbol{\omega}$ gives the representation (32) for the predictive distribution $p(\mathbf{y}|\mathbf{z}, \mathcal{D})$.

Appendix B

For $p(\mathbf{y}|\mathbf{x},\boldsymbol{\omega}) = \mathcal{N}(\mathbf{y}, \mathbf{f}(\mathbf{x},\boldsymbol{\omega}), \mathsf{C}_{\mathbf{y}}^{\delta} = \sigma_{\mathbf{y}}^2\mathsf{I})$, we compute the predictive mean of the output $\mathbf{y}$ given the input $\mathbf{x}$, as

$$\begin{aligned}
\mathbb{E}(\mathbf{y}) &= \int \mathbf{y} p(\mathbf{y}|\mathbf{x},\mathcal{D})\mathrm{d}\mathbf{y} \\
&\approx \int \mathbf{y} q_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})\mathrm{d}\mathbf{y} \\
&= \int \mathbf{y}\left(\int p(\mathbf{y}|\mathbf{x},\boldsymbol{\omega}) q_{\boldsymbol{\theta}}(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega}\right)\mathrm{d}\mathbf{y} \\
&= \int\left(\int \mathbf{y}\mathcal{N}(\mathbf{y};\mathbf{f}(\mathbf{x},\boldsymbol{\omega}),\sigma_{\mathbf{y}}^2\mathsf{I})\mathrm{d}\mathbf{y}\right) q_{\boldsymbol{\theta}}(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} \\
&= \int \mathbf{f}(\mathbf{x},\boldsymbol{\omega}) q_{\boldsymbol{\theta}}(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} \\
&\approx \frac{1}{T}\sum_{t=1}^{T} \mathbf{f}(\mathbf{x},\boldsymbol{\omega}_t),
\end{aligned}$$

and by using the result:

$$\begin{aligned}
\mathbb{E}(\mathbf{y}\mathbf{y}^T) &= \int \mathbf{y}\mathbf{y}^T p(\mathbf{y}|\mathbf{x},\mathcal{D})\mathrm{d}\mathbf{y} \\
&\approx \int \mathbf{y}\mathbf{y}^T q_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})\mathrm{d}\mathbf{y} \\
&= \int \mathbf{y}\mathbf{y}^T\left(\int p(\mathbf{y}|\mathbf{x},\boldsymbol{\omega}) q_{\boldsymbol{\theta}}(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega}\right)\mathrm{d}\mathbf{y} \\
&= \int\left(\int \mathbf{y}\mathbf{y}^T\mathcal{N}(\mathbf{y};\mathbf{f}(\mathbf{x},\boldsymbol{\omega}),\sigma_{\mathbf{y}}^2\mathsf{I})\mathrm{d}\mathbf{y}\right) q_{\boldsymbol{\theta}}(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} \\
&= \int [\sigma_{\mathbf{y}}^2\mathsf{I} + \mathbf{f}(\mathbf{x},\boldsymbol{\omega})\mathbf{f}(\mathbf{x},\boldsymbol{\omega})^T] q_{\boldsymbol{\theta}}(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} \\
&\approx \sigma_{\mathbf{y}}^2\mathsf{I} + \frac{1}{T}\sum_{t=1}^{T} \mathbf{f}(\mathbf{x},\boldsymbol{\omega}_t)\mathbf{f}(\mathbf{x},\boldsymbol{\omega}_t)^T,
\end{aligned}$$

the covariance matrix as

$$\mathrm{Cov}(\mathbf{y}) \approx \sigma_{\mathbf{y}}^2\mathsf{I} + \frac{1}{T}\sum_{t=1}^{T} \mathbf{f}(\mathbf{x},\boldsymbol{\omega}_t)\mathbf{f}(\mathbf{x},\boldsymbol{\omega}_t)^T - \mathbb{E}(\mathbf{y})\mathbb{E}(\mathbf{y})^T.$$

The predictive mean and covariance matrix of a dropout network with assumed density filtering given by Equations (87) and (88), respectively, can be computed in the same manner by taking into account that in this case, the predictive power of the network is given by

$$p(\mathbf{y}|\mathbf{x},\boldsymbol{\omega}) = \mathcal{N}(\mathbf{y};\boldsymbol{\mu}_L(\mathbf{x},\boldsymbol{\omega}), \mathrm{diag}[v_{j,L}(\mathbf{x},\boldsymbol{\omega})]_{j=1}^{N_y}),$$

where $\boldsymbol{\mu}_L = [\mu_{1.L},\ldots,\mu_{N_L L}]^T$ and $\boldsymbol{v}_L = [v_{1.L},\ldots,v_{N_L L}]^T$ are the output predictions and their variances.

Appendix C

In this appendix, which is borrowed from [37], we show that the variational free energy has the standard form representation of the dropout loss function (as the sum of a square loss function and an L_2 regularization term).

For $\boldsymbol{\omega} = \{\mathsf{W}_l, \mathbf{b}_l\}_{l=1}^{L}$ and $\mathsf{W}_l = [\mathbf{w}_{k,l}]_{k=1}^{N_{l-1}} \in \mathbb{R}^{N_l \times N_{l-1}}$, we construct the variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ as

$$q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) = \prod_{l=1}^{L} q_{\boldsymbol{\theta}}(\mathsf{W}_l, \mathbf{b}_l) = \left(\prod_{l=1}^{L} q(\mathsf{W}_l)\right)\left(\prod_{l=1}^{L} q(\mathbf{b}_l)\right), \tag{A9}$$

with:

$$q(\mathsf{W}_l) = \prod_{k=1}^{N_{l-1}} q(\mathbf{w}_{k,l}), \tag{A10}$$

$$q(\mathbf{w}_{k,l}) = p\mathcal{N}(\mathbf{m}_{k,l}, \sigma^2 \mathsf{I}_{N_l}) + (1-p)\mathcal{N}(\mathbf{0}, \sigma^2 \mathsf{I}_{N_l}), \tag{A11}$$

and:

$$q(\mathbf{b}_l) = \mathcal{N}(\mathbf{n}_l, \sigma^2 \mathsf{I}_{N_l}). \tag{A12}$$

Here, $p \in [0,1]$ is an activation probability, $\sigma > 0$ a scalar, and $\mathsf{M}_l = [\mathbf{m}_{k,l}]_{k=1}^{N_{l-1}}$ and $\mathbf{n}_l$ are a variational parameter to be determined; thus, $\boldsymbol{\theta} = \{\mathsf{M}_l, \mathbf{n}_l\}_{l=1}^{L}$. The key point of the derivation is the representation of $q(\mathbf{w}_{k,l})$ as a mixture of two Gaussians with the same variance (cf. Equation (A11)). When the standard deviation σ tends towards 0, the Gaussians tend to Dirac delta distributions showing that when sampling from the mixture of the two Gaussians is equivalent to sampling from a Bernoulli distribution that returns either the value $\mathbf{0}$ with probability $1-p$ or $\mathbf{m}_{k,l}$ with probability p, that is:

$$q(\mathbf{w}_{k,l}) = \left\{ \begin{array}{ll} \mathbf{m}_{k,l} & \text{with probability } p \\ \\ \mathbf{0} & \text{with probability } 1-p \end{array} \right|.$$

As a result, we obtain:

$$\begin{aligned} \mathbf{W}_l &= [\mathbf{w}_{1,l}, \mathbf{w}_{2,l}, \ldots, \mathbf{w}_{N_{l-1},l}] \\ &= [\mathbf{m}_{1,l}, \mathbf{m}_{2,l}, \ldots, \mathbf{m}_{N_{l-1},l}] \begin{bmatrix} z_{1,l-1} & 0 & \ldots & 0 \\ 0 & z_{2,l-1} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & z_{N_{l-1},l-1} \end{bmatrix} \\ &= \mathbf{M}_l \mathbf{Z}_{l-1}, \end{aligned}$$

where $\mathsf{Z}_{l-1} = \mathrm{diag}[z_{k,l-1}]_{k=1}^{N_{l-1}}$ with $z_{k,l-1} \sim \mathrm{Bernoulli}(p)$. Note that the binary variable $z_{k,l-1}$ corresponds to the unit k in layer $l-1$ being dropped out as an input to layer l. For $\mathbf{b}_l$, we take into account that $q(\mathbf{b}_l) = \lim_{\sigma \to 0} \mathcal{N}(\mathbf{n}_l, \sigma^2 \mathsf{I}_{N_l}) = \delta(\mathbf{b}_l - \mathbf{n}_l)$; hence, in the limit $\sigma \to 0$, $\mathbf{b}_l$ is approximately deterministic, and we have $\mathbf{b}_l \approx \mathbf{n}_l$.

The variational parameters M_l and $\mathbf{n}_l$ are computed by minimizing the variational free energy (39), that is:

$$F(\boldsymbol{\theta}, \mathcal{D}) = -\int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \log p(\mathcal{D}|\boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega} + \mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) | p(\boldsymbol{\omega})). \tag{A13}$$

The two terms in the above equation are computed as follows.

1. For the first term, written as

$$\int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \log p(\mathcal{D}|\boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega} = \sum_{n=1}^{N} \int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega}, \tag{A14}$$

we use the following reparameterization trick. Let $\boldsymbol{\epsilon} \sim q(\boldsymbol{\epsilon})$ be an auxiliary variable representing the stochasticity during the training, such that $\boldsymbol{\omega} = t(\boldsymbol{\epsilon}, \boldsymbol{\theta})$ for some function t. Assuming that $q_{\boldsymbol{\theta}}(\boldsymbol{\omega}|\boldsymbol{\epsilon}) = \delta(\boldsymbol{\omega} - t(\boldsymbol{\epsilon}, \boldsymbol{\theta}))$, we find:

$$q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) = \int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}|\boldsymbol{\epsilon})\, q(\boldsymbol{\epsilon}) \mathrm{d}\boldsymbol{\epsilon} = \int \delta(\boldsymbol{\omega} - t(\boldsymbol{\epsilon}, \boldsymbol{\theta}))\, q(\boldsymbol{\epsilon}) \mathrm{d}\boldsymbol{\epsilon}, \tag{A15}$$

implying:

$$\begin{aligned} &\int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega} \\ &= \int \int \delta(\boldsymbol{\omega} - t(\boldsymbol{\epsilon}, \boldsymbol{\theta})) q(\boldsymbol{\epsilon}) \log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega} \mathrm{d}\boldsymbol{\epsilon} \\ &= \int q(\boldsymbol{\epsilon}) \log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, t(\boldsymbol{\epsilon}, \boldsymbol{\theta}))\, \mathrm{d}\boldsymbol{\epsilon}. \end{aligned} \tag{A16}$$

Computing the above integral by a Monte Carlo approach with a single sample $\widehat{\boldsymbol{\epsilon}} \sim q(\boldsymbol{\epsilon})$, *yields:*

$$\int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega} = \log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \widehat{\boldsymbol{\omega}}), \tag{A17}$$

where $\widehat{\boldsymbol{\omega}} = t(\widehat{\boldsymbol{\epsilon}}, \boldsymbol{\theta})$. In our case and in view of Equations (A10)–(A12), we reparametrize the integrands by setting:

$$\mathsf{W}_l = (\mathsf{M}_l + \sigma \mathsf{S}_l)\mathsf{Z}_{l-1} + \sigma \mathsf{S}_l(\mathsf{I}_{N_{l-1}} - \mathsf{Z}_{l-1}), \tag{A18}$$

$$\mathbf{b}_l = \mathbf{n}_l + \sigma \boldsymbol{\epsilon}_l, \tag{A19}$$

where:

$$\mathsf{S}_l = [\mathbf{s}_{k,l}]_{k=1}^{N_{l-1}}, \;\; \mathbf{s}_{k,l} \sim \mathcal{N}(\mathbf{0}, \mathsf{I}_{N_l}), \tag{A20}$$

$$\mathsf{Z}_{l-1} = \mathrm{diag}[z_{k,l-1}]_{k=1}^{N_{l-1}}, \;\; z_{k,l-1} \sim \mathrm{Bernoulli}(p), \tag{A21}$$

$$\boldsymbol{\epsilon}_l \sim \mathcal{N}(\mathbf{0}, \mathsf{I}_{N_l}), \tag{A22}$$

to obtain:

$$\int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \log p(\mathcal{D}|\boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega} = \sum_{n=1}^{N} \log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \widehat{\boldsymbol{\omega}}_n), \tag{A23}$$

where:

$$\widehat{\boldsymbol{\omega}}_n = \{\widehat{\mathsf{W}}_l^{(n)}, \widehat{\mathbf{b}}_l^{(n)}\}_{l=1}^{L}, \tag{A24}$$

$$\widehat{\mathsf{W}}_l^{(n)} = (\mathsf{M}_l + \sigma \widehat{\mathsf{S}}_l^{(n)})\widehat{\mathsf{Z}}_{l-1}^{(n)} + \sigma \widehat{\mathsf{S}}_l^{(n)}(\mathsf{I}_{N_{l-1}} - \widehat{\mathsf{Z}}_{l-1}^{(n)}), \tag{A25}$$

$$\widehat{\mathbf{b}}_l^{(n)} = \mathbf{n}_l + \sigma \widehat{\boldsymbol{\epsilon}}_l^{(n)}, \tag{A26}$$

for the realizations $\widehat{\mathsf{S}}_l^{(n)}$, $\widehat{\mathsf{Z}}_{l-1}^{(n)}$, and $\widehat{\boldsymbol{\epsilon}}_l^{(n)}$ given by Equations (A20)–(A22). Taking the limit $\sigma \to 0$, we find that the realizations $\widehat{\mathsf{W}}_l^{(n)}$ and $\widehat{\mathbf{b}}_l^{(n)}$ can be approximated as

$$\widehat{\mathsf{W}}_l^{(n)} \approx \mathsf{M}_l \widehat{\mathsf{Z}}_{l-1}^{(n)}, \;\; \widehat{\mathbf{b}}_l^{(n)} \approx \mathbf{n}_l. \tag{A27}$$

2. In the case of W_l, the second term in Equation (A13) is the KL divergence between a mixture of Gaussians and a single Gaussian, that is:

$$\mathrm{KL}(q(\mathsf{W}_l)|p(\mathsf{W}_l)) = \int q(\mathsf{W}_l) \log \left[\frac{q(\mathsf{W}_l)}{p(\mathsf{W}_l)} \right] \mathrm{d}\mathsf{W}_l,$$

where $q(\mathsf{W}_l)$ is as in Equations (A10) and (A11) and $p(\mathsf{W}_l) = \prod_{k=1}^{N_{l-1}} p(\mathbf{w}_{k,l})$ with $p(\mathbf{w}_{k,l}) = \mathcal{N}(\mathbf{0}, \mathsf{I}_{N_l})$. This term can be evaluated by using the following result: For K, $N \in \mathbb{N}$, let $\mathbf{p} = (p_1, \ldots, p_K)$ be a probability vector, $q(\mathbf{x}) = \sum_{k=1}^{K} p_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ with $\mathbf{x} \in \mathbb{R}^N$ a mixture of Gaussians with K components, and $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathsf{I}_N)$. Then, for sufficiently large N, we have the approximation:

$$\mathrm{KL}(q(\mathbf{x})|p(\mathbf{x})) \approx \sum_{k=1}^{K} \Big[\boldsymbol{\mu}_k^T \boldsymbol{\mu}_k + \mathrm{tr}(\boldsymbol{\Sigma}_k) - N(1 + \log 2\pi) - \log |\boldsymbol{\Sigma}_k| \Big]. \tag{A28}$$

Consequently, for large numbers of hidden units N_l, $l = 1, \ldots, L$, we find:

$$\mathrm{KL}(q(\mathsf{W}_l)|p(\mathsf{W}_l)) \approx N_l N_{l-1} (\sigma^2 - \log(\sigma^2) - 1) + \frac{p}{2} \sum_{k=1}^{N_{l-1}} \mathbf{m}_{k,l}^T \mathbf{m}_{k,l} + C, \tag{A29}$$

where C is a constant. In the case of $\mathbf{b}_l$, the KL divergence $\mathrm{KL}(q(\mathbf{b}_l)|p(\mathbf{b}_l))$, where (cf. Equation (A12)) $q(\mathbf{b}_l) = \mathcal{N}(\mathbf{n}_l, \sigma^2 \mathsf{I}_{N_l})$ and $p(\mathbf{b}_l) = \mathcal{N}(\mathbf{0}, \mathsf{I}_{N_l})$, is a mixture of two single Gaussian and can be analytically computed as

$$\mathrm{KL}(q(\mathbf{b}_l)|p(\mathbf{b}_l)) = \frac{1}{2} [\mathbf{n}_l^T \mathbf{n}_l + N_l (\sigma^2 - \log(\sigma^2) - 1)] + C. \tag{A30}$$

Collecting all the results, we obtain:

$$F(\boldsymbol{\theta}, \mathcal{D}) = -\sum_{n=1}^{N} \log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \widehat{\boldsymbol{\omega}}_n) + \frac{1}{2} \sum_{l=1}^{L} \Big(p \|\mathsf{M}_l\|_2^2 + \|\mathbf{n}_l\|_2^2 \Big).$$

Thus, the variational free energy $F(\boldsymbol{\theta}, \mathcal{D})$ has the standard form representation as the sum of a square loss function and an L_2 regularization term.

Appendix D

In this appendix, we describe the uncertainty propagation based on assumed density filtering by following the analysis given in [58].

The feed-forward operation of a neural network can be written as (cf. Equations (2) and (3)):

$$\mathbf{y}_l = \mathbf{f}_l(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l) = g_l(\mathsf{W}_l \mathbf{y}_{l-1} + \mathbf{b}_l),$$

where $\boldsymbol{\omega}_l = \{\mathsf{W}_l, \mathbf{b}_l\}$. Thus, each layer function $\mathbf{f}_l(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l)$ is a nonlinear transformation of the previous activation $\mathbf{y}_{l-1}$ parametrized by $\boldsymbol{\omega}_l$. The deep neural network can then be expressed as a succession of nonlinear layers:

$$\mathbf{f}(\mathbf{x}, \boldsymbol{\omega}) = \mathbf{f}_L(\mathbf{f}_{L-1}(\ldots \mathbf{f}_1(\mathbf{y}_0; \boldsymbol{\omega}_1)).$$

To formalize the deep probabilistic model, we replace each activation, including input and output, by probability distributions. In particular, we assume that the joint density of all activations is given by

$$\begin{aligned} p(\mathbf{y}_{0:L}) &= p(\mathbf{y}_0) \prod_{l=1}^{L} p(\mathbf{y}_l|\mathbf{y}_{l-1}), \\ p(\mathbf{y}_l|\mathbf{y}_{l-1}) &= \delta(\mathbf{y}_l - \mathbf{f}_l(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l)), \\ p(\mathbf{y}_0) &= \prod_{k=1}^{N_l} \mathcal{N}(y_{k,0}; x_k, \sigma_{\mathbf{x}k}^2), \end{aligned}$$

where $p(\mathbf{y}_{0:L}) = p(\mathbf{y}_0, \ldots, \mathbf{y}_L)$ and$\mathbf{C}_{\mathbf{x}}^{\delta} = \text{diag}[\sigma_{\mathbf{x}k}^2]_{k=1}^{N_x}$. Because this distribution is intractable, we apply the assumed density filtering approach to the network activations. The goal of this approach is to find a tractable approximation $q(\mathbf{y}_{0:L})$ of $p(\mathbf{y}_{0:L})$, that is:

$$p(\mathbf{y}_{0:L}) \approx q(\mathbf{y}_{0:L}),$$

where:

$$\begin{aligned} q(\mathbf{y}_{0:L}) &= q(\mathbf{y}_0) \prod_{l=1}^{L} q(\mathbf{y}_l), \\ q(\mathbf{y}_l) &= \prod_{k=1}^{N_l} \mathcal{N}(y_{k,l}; \mu_{k,l}, v_{k,l}), \\ q(\mathbf{y}_0) &= p(\mathbf{y}_0), \end{aligned}$$

and $\mu_{k,l}$ and $v_{k,l}$ are the mean and variance of the activation of unit k in layer l, respectively. Thus, starting from input activation $q(\mathbf{y}_0) = p(\mathbf{y}_0)$, we approximate subsequent layer activations by independent Gaussian distributions $q(\mathbf{y}_l)$. To compute the approximant $q(\mathbf{y}_{0:L})$, we use an iterative process (layer by layer) initialized by $q(\mathbf{y}_0) = p(\mathbf{y}_0)$. In particular, for a layer $l \geq 1$, we assume that $q(\mathbf{y}_0), \ldots, q(\mathbf{y}_{l-1})$ are known, or equivalently, that $\{(\boldsymbol{\mu}_{l_1}, \boldsymbol{v}_{l_1})\}_{l_1=0}^{l-1}$ are known, and aim to compute $(\boldsymbol{\mu}_l, \boldsymbol{v}_l)$, where $\mu_{k,l} = [\boldsymbol{\mu}_l]_k$ and $v_{k,l} = [\boldsymbol{v}_l]_k$. For this purpose, we take into account that the layer function $\mathbf{f}_l$ transforms the activation $\mathbf{y}_{l-1}$ into the distribution:

$$\widehat{p}(\mathbf{y}_{0:l}) = p(\mathbf{y}_l|\mathbf{y}_{l-1}) q(\mathbf{y}_{0:l-1}) = p(\mathbf{y}_l|\mathbf{y}_{l-1}) \prod_{l_1=0}^{l-1} q(\mathbf{y}_{l_1}),$$

where $p(\mathbf{y}_l|\mathbf{y}_{l-1}) = \delta(\mathbf{y}_l - \mathbf{f}_l(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l))$ is the true posterior at layer l and $q(\mathbf{y}_{0:l-1}) = \prod_{l_1=0}^{l-1} q(\mathbf{y}_{l_1})$ the previous approximating factor. Furthermore, we compute the first and second-order moments of $\widehat{p}(\mathbf{y}_{0:l})$. This will be done in two steps. In the first step, we derive the moments of an activation variable y_k that belongs to all layers excluding the last layer, i.e., y_k is an element of $\mathbf{y}_{0:l-1} = \{\mathbf{y}_0, \ldots, \mathbf{y}_{l-1}\}$, while in the second step, we assume that y_k is an activation variable contained in the last layer $\mathbf{y}_l = \{y_{1,l}, \ldots, y_{N_l,l}\}$. Thus,

1. For $y_k \in \mathbf{y}_{0:l-1}$, we use the relations:

$$\widehat{p}(\mathbf{y}_{0:l}) = \delta(\mathbf{y}_l - \mathbf{f}_l(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l)) q(y_k) q(\mathbf{y}_{\bar{k},0:l-1}),$$

and $\int \delta(x - x_0) \mathrm{d}x = 1$ (yielding $\int (\mathbf{y}_l - \mathbf{f}_l(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l)) \mathrm{d}\mathbf{y}_l = 1$) to obtain:

$$\mathbb{E}_{\widehat{p}}(y_k) = \int \widehat{p}(\mathbf{y}_{0:l}) y_k \mathrm{d}\mathbf{y} = \int q(y_k) y_k \mathrm{d}y_k = \mathbb{E}_{q(y_k)}(y_k), \tag{A31}$$

where $q(\mathbf{y}_{\bar{k},0:l-1}^{(0:l-1)})$ corresponds to the density of all variables excluding y_k, and $\mathrm{d}\mathbf{y} = \prod_{l_1=0}^{l} \mathrm{d}\mathbf{y}_{l_1}$, while:

2. For $y_k \in \mathbf{y}_l$, we use the relations:

$$\widehat{p}(\mathbf{y}_{0:l}) = \delta(\mathbf{y}_{\bar{k},l} - \mathbf{f}_{\bar{k},l}(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l)) \, \delta(y_k - f_{k,l}(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l)) q(\mathbf{y}_{0:l-1}),$$

and $\int x \delta(x - x_0) \mathrm{d}x = x_0$ (yielding $\int \delta(y_k - f_{k,l}(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l)) \mathrm{d}y_k = f_{k,l}(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l)$), to obtain:

$$\begin{aligned} \mathbb{E}_{\widehat{p}}(y_k) &= \int \widehat{p}(\mathbf{y}_{0:l}) y_k \mathrm{d}\mathbf{y} = \int q(\mathbf{y}_{l-1}) f_{k,l}(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l) \mathrm{d}\mathbf{y}_{l-1} \\ &= \mathbb{E}_{q(\mathbf{y}_{l-1})}(f_{k,l}(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l)). \end{aligned} \tag{A32}$$

Replacing y_k with y_k^2 and repeating the above arguments, we find

$$\mathbb{E}_{\widehat{p}}(y_k^2) = \mathbb{E}_{q(y_k)}(y_k^2), \;\; y_k \in \mathbf{y}_{0:l-1}, \tag{A33}$$

$$\mathbb{E}_{\widehat{p}}(y_k^2) = \mathbb{E}_{q(\mathbf{y}_{l-1})}(f_{k,l}^2(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l)), \;\; y_k \in \mathbf{y}_l. \tag{A34}$$

From Equations (A31) and (A33), we see that for all layers except for the lth layer, the moments remain unchanged after the update. The moments for the lth layer will be computed by means of Equations (A32) and (A34). For a linear activation function, i.e., $f_{k,l}(\mathbf{y}_{l-1}; \boldsymbol{\omega}_l) = \sum_{i=1}^{N_{l-1}} w_{ki,l-1} y_{i,l-1} + b_{k,l}$, we find that the expressions of $\mu_{k,l}$ and $v_{k,l}$ are given by Equations (78) and (79), respectively, while for a ReLU activation function $\mathrm{ReLU}(x) = \max(0, x)$, these are given by Equations (80) and (81), respectively.

References

1. Tibshirani, R. A Comparison of Some Error Estimates for Neural Network Models. *Neural Comput.* **1996**, *8*, 152–163, doi:10.1162/neco.1996.8.1.152.
2. Loquercio, A.; Segu, M.; Scaramuzza, D. A General Framework for Uncertainty Estimation in Deep Learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3153–3160, doi:10.1109/lra.2020.2974682.
3. Oh, S.; Byun, J. Bayesian Uncertainty Estimation for Deep Learning Inversion of Electromagnetic Data. *IEEE Geosci. Remote Sens. Lett.* **2021**, 1–5, doi:10.1109/lgrs.2021.3072123.
4. Chevallier, F.; Chéruy, F.; Scott, N.A.; Chédin, A. A Neural Network Approach for a Fast and Accurate Computation of a Longwave Radiative Budget. *J. Appl. Meteorol.* **1998**, *37*, 1385–1397, doi:10.1175/1520-0450(1998)037<1385:annafa>2.0.co;2.
5. Chevallier, F.; Morcrette, J.J.; Chéruy, F.; Scott, N.A. Use of a neural-network-based long-wave radiative-transfer scheme in the ECMWF atmospheric model. *Q. J. R. Meteorol. Soc.* **2000**, *126*, 761–776, doi:10.1002/qj.49712656318.
6. Cornford, D.; Nabney, I.T.; Ramage, G. Improved neural network scatterometer forward models. *J. Geophys. Res. Ocean.* **2001**, *106*, 22331–22338, doi:10.1029/2000jc000417.
7. Krasnopolsky, V.M. *The Application of Neural Networks in the Earth System Sciences*; Springer: Dordrecht, The Netherlands, 2013, doi:10.1007/978-94-007-6073-8.
8. Efremenko, D.S. Discrete Ordinate Radiative Transfer Model With the Neural Network Based Eigenvalue Solver: Proof Of Concept. *Light Eng.* **2021**, *1*, 56–62, doi:10.33383/2020-075.
9. Fan, Y.; Li, W.; Gatebe, C.K.; Jamet, C.; Zibordi, G.; Schroeder, T.; Stamnes, K. Atmospheric correction over coastal waters using multilayer neural networks. *Remote Sens. Environ.* **2017**, *199*, 218–240, doi:10.1016/j.rse.2017.07.016.
10. Fan, C.; Fu, G.; Noia, A.D.; Smit, M.; Rietjens, J.H.; Ferrare, R.A.; Burton, S.; Li, Z.; Hasekamp, O.P. Use of A Neural Network-Based Ocean Body Radiative Transfer Model for Aerosol Retrievals from Multi-Angle Polarimetric Measurements. *Remote Sens.* **2019**, *11*, 2877, doi:10.3390/rs11232877.
11. Gao, M.; Franz, B.A.; Knobelspiesse, K.; Zhai, P.W.; Martins, V.; Burton, S.; Cairns, B.; Ferrare, R.; Gales, J.; Hasekamp, O.; et al. Efficient multi-angle polarimetric inversion of aerosols and ocean color powered by a deep neural network forward model. *Atmos. Meas. Tech.* **2021**, *14*, 4083–4110, doi:10.5194/amt-14-4083-2021.
12. Shi, C.; Hashimoto, M.; Shiomi, K.; Nakajima, T. Development of an Algorithm to Retrieve Aerosol Optical Properties Over Water Using an Artificial Neural Network Radiative Transfer Scheme: First Result From GOSAT-2/CAI-2. *IEEE Trans. Geosci. Remote Sens.* **2021**, *59*, 9861–9872, doi:10.1109/tgrs.2020.3038892.
13. Jiménez, C.; Eriksson, P.; Murtagh, D. Inversion of Odin limb sounding submillimeter observations by a neural network technique. *Radio Sci.* **2003**, *38*, 27-1–27-8, doi:10.1029/2002rs002644.
14. Holl, G.; Eliasson, S.; Mendrok, J.; Buehler, S.A. SPARE-ICE: Synergistic ice water path from passive operational sensors. *J. Geophys. Res. Atmos.* **2014**, *119*, 1504–1523, doi:10.1002/2013jd020759.
15. Strandgren, J.; Bugliaro, L.; Sehnke, F.; Schröder, L. Cirrus cloud retrieval with MSG/SEVIRI using artificial neural networks. *Atmos. Meas. Tech.* **2017**, *10*, 3547–3573, doi:10.5194/amt-10-3547-2017.
16. Efremenko, D.S.; Loyola R, D.G.; Hedelt, P.; Spurr, R.J.D. Volcanic SO_2 plume height retrieval from UV sensors using a full-physics inverse learning machine algorithm. *Int. J. Remote Sens.* **2017**, *38*, 1–27, doi:10.1080/01431161.2017.1348644.
17. Wang, D.; Prigent, C.; Aires, F.; Jimenez, C. A Statistical Retrieval of Cloud Parameters for the Millimeter Wave Ice Cloud Imager on Board MetOp-SG. *IEEE Access* **2017**, *5*, 4057–4076, doi:10.1109/access.2016.2625742.
18. Brath, M.; Fox, S.; Eriksson, P.; Harlow, R.C.; Burgdorf, M.; Buehler, S.A. Retrieval of an ice water path over the ocean from ISMAR and MARSS millimeter and submillimeter brightness temperatures. *Atmos. Meas. Tech.* **2018**, *11*, 611–632, doi:10.5194/amt-11-611-2018.
19. Håkansson, N.; Adok, C.; Thoss, A.; Scheirer, R.; Hörnquist, S. Neural network cloud top pressure and height for MODIS. *Atmos. Meas. Tech.* **2018**, *11*, 3177–3196, doi:10.5194/amt-11-3177-2018.
20. Hedelt, P.; Efremenko, D.S.; Loyola, D.G.; Spurr, R.; Clarisse, L. Sulfur dioxide layer height retrieval from Sentinel-5 Precursor/TROPOMI using FP_ILM. *Atmos. Meas. Tech.* **2019**, *12*, 5503–5517, doi:10.5194/amt-12-5503-2019.

21. Noia, A.D.; Hasekamp, O.P.; van Harten, G.; Rietjens, J.H.H.; Smit, J.M.; Snik, F.; Henzing, J.S.; de Boer, J.; Keller, C.U.; Volten, H. Use of neural networks in ground-based aerosol retrievals from multi-angle spectropolarimetric observations. *Atmos. Meas. Tech.* **2015**, *8*, 281–299, doi:10.5194/amt-8-281-2015.
22. Noia, A.D.; Hasekamp, O.P.; Wu, L.; van Diedenhoven, B.; Cairns, B.; Yorks, J.E. Combined neural network/Phillips–Tikhonov approach to aerosol retrievals over land from the NASA Research Scanning Polarimeter. *Atmos. Meas. Tech.* **2017**, *10*, 4235–4252, doi:10.5194/amt-10-4235-2017.
23. Aires, F. Neural network uncertainty assessment using Bayesian statistics with application to remote sensing: 1. Network weights. *J. Geophys. Res.* **2004**, *109*, doi:10.1029/2003jd004173.
24. Aires, F. Neural network uncertainty assessment using Bayesian statistics with application to remote sensing: 2. Output errors. *J. Geophys. Res.* **2004**, *109*, doi:10.1029/2003jd004174.
25. Pfreundschuh, S.; Eriksson, P.; Duncan, D.; Rydberg, B.; Håkansson, N.; Thoss, A. A neural network approach to estimating a posteriori distributions of Bayesian retrieval problems. *Atmos. Meas. Tech.* **2018**, *11*, 4627–4643, doi:10.5194/amt-11-4627-2018.
26. Arnold, V. On the representation of functions of several variables as a superposition of functions of a smaller number of variables. *Math. Teach. Appl. Hist. Matem. Prosv. Ser. 2* **1958**, *3*, 41–61.
27. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.* **1989**, *2*, 303–314, doi:10.1007/bf02551274.
28. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536, doi:10.1038/323533a0.
29. Nix, D.; Weigend, A. Estimating the mean and variance of the target probability distribution. In Proceedings of the 1994 IEEE International Conference on Neural Networks (ICNN–94), Orlando, FL, USA, 28 June–2 July 1994, doi:10.1109/icnn.1994.374138.
30. Wright, W.; Ramage, G.; Cornford, D.; Nabney, I. Neural Network Modelling with Input Uncertainty: Theory and Application. *J. VLSI Signal Process.* **2000**, *26*, 169–188, doi:10.1023/a:1008111920791.
31. LeCun, Y.; Denker, J.; Solla, S. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*; Touretzky, D., Ed.; Morgan-Kaufmann: Burlington, MA, USA, 1990; Volume 2, pp. 598–605.
32. MacKay, D.J.C. A Practical Bayesian Framework for Backpropagation Networks. *Neural Comput.* **1992**, *4*, 448–472, doi:10.1162/neco.1992.4.3.448.
33. Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; Wierstra, D. Weight Uncertainty in Neural Networks. In Proceedings of the 32nd International Conference on International Conference on Machine Learning, ICML'15, Lille, France. July 6-11. 2015; pp. 1613–1622.
34. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. *arXiv* **2014**, arXiv:stat.ML/1312.6114.
35. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv* **2012**, arXiv:cs.NE/1207.0580.
36. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
37. Gal, Y.; Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *arXiv* **2016**, arXiv:stat.ML/1506.02142.
38. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv* **2015**, arXiv:cs.LG/1502.03167.
39. Teye, M.; Azizpour, H.; Smith, K. Bayesian Uncertainty Estimation for Batch Normalized Deep Networks. *arXiv* **2018**, arXiv:stat.ML/1802.06455.
40. Efremenko, D.S.; Molina García, V.; Gimeno García, S.; Doicu, A. A review of the matrix-exponential formalism in radiative transfer. *J. Quant. Spectrosc. Radiat. Transf.* **2017**, *196*, 17–45, doi:10.1016/j.jqsrt.2017.02.015.
41. Efremenko, D.; Kokhanovsky, A. *Foundations of Atmospheric Remote Sensing*; Springer, Cham: 2021, doi:10.1007/978-3-030-66745-0.
42. Efremenko, D.; Doicu, A.; Loyola, D.; Trautmann, T. Acceleration techniques for the discrete ordinate method. *J. Quant. Spectrosc. Radiat. Transf.* **2013**, *114*, 73–81, doi:10.1016/j.jqsrt.2012.08.014.
43. Goody, R.; West, R.; Chen, L.; Crisp, D. The correlated-k method for radiation calculations in nonhomogeneous atmospheres. *J. Quant. Spectrosc. Radiat. Transf.* **1989**, *42*, 539–550, doi:10.1016/0022-4073(89)90044-7.
44. Efremenko, D.; Doicu, A.; Loyola, D.; Trautmann, T. Optical property dimensionality reduction techniques for accelerated radiative transfer performance: Application to remote sensing total ozone retrievals. *J. Quant. Spectrosc. Radiat. Transf.* **2014**, *133*, 128–135, doi:10.1016/j.jqsrt.2013.07.023.
45. Molina García, V.; Sasi, S.; Efremenko, D.S.; Doicu, A.; Loyola, D. Radiative transfer models for retrieval of cloud parameters from EPIC/DSCOVR measurements. *J. Quant. Spectrosc. Radiat. Transf.* **2018**, *213*, 228–240, doi:10.1016/j.jqsrt.2018.03.014.
46. del Águila, A.; Efremenko, D.S.; Trautmann, T. A Review of Dimensionality Reduction Techniques for Processing Hyper-Spectral Optical Signal. *Light Eng.* **2019**, *27*, 85–98, doi:10.33383/2019-017.
47. Schreier, F.; Gimeno García, S.; Hedelt, P.; Hess, M.; Mendrok, J.; Vasquez, M.; Xu, J. GARLIC — A general purpose atmospheric radiative transfer line-by-line infrared-microwave code: Implementation and evaluation. *J. Quant. Spectrosc. Radiat. Transf.* **2014**, *137*, 29–50, doi:10.1016/j.jqsrt.2013.11.018.
48. Schreier, F. Optimized implementations of rational approximations for the Voigt and complex error function. *J. Quant. Spectrosc. Radiat. Transf.* **2011**, *112*, 1010–1025, doi:10.1016/j.jqsrt.2010.12.010.

49. Gordon, I.; Rothman, L.; Hill, C.; Kochanov, R.; Tan, Y.; Bernath, P.; Birk, M.; Boudon, V.; Campargue, A.; Chance, K.; et al. The HITRAN2016 molecular spectroscopic database. *J. Quant. Spectrosc. Radiat. Transf.* **2017**, *203*, 3–69, doi:10.1016/j.jqsrt.2017.06.038.
50. Bodhaine, B.A.; Wood, N.B.; Dutton, E.G.; Slusser, J.R. On Rayleigh Optical Depth Calculations. *J. Atmos. Ocean. Technol.* **1999**, *16*, 1854–1861, doi:10.1175/1520-0426(1999)016<1854:orodc>2.0.co;2.
51. Anderson, G.; Clough, S.; Kneizys, F.; Chetwynd, J.; Shettle, E. *AFGL Atmospheric Constituent Profiles (0-120 km). AFGL-TR-86-0110*; Air Force Geophysics Laboratory: Hanscom Air Force Base, MA, USA, 1986.
52. Loyola R, D.G.; Pedergnana, M.; García, S.G. Smart sampling and incremental function learning for very large high dimensional data. *Neural Netw.* **2016**, *78*, 75–87, doi:10.1016/j.neunet.2015.09.001.
53. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:cs.LG/1412.6980.
54. Rodgers, C. *Inverse Methods for Atmospheric Sounding: Theory and Practice*; Wolrd Scientific Publishing, Singapore: 2000.
55. Doicu, A.; Trautmann, T.; Schreier, F. *Numerical Regularization for Atmospheric Inverse Problems*; Springer, Berlin: 2010.
56. Tresp, V.; Ahmad, S.; Neuneier, R. Training Neural Networks with Deficient Data. In *Advances in Neural Information Processing Systems*; Cowan, J., Tesauro, G., Alspector, J., Eds.; Morgan-Kaufmann: Burlington, MA, USA, 1994; Volume 6, pp. 128–135.
57. Kendall, A.; Gal, Y. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *arXiv* **2017**, arXiv:cs.CV/1703.04977.
58. Gast, J.; Roth, S. Lightweight Probabilistic Deep Networks. *arXiv* **2018**, arXiv:cs.CV/1805.11327.
59. Jaulin, L.; Kieffer, M.; Didrit, O.; Walter, É. *Applied Interval Analysis*; Springer: London, UK, 2001, doi:10.1007/978-1-4471-0249-6.
60. Kearfott, R.B.; Dawande, M.; Du, K.; Hu, C. Algorithm 737: INTLIB—a portable Fortran 77 interval standard-function library. *ACM Transactions on Mathematical Software* **1994**, *20*, 447–459. doi:10.1145/198429.198433.