Trustworthy Bayesian Perceptrons

Markus Walker[*†], **Hayk Amirkhanian**[*§], **Marco F. Huber**[‡§], and **Uwe D. Hanebeck**[†]
[†]Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Institute for Anthropomatics and Robotics
Karlsruhe Institute of Technology (KIT), Germany
markus.walker@kit.edu, uwe.hanebeck@kit.edu

[‡]Institute of Industrial Manufacturing and Management IFF
University of Stuttgart, Germany

[§]Department Cyber Cognitive Intelligence (CCI)
Fraunhofer Institute for Manufacturing Engineering and Automation IPA, Stuttgart, Germany
hayk.amirkhanian.namagerdi@ipa.fraunhofer.de, marco.huber@ieee.org

***Abstract*—Bayesian Neural Networks (BNNs) offer a sophisticated framework for extending classical neural network point estimates to encompass predictive distributions. Despite the high potential of BNNs, established BNN training methods such as Variational Inference (VI) and Markov Chain Monte Carlo (MCMC) grapple with issues such as scalability and hyperparameter dependence. In addressing these issues, our research focuses on the fundamental elements of BNNs, in particular perceptrons and their predictive capabilities. We introduce a new perspective on the closed-form solution for backward-pass computation for the Bayesian perceptron and prove that the state-of-the-art solution is equivalent to statistical linearization. To assess the efficacy of Bayesian perceptrons and provide insights into their performance in distinct input space regions, a novel methodology utilizing *k*-d trees as a space partitioning method is introduced to evaluate prediction quality within specific input space regions.**

***Index Terms*—Bayesian Neural Networks, uncertainty quantification, statistical linearization, trust regions, calibration, statistical testing.**

I. Introduction

Bayesian Neural Networks (BNNs) are widely used for uncertainty quantification in various fields, including reinforcement learning [1] and model predictive control [2] in robotics. They extend classical neural networks by providing predictive distributions, which have demonstrated promising predictive capabilities.

Although BNNs provide an elegant solution to uncertainty quantification, they have two main issues, as shown in Fig. 1. Firstly, training BNNs as a complex black box model with high predictive capabilities lacks an exact solution, just as with traditional neural networks, making it necessary to use approximate inference methods such as Markov Chain Monte Carlo (MCMC) [3], Variational Inference (VI) [4], or Kalman filtering techniques [5]. As a consequence of approximate inference in training and prediction, BNNs are error-prone and therefore, predictions need to be checked. This leads to the second issue, namely testing, which is about assessing the quality of the predictions. Unfortunately, testing also poses challenges because the true data-generating process is generally hidden, and only a finite amount of realizations is available as test data. However, there are test measures and strategies

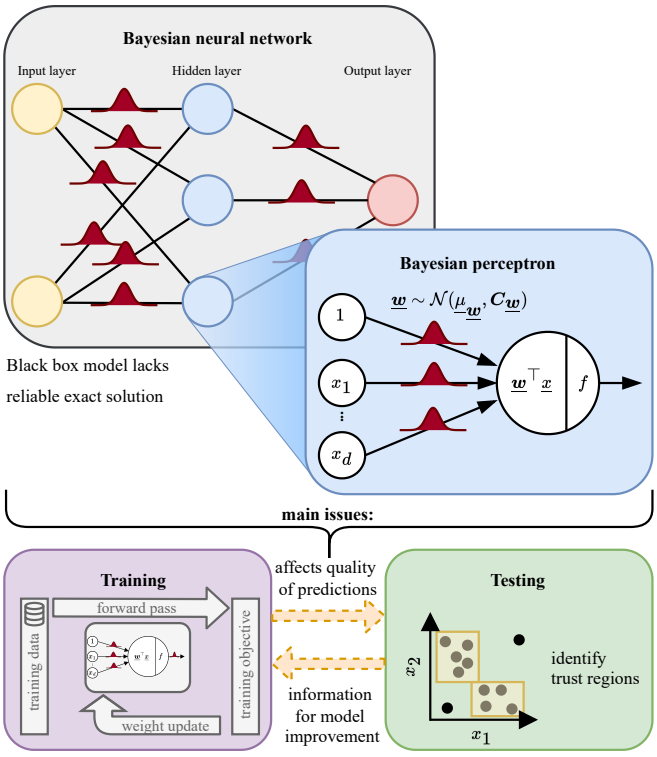


Fig. 1. Illustration of the two main issues associated with BNNs, namely training and testing, and their relationship. The main objective is to gain insights into scalable solutions and their predictive capabilities. This paper examines these issues on a small scale for perceptrons.

This work was supported by the German Federal Ministry of Education and Research under grant RobustNets (FKZ 01IS17042), the Baden-Württemberg Ministry of Economic Affairs, Labor, and Tourism within the KI-Fortschrittszentrum "Lernende Systeme und Kognitive Robotik" (036-140100), and the Fraunhofer-Gesellschaft within the project ML4Safety. This work is also part of the German Research Foundation (DFG) AI Research Unit 5339 regarding the combination of physics-based simulation with AI-based methodologies for the fast maturation of manufacturing processes.

*These authors contributed equally.

that provide valuable information about model quality, such as mean squared error, negative log-likelihood, or uncertainty calibration error (UCE) [6], and their results can be used to improve the model through techniques such as active learning or human involvement, which in turn addresses the training problem. It is therefore of interest to address both of these issues in order to gain insight into scalable solutions and their predictive capabilities, which will enable them to be used as trustworthy systems.

For a deeper understanding, this leads our attention to the core building block of BNNs, namely the perceptron, to look at training and testing at a small scale specifically. In the literature on BNNs, when discussing training of perceptrons, [7] proposed a closed-form solution for the Bayesian perceptron that assumes normally distributed weights. However, due to sources of error such as model assumptions, approximate inference, and a finite number of training data points, there are differences in the quality of predictions depending on the input values [8]. To address this issue, [8] proposed a twofold testing strategy that first identifies candidate regions in the input space where test data is available, and then assesses their quality per candidate region using statistical tests. However, this approach is limited to single-input, single-output systems, meaning that only one-dimensional input spaces can be identified and tested. But generally, BNNs and perceptrons are not limited to one-dimensional inputs. Thus, we expand the identification of candidate regions from [8] to multi-input systems, such as perceptrons with vector-valued inputs.

Contribution: This paper presents a comprehensive analysis of the basic building block of BNNs, the perceptron, in terms of its training and predictive capabilities. In addition, we prove the equivalence of the weight update steps of the Bayesian perceptron and statistical linearization. Furthermore, we propose a novel methodology that uses k-d trees as a spatial partitioning method of the input space of BNNs to identify candidate regions enabling the evaluation of the quality of predictions within certain regions of the input space, which can be assessed with arbitrary test metrics or statistical tests. We demonstrate the predictive performance of the Bayesian perceptron for nonlinear regression and binary classification examples and further show that our proposed candidate region identification and testing method can pinpoint regions in the input space with trustworthy predictions.

Notation: Throughout this paper, vectors will be indicated by underlined letters, e.g., $\underline{x}$, boldface letters, for instance, $\underline{\boldsymbol{x}}$, will represent random variables, and boldface capital letters, e.g., $\mathbf{A}$, will indicate matrices.

II. Related Work

A. Approximate Inference

The gold standard for training BNNs undoubtedly are MCMC-based approaches, which operate by approximating the posterior distribution through sampling methods. Since the initial form of MCMC, i.e., the Metropolis–Hastings algorithm [9], is notably slow due to the generation of numerous samples, several extensions have been introduced. These include Gibbs sampling [10], hybrid Monte Carlo [11], Hamiltonian Monte Carlo [12], and the No-U-Turn Sampler [13].

Variational Inference (VI) [14] offers an optimization-based alternative for training an BNNs, using a surrogate function to approximate the true weight posterior and simplifying the learning problem into a tractable optimization task. In BNN training, VI approximates the weight posterior by a simpler distribution, i.e., the variational distribution, which typically is a normal distribution, using gradient descent to minimize the empirical lower bound to the reverse Kullback–Leibler divergence. Various implementations of VI have been introduced, such as Stochastic Variational Inference [15] or [16] that showed that dropout techniques provide a computationally efficient way of approximating the variational distribution.

Expectation Propagation (EP) [17] minimizes the forward Kullback–Leibler divergence, deviating from the conventional reverse Kullback–Leibler divergence optimization utilized in VI. This departure from VI yields no assured convergence for EP methods. However, convergence can be established by using so-called damped versions of EP, such as double-loop EP [18] or damped EP [19]. Notably, a prominent instantiation of EP designed for BNNs, termed probabilistic backpropagation [20], has gained renowned interest in this field. This technique introduces hyperprior distributions for the weights and is based on a moment-matching paradigm during the forward pass. For the subsequent backward pass, it computes gradients of the marginal likelihood concerning the parameters of the posterior approximation.

The Bayesian perceptron, as presented in [7], serves as the foundational concept upon which Kalman Bayesian Neural Networks (KBNN) [21] later evolved. In this framework, both the weights and predictions of the perceptron are assumed to be normally distributed random variables. Analytical formulations for forecasting the perceptron's output and for weight learning are provided, accommodating widely used activation functions such as sigmoid or ReLU. This methodology obviates the requirement for computationally intensive gradient calculations and allows sequential learning.

Tractable Approximate Gaussian Inference (TAGI) [22] and KBNN exhibit notable similarities, primarily rooted in their reliance on sequential Bayesian filtering across each layer. However, they differ in fundamental aspects: TAGI operates under the mean-field assumption, positing full independence among weights, while KBNN allows full covariance matrices within each layer. Additionally, TAGI utilizes the moment-generating function and linearization, whereas KBNN relies on moment matching.

B. Calibration Measures

The evaluation of the quality of the predictions includes the assessment of the accuracy of the prediction distributions in the representation of the actual data generation process, i.e., the measure of calibration. However, this is a difficult problem due to the lack of ground truth uncertainty estimates. There are several ways to evaluate the predictions of machine learning models, for example, calibration plots can be used for both

classification [23] and regression [24] to visually compare the expected and observed confidence levels across all test data.

In classification tasks, the expected calibration error (ECE) [25] is widely used to assess calibration, measuring the difference between the predicted confidence of the model and its accuracy, using binned test data. The ECE is the sum of errors over all bins and is given by

$$ECE = \sum_{l=1}^{L} \frac{|B_l|}{M} |\mathrm{acc}(B_l) - \mathrm{conf}(B_l)| \ , \tag{1}$$

where $|B_l|$ is the size of each bin, $\frac{|B_l|}{M}$ is the empirical probability of test data falling into the l-th bin, $\mathrm{acc}(B_l)$ is the rate of correct classifications overall classifications per bin and $\mathrm{conf}(B_l)$ is the average predicted probability score per bin, e.g., the averaged output of the sigmoid function in a logistic regression setup.

In regression, the quality of uncertainty estimates is usually measured by scoring rules. For normally distributed univariate predictions, calibration measures such as the UCE [6] and the expected normalized calibration error [26] compare predicted variances with the mean squared error, utilizing their relationship. To measure calibration for arbitrary dimensional normally distributed predictions [27] proposed the quantile calibration error, which compares the observed frequencies and the desired quantile values of the chi-squared distributed errors.

C. Trust Region Identification

Calibration measures typically provide a single score for all data. To extend these single scores to trust regions for Bayesian models, i.e., which input space regions lead to calibrated and trustworthy predictions, [8] proposed a general twofold testing strategy:

1) The initial step involves identifying candidate regions where information is present, that is, regions where test data are available.
2) Based on the identified candidate regions, statistical tests are used to measure the calibration and assess uncertainties within these areas using test data. This step represents the investigation of the local calibration.

Furthermore, [8] proposed a variant for their general strategy and implemented it for the special case of single-input, single-output systems, employing a second model as a reference model in the initial step of identifying candidate regions. It is assumed that the predictions of the approximate model and the reference model will differ for the same test input value if not enough information is available during training. The 1-Wasserstein distance is used to measure the differences between predictions of the two models. If the distance exceeds a certain threshold, the one-dimensional input space is split, resulting in candidate regions represented by intervals.

The second step of the general testing strategy in their proposed version is implemented using statistical tests, such as the binomial test or the averaged normalized estimation error squared (ANEES) test. Therefore, a binary decision is made for each candidate region. In the case of significant deviations between the data and the predictions, the candidate region is rejected by statistical tests and therefore declared as untrustworthy. The binomial test enables the testing of specific confidence intervals and makes no assumptions about the distribution of predictions and data, and is therefore categorized as a nonparametric test. E.g., it can be utilised to verify if the $95\,\%$ confidence interval of the predictions truly encompasses $95\,\%$ of the output test data. For normally distributed predictions, the parametric ANEES test [28] is used to test whether the data are consistent with the predicted distribution. Its test statistic is given by

$$T_{\mathrm{ANEES}} = \frac{1}{S} \sum_{s=1}^{S} \frac{(y_s - \mu_{\boldsymbol{y},s})^2}{\sigma^2_{\boldsymbol{y},s}} \ , \tag{2}$$

where S is the number of considered squared Mahalanobis distances, also referred to as normalized estimation error squared, i.e., the number of test points within a candidate region, $\mu_{\boldsymbol{y},s}$ is the predicted mean, $\sigma^2_{\boldsymbol{y},s}$ is the predicted variance and y_s is an output data point. For normally distributed predictions, the average of the squared Mahalanobis distances results in a chi-squared distributed test statistic whose critical values are given by

$$[c_\mathrm{l}, c_\mathrm{u}] = \frac{1}{S} \left[F^{-1}_{\chi^2_k}\left(\frac{\alpha}{2}\right), \ F^{-1}_{\chi^2_k}\left(1 - \frac{\alpha}{2}\right) \right] \ , \tag{3}$$

where $F^{-1}_{\chi^2_k}$ is the chi-square inverse cumulative distribution function with $k = S$ degrees of freedom and α is the significance level. If T_{ANEES} is less than the lower critical value c_l, the uncertainty of the estimated normal distribution is greater than the uncertainty reflected by the data, i.e., the uncertainty is overestimated. If T_{ANEES} exceeds the upper critical value c_u, there may be a significant bias or an underestimation of uncertainty.

III. Learning Setup

We consider the perceptron in a supervised learning setup with a training data set $\mathcal{D} = \{(\underline{x}_n, y_n)\}_{n=1}^N$ comprising N independent and identically distributed (i.i.d.) pairs consisting of the d-dimensional inputs $\underline{x}_n \in \mathbb{R}^d$ and the one-dimensional outputs $y_n \in \mathbb{R}$. The perceptron is defined by

$$\begin{aligned} y &= f(a) \ , \\ a &= \underline{w}^\top \cdot \underline{x} + w_0 \ , \end{aligned} \tag{4}$$

with a nonlinear transformation $f(.)$, also known as the activation function, from the activation a to the output y, where the activation a is a weighted sum of input $\underline{x}$ and the bias w_0. For simpler notation, the bias w_0 can be included in the weight vector $\underline{w} = \begin{bmatrix} w_0 & w_1 & w_2 & \dots & w_d \end{bmatrix}^\top$ by expanding the input vector by a constant 1, which redefines the input vector as $\underline{x} = \begin{bmatrix} 1 & x_1 & x_2 & \dots & x_d \end{bmatrix}^\top$ and therefore (4) to $a = \underline{w}^\top \cdot \underline{x}$.

In the probabilistic perspective, we utilize the notation $\boldsymbol{y} = f\left(\underline{\boldsymbol{w}}^\top \underline{x}\right)$ to expand the deterministic perceptron to a Bayesian perceptron, where all weights are represented by $\underline{\boldsymbol{w}}$ and are considered as random variables with a prior distribution of

$p(\underline{w})$. In training, the objective is to learn the weight posterior distribution $p(\underline{w} \mid \mathcal{D})$, which is obtained by

$$p(\underline{w} \mid \mathcal{D}) = \frac{p(\mathcal{Y} \mid \mathcal{X}, \underline{w})\, p(\underline{w})}{p(\mathcal{Y} \mid \mathcal{X})} \ ,$$

where $p(\mathcal{Y} \mid \mathcal{X}, \underline{w})$ is the likelihood, $p(\mathcal{Y} \mid \mathcal{X})$ is a normalization constant, $\mathcal{X} = \{\underline{x}_1, \ldots, \underline{x}_N\}$ and $\mathcal{Y} = \{y_1, \ldots, y_N\}$ are the input and output data from the training data set $\mathcal{D}$, respectively. The predictive distribution can be obtained by

$$p(y \mid \underline{x}, \mathcal{D}) = \int_{\Omega_{\underline{\boldsymbol{w}}}} p(y \mid \underline{x}, \underline{w})\, p(\underline{w} \mid \mathcal{D})\, \mathrm{d}\underline{w} \ ,$$

for the given input $\underline{x}$, using the learned posterior distribution $p(\underline{w} \mid \mathcal{D})$. However, in general, there is no exact solution of $p(\underline{w} \mid \mathcal{D})$ and $p(y \mid x, \mathcal{D})$, and in practice we have to use approximate inference techniques such as those considered in [7], which we prove in the following to be equivalent to the statistical linearization of a perceptron.

Note that from now on we omit the training data set $\mathcal{D}$ to simplify the notation. Hereafter, the input $\underline{x}$ is taken as given, the predicted output $p(y \mid \underline{x})$ as normally distributed with $\mathcal{N}(\mu_{\boldsymbol{y}}, \sigma_{\boldsymbol{y}}^2)$ and $\underline{\boldsymbol{w}}$ as normally distributed weights $\underline{\boldsymbol{w}} \sim \mathcal{N}(\underline{\mu}_{\underline{\boldsymbol{w}}}, \mathbf{C}_{\underline{\boldsymbol{w}}})$ whose posterior distribution is to be approximated.

IV. Statistical Linearization of a Perceptron

Statistical linearization was introduced by [29] and is derived by minimizing the mean squared error between the nonlinear function and its linearized version. The calculation of the output can be split up into two steps, first multiplying $\underline{\boldsymbol{w}}$ and $\underline{x}$, followed by the application of the activation function. Both steps use the statistical linearization given by

$$\underline{\boldsymbol{w}}^\top \underline{x} \approx \underline{v}^\top \begin{bmatrix} \underline{\boldsymbol{w}} \\ \underline{x} \end{bmatrix} + \boldsymbol{b}_{\underline{v}^\top} \text{ and } \underline{v}^\top = \underline{c}_{\underline{\tilde{\boldsymbol{x}}}\boldsymbol{a}}^\top \begin{bmatrix} \mathbf{C}_{\underline{\boldsymbol{w}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{\underline{x}} \end{bmatrix}^{-1} \ ,$$

with $\underline{c}_{\underline{\tilde{\boldsymbol{x}}}\boldsymbol{a}}^\top = \begin{bmatrix} \sigma_{\tilde{\boldsymbol{x}}_1\boldsymbol{a}}^2 & \ldots & \sigma_{\tilde{\boldsymbol{x}}_{\tilde{d}}\boldsymbol{a}}^2 \end{bmatrix}$ being the covariance vector of the augmented $\tilde{d}$-dimensional state vector $\underline{\tilde{\boldsymbol{x}}} = \begin{bmatrix} \underline{\boldsymbol{w}}^\top & \underline{x}^\top \end{bmatrix}^\top$ and $\boldsymbol{a}$. Employing $\boldsymbol{b}_{\underline{v}^\top} \sim \mathcal{N}(\mu_{\boldsymbol{a}} - \underline{v}^\top \mu_{\underline{\tilde{\boldsymbol{x}}}})$ guarantees an unbiased linearization. Since the output is a one-dimensional variable, $\underline{v}^\top$ is a row vector. As $\underline{x}$ is deterministic, $\mathbf{C}_{\underline{x}} = \mathbf{0}$. In theory, this makes the calculation of the inverse impossible. However, as the input does not need to be updated, one can simply assume a zero-block there.

The activation function can be linearized as

$$\begin{aligned} \boldsymbol{y} = f\left(\underline{\boldsymbol{w}}^\top \underline{x}\right) &\approx g \cdot \left(\underline{v}^\top \begin{bmatrix} \underline{\boldsymbol{w}} \\ \underline{x} \end{bmatrix} + \boldsymbol{b}_{\underline{v}^\top} \right) + \boldsymbol{b}_g \\ &\approx \underbrace{g\, \underline{v}^\top}_{=:\underline{h}^\top} \underbrace{\begin{bmatrix} \underline{\boldsymbol{w}} \\ \underline{x} \end{bmatrix}}_{=:\underline{\tilde{\boldsymbol{x}}}} + \underbrace{g\, \boldsymbol{b}_{\underline{v}^\top} + \boldsymbol{b}_g}_{=:\boldsymbol{b}_{\underline{h}^\top}} = \underline{h}^\top \underline{\tilde{\boldsymbol{x}}} + \boldsymbol{b}_{\underline{h}^\top} \ , \end{aligned} \tag{5}$$

with $g = \sigma_{\boldsymbol{a}\boldsymbol{y}}^2 / \sigma_{\boldsymbol{a}}^2$ being a scalar and $\boldsymbol{b}_g \sim \mathcal{N}(\mu_{\boldsymbol{y}} - g\, \mu_{\boldsymbol{a}}, \sigma_{\boldsymbol{y}}^2 - g\, \sigma_{\boldsymbol{a}}^2)$. Note that $\boldsymbol{b}_{\underline{h}^\top} \sim \mathcal{N}(\mu_{\boldsymbol{y}} - \underline{h}^\top \underline{\mu}_{\underline{\tilde{\boldsymbol{x}}}}, \sigma_{\boldsymbol{y}}^2 - \underline{h}^\top \underline{c}_{\underline{\tilde{\boldsymbol{x}}}\boldsymbol{y}})$ is the bias of the linearization and not of the perceptron itself. The proof that this linearization is unbiased and does not change the variance is trivial and will be omitted.

In [30], it was shown how to represent a neural network as a state space model. Through the linearization the perceptron is represented as a linear state space model, which can be used in a Kalman filter for estimation purposes. If one includes the bias $\boldsymbol{b}_{\underline{h}^\top}$ into the row vector $\underline{h}^\top$, then the update step for the augmented state is given by

$$\underline{\mu}_{\underline{\tilde{\boldsymbol{x}}}}^+ = \underline{\mu}_{\underline{\tilde{\boldsymbol{x}}}} + \underline{k}_{\underline{\tilde{\boldsymbol{x}}}} \left(y - \mu_{\boldsymbol{y}}\right) \ , \tag{6}$$

$$\mathbf{C}_{\underline{\tilde{\boldsymbol{x}}}}^+ = \mathbf{C}_{\underline{\tilde{\boldsymbol{x}}}} - \underline{k}_{\underline{\tilde{\boldsymbol{x}}}}\, \underline{h}^\top\, \mathbf{C}_{\underline{\tilde{\boldsymbol{x}}}} \ , \tag{7}$$

with the realization y and Kalman gain

$$\underline{k}_{\underline{\tilde{\boldsymbol{x}}}} = \mathbf{C}_{\underline{\tilde{\boldsymbol{x}}}}\, \underline{h} (\underbrace{\underline{h}^\top\, \mathbf{C}_{\underline{\tilde{\boldsymbol{x}}}}\, \underline{h}}_{=\sigma_{\boldsymbol{y}}^2})^{-1} \ .$$

It does not matter whether one updates $\underline{\boldsymbol{w}}$ or $\begin{bmatrix} \underline{\boldsymbol{w}}^\top & \underline{x}^\top \end{bmatrix}^\top$ since in the latter case one can easily adjust $\underline{k}_{\underline{\boldsymbol{w}}}$ to $\underline{k}_{\underline{\tilde{\boldsymbol{x}}}}$ because in that case, $\mathbf{C}_{\underline{\tilde{\boldsymbol{x}}}}$ becomes

$$\mathbf{C}_{\underline{\tilde{\boldsymbol{x}}}} = \begin{bmatrix} \mathbf{C}_{\underline{\boldsymbol{w}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \ .$$

Hereafter, we update $\underline{\boldsymbol{w}}$ with $\underline{k}_{\underline{\boldsymbol{w}}}$, and consequently, the subsequent $\underline{h}^\top$ has a different dimension than in (5), but for simplicity, different notations are omitted.

In the Bayesian perceptron [7], the weights were updated through a Bayesian approach, resulting in

$$\underline{\mu}_{\underline{\boldsymbol{w}}}^+ = \underline{\mu}_{\underline{\boldsymbol{w}}} + \underline{l} \left(\mu_{\boldsymbol{a}}^+ - \mu_{\boldsymbol{a}}\right) \ , \tag{8}$$

$$\mathbf{C}_{\underline{\boldsymbol{w}}}^+ = \mathbf{C}_{\underline{\boldsymbol{w}}} + \underline{l} \left((\sigma_{\boldsymbol{a}}^+)^2 - \sigma_{\boldsymbol{a}}^2\right) \underline{l}^\top \ , \tag{9}$$

with $\underline{l} = {}^{(\mathbf{C}_{\underline{\boldsymbol{w}}}\, \underline{x})}/_{\sigma_{\boldsymbol{a}}^2}$ and

$$\begin{aligned} \mu_{\boldsymbol{a}}^+ &= \mu_{\boldsymbol{a}} + \tilde{l} \left(y - \mu_{\boldsymbol{y}}\right) \ , \\ (\sigma_{\boldsymbol{a}}^+)^2 &= \sigma_{\boldsymbol{a}}^2 - \tilde{l}\, \sigma_{\boldsymbol{y}\boldsymbol{a}}^2 \end{aligned}$$

with $\tilde{l} = {}^{\sigma_{\boldsymbol{a}\boldsymbol{y}}^2}/_{\sigma_{\boldsymbol{y}}^2}$.

The main difference between the two update steps is that [30] formulates and updates BNNs (or, in this case, a Bayesian Perceptron) as a linear state space model. This means, that there is a single update step for all weights. However, [7] updates the weights sequentially for each layer and within each layer also updates the activation $\boldsymbol{a}$ first. Fortunately, both procedures result in identical update steps as proven by the following theorem.

Theorem: Both Bayesian Perceptron formulations, namely those in (6) and (7), are equivalent to [7], in the sense that they yield identical update steps for both the mean and covariance matrix of the weights.

Proof: The equivalence of both mean weight update steps in (6) and (8) is given by

$$\begin{aligned}\underline{\mu}_{\underline{w}}^{+} &= \underline{\mu}_{\underline{w}} + \underline{l}\,(\mu_{\boldsymbol{a}}^{+} - \mu_{\boldsymbol{a}}) \\ &= \underline{\mu}_{\underline{w}} + \underline{l}\,(\mu_{\boldsymbol{a}} + \tilde{l}\,(y - \mu_{\boldsymbol{y}}) - \mu_{\boldsymbol{a}}) \\ &= \underline{\mu}_{\underline{w}} + \underline{l}\,\tilde{l}\,(y - \mu_{\boldsymbol{y}}) \\ &= \underline{\mu}_{\underline{w}} + \underbrace{\mathbf{C}_{\underline{w}}\,\underline{x}\,(\sigma_{\boldsymbol{a}}^2)^{-1}\sigma_{\boldsymbol{ay}}^2}_{=\underline{h}}(\sigma_{\boldsymbol{y}}^2)^{-1}\,(y - \mu_{\boldsymbol{y}}) \;, \\ &\qquad\qquad \underbrace{\qquad\qquad\qquad\qquad}_{\underline{k}_{\underline{w}}}\end{aligned}$$

with $\tilde{l} = {}^{\sigma_{\boldsymbol{ay}}^2}/_{\sigma_{\boldsymbol{y}}^2)}$ being the Kalman gain of [7]. The last step used the fact, that $\underline{v}^\top$ is equal to $\underline{x}^\top$ in the appropriate dimensions. This can be proven by

$$\begin{aligned}\underline{v}^\top &= \underline{c}_{\tilde{\underline{x}}\,\boldsymbol{a}}^\top \begin{bmatrix} \mathbf{C}_{\underline{w}}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{\underline{x}} \end{bmatrix} = \underline{c}_{\tilde{\underline{x}}\,\boldsymbol{a}}^\top \begin{bmatrix} \mathbf{C}_{\underline{w}}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ \underline{c}_{\tilde{\underline{x}}\,\boldsymbol{a}} &= \begin{bmatrix} \mathbf{C}_{\underline{w}}\mu_{\underline{x}} \\ \mathbf{C}_{\underline{x}}\mu_{\underline{w}} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{\underline{w}}\mu_{\underline{x}} \\ \underline{0} \end{bmatrix} \iff \underline{c}_{\tilde{\underline{x}}\,\boldsymbol{a}}^\top = \begin{bmatrix} \underline{x}^\top\mathbf{C}_{\underline{w}} & \underline{0}^\top \end{bmatrix} \\ \Rightarrow \underline{v}^\top &= \begin{bmatrix} \underline{x}^\top\mathbf{C}_{\underline{w}} & \underline{0}^\top \end{bmatrix} \begin{bmatrix} \mathbf{C}_{\underline{w}}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \underline{x}^\top & \underline{0}^\top \end{bmatrix} \;.\end{aligned}$$

Similarly, the equivalence for updating the covariance matrices in equations (7) and (9) is proven by

$$\begin{aligned}\mathbf{C}_{\underline{w}}^{+} &= \mathbf{C}_{\underline{w}} + \underline{l}\,((\sigma_{\boldsymbol{a}}^2)^{+} - \sigma_{\boldsymbol{a}}^2)\,\underline{l}^\top \\ &= \mathbf{C}_{\underline{w}} + \underline{l}\,(\sigma_{\boldsymbol{a}}^2 - \tilde{l}\,\sigma_{\boldsymbol{ay}}^2 - \sigma_{\boldsymbol{a}}^2)\,\underline{l}^\top \\ &= \mathbf{C}_{\underline{w}} - \underline{l}\,\tilde{l}\,\sigma_{\boldsymbol{ay}}^2\,\underline{l}^\top \\ &= \mathbf{C}_{\underline{w}} - \underbrace{\mathbf{C}_{\underline{w}}\,\underline{x}\,(\sigma_{\boldsymbol{a}}^2)^{-1}\,\sigma_{\boldsymbol{ay}}^2\,(\sigma_{\boldsymbol{y}}^2)^{-1}}_{=\underline{k}_{\underline{w}}}\;\underbrace{\sigma_{\boldsymbol{ya}}^2}_{=\sigma_{\boldsymbol{ay}}^2}\,(\sigma_{\boldsymbol{a}}^2)^{-1}\underline{x}^\top\,\mathbf{C}_{\underline{w}} \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \underbrace{\qquad\qquad}_{=\underline{h}^\top} \\ &= \mathbf{C}_{\underline{w}} - \underline{k}_{\underline{w}}\,\underline{h}^\top\,\mathbf{C}_{\underline{w}} \;.\end{aligned}$$

■

Given the iterative nature of linearization, it is conceivable to extend this process to BNNs comprising multiple layers. Consequently, one can conceptualize a BNN as a linear state space model, thus facilitating the application of weight estimation methodologies that have been extensively investigated within this domain. For further exploration of these techniques, interested readers are referred to the works [31] and [32].

V. Candidate Region Identification

Typically neither single perceptrons nor entire neural networks (composed of perceptrons) in both deterministic and Bayesian settings are restricted to one-dimensional input spaces. Therefore, we extend the two-step testing procedure from [8], to an input space of arbitrary dimension. Our proposed scheme is shown in Fig. 2.

To efficiently handle higher input dimensions, we use k-d trees as a space partitioning data structure. The k-d tree partitions space by hyperplanes along k orthogonal axes using a binary tree structure, resulting in hyperrectangular space partitions [33]. Each non-leaf node divides the space of the tree branch by bisecting the longest side of the hyperrectangular region into two subspaces. The split point is chosen as the median of the input data along the longest side to ensure a balanced tree, i.e., a tree where each leaf node represents a hyperrectangular region r_l with approximately the same number of data points in it. For more details on splitting methods and graphical visualization, see [34]. By specifying a maximum leaf node size $M \in \mathbb{N}_{>0}$, each node will be split during tree construction if the number of points per node exceeds M. In cases where leaf nodes lead to unbounded regions, the hyperrectangles are restricted to the actual data range. Considering the testing context, this is equivalent to avoiding extrapolation, as predictions can only be checked where test

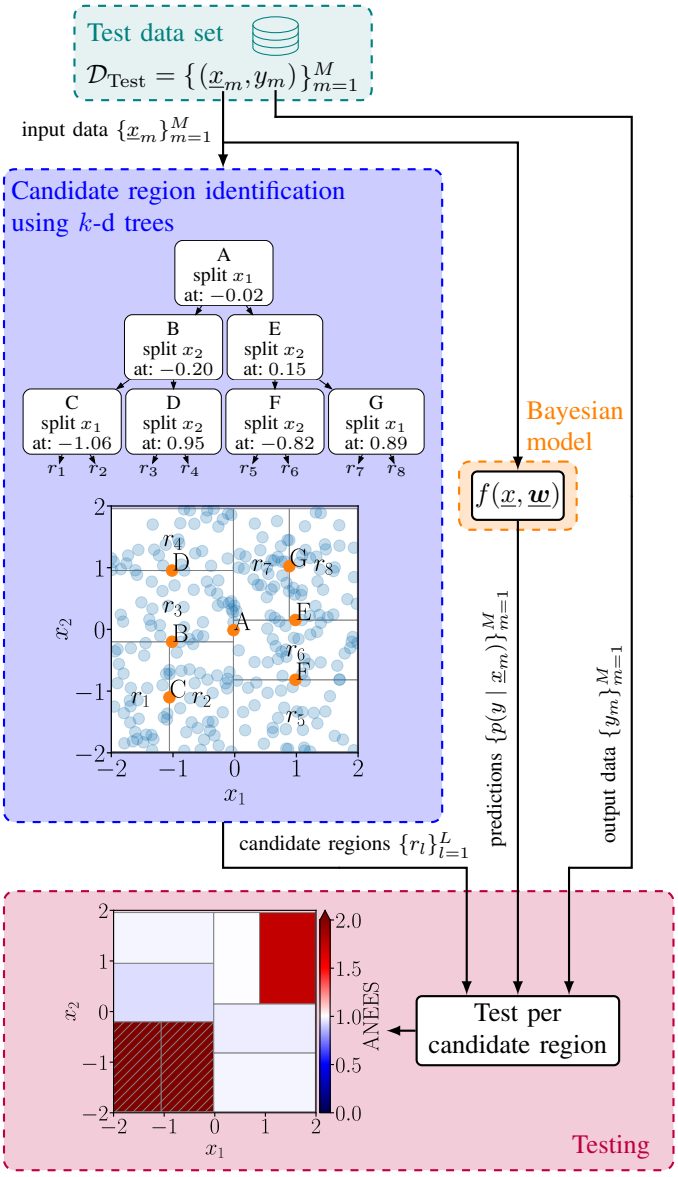
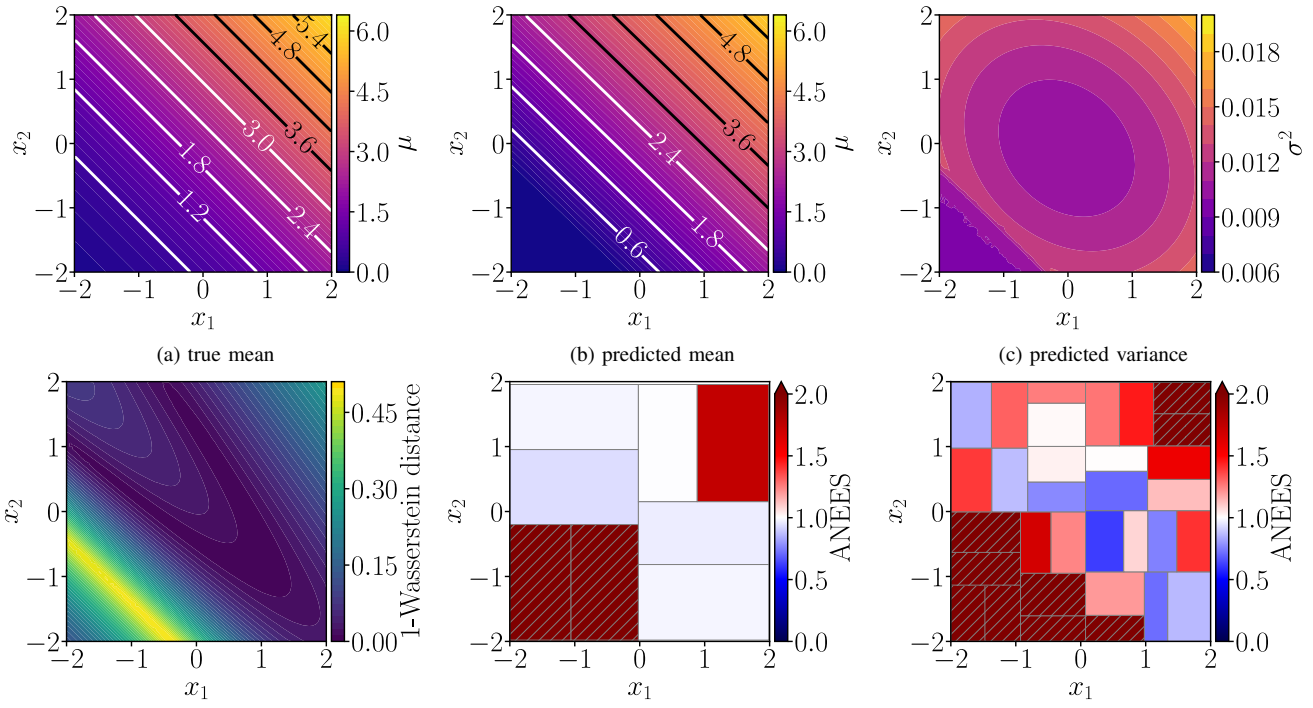


Fig. 2. Illustration of our proposed twofold scheme for testing the quality of predictions from Bayesian models. The first step consists of partitioning the input space into candidate regions using k-d trees, and the second step assesses the quality of the predictions per candidate region using a test metric such as the ANEES.



(a) true mean (b) predicted mean (c) predicted variance

(d) 1-Wasserstein distance (e) 250 test points (f) 1000 test points

Fig. 3. Results of the nonlinear regression using 50 training points. The true mean of the data-generating process is shown in (a). Note that the true variance remains constant at 0.01 and is not displayed. The predicted distributions are shown in (b) and (c). The 1-Wasserstein distances between the true data-generating process are shown in (d). The results of our proposed testing method are shown using (e) 250 and (f) 1000 test points. The shaded areas indicate candidate regions that were rejected and are therefore considered untrustworthy based on the ANEES test statistic and its critical values.

data is available. E.g., region r_1 in Fig. 2 is defined by the lines passing through B and C, and is also limited by the lowest input values along both coordinate axes, i.e., $x_1 \approx -2$ and $x_2 \approx -2$.

The maximum leaf node size M can be adjusted to control the number of test points per candidate region. This is particularly relevant in the context of testing, when a minimum number of test points is required to obtain meaningful results. When the nodes in a tree are divided, and the resulting leaf node size exceeds the maximum allowed size M, the actual leaf node size falls within the range of $M/2 \leq m \leq M$ for balanced trees. Thereby, if one needs at least, say, 20 test data points per candidate region, one should set M to at least 40. Note that unlike [8], no reference model is required to identify candidate regions, saving expensive computation time to train a second model, which is particularly desirable for scalable extension to models more complex than single perceptrons.

VI. Canidate Region Testing

Once the candidate regions are identified, the predictions and test data points within those regions are compared using test metrics or statistical tests.

For regression tasks with normally distributed predictions, we use the ANEES test statistic (2) with its critical values (3), as is also done in [8]. Note that the ANEES test can also be used for $S = 1$, i.e., for candidate regions with only one test point. However, in this case the critical values are given by $[0.05, 7.38]$, so only large deviations between predictions and data can be detected. Therefore, it is recommended to use more test points per candidate region to make stronger test statements with more narrowly defined critical values, and thus the power of the statistical tests can be controlled by the maximum leaf size. However, larger maximum leaf sizes also lead to larger candidate regions, resulting in a trade-off between the resolution in the input space by candidate regions and the meaningfulness of statistical test results. As an analogy, this trade-off is comparable to the uncertainty principle in the short-time Fourier transform, where there is a trade-off between time and frequency resolution [35].

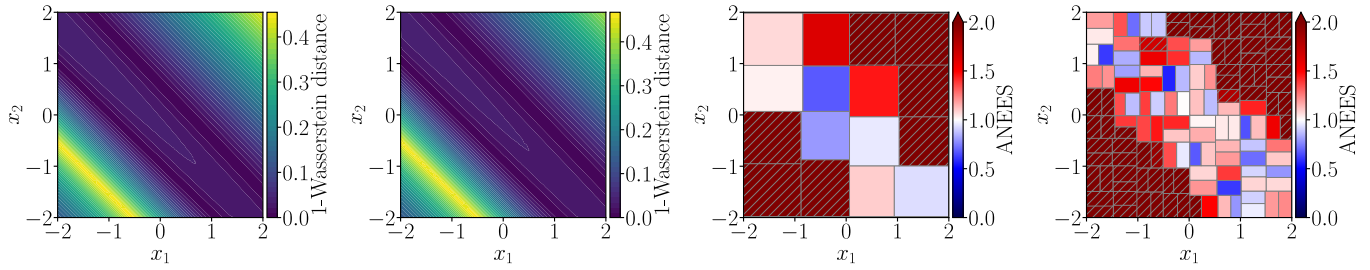
For classification tasks, we use the ECE (1) without binned test data. This is because the data within a candidate region already represents a subset of the test data. Note that candidate regions can be evaluated using arbitrary calibration or test measures.

VII. Evaluation

We now demonstrate the predictive capabilities of Bayesian perceptrons on a nonlinear regression example and a binary classification example and assess the quality of predictions within candidate regions.

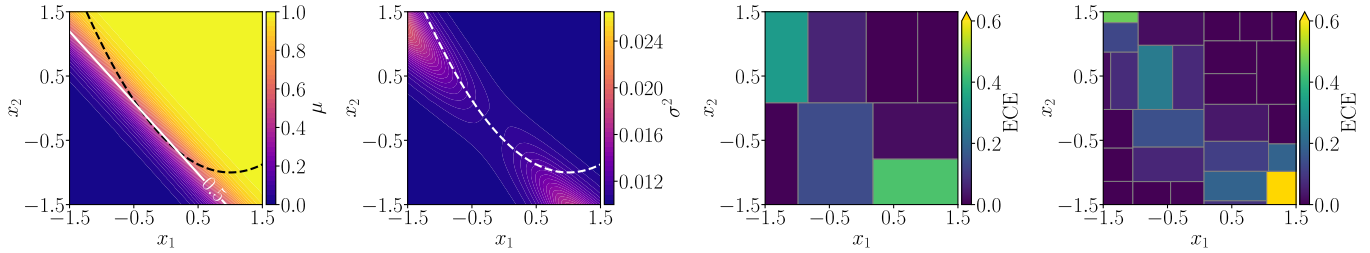
A. Nonlinear Regression

In the first experiment, we generate 50 training points and 250 test points from the noisy soft plus $\boldsymbol{y} = \log(1 + e^{\underline{\gamma}^\top \cdot \underline{x} + \delta}) + \boldsymbol{\epsilon}$, with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 0.01)$, $\underline{\gamma}^\top = \begin{bmatrix} 1 & 1 \end{bmatrix}$, $\delta = 2$ and $\underline{x} \in \mathbb{R}^2$, which can be approximated by a



(a) 500 training points (b) 5000 training points (c) 500 testing points (d) 5000 testing points

Fig. 4. Results of the nonlinear regression show the 1-Wasserstein distances between the true data generating process and the predictions for (a) 500 and (b) 5000 training points. The corresponding tested regions, using 500 and 5000 test points, are shown in (c) and (d), respectively.



(a) predicted mean (b) predicted variance (c) 250 test points (d) 1000 test points

Fig. 5. Binary classification results with predicted mean (a) and variance (b), where the true decision boundary is marked by the dashed line, and tested candidate regions for 250 (c) and 1000 (d) used test instances.

perceptron with a ReLU activation function. The training inputs $\underline{x}_n \in \mathcal{X}_{\text{Train}}$ and the test inputs $\underline{x}_n \in \mathcal{X}_{\text{Test}}$ are drawn uniformly from the two-dimensional input space $[-1, 1] \times [-1, 1]$ and $[-2, 2] \times [-2, 2]$, respectively. The prior $p(\underline{w})$ is chosen as an isotropic normal distribution with the identity matrix $\mathbf{I}_3$ as the covariance matrix, and the prior mean $\underline{\mu}$ is sampled from $\mathcal{N}(\underline{0}, \mathbf{I}_3)$ to avoid a symmetrical weight mean initialization. The results of the predictions, the 1-Wasserstein distance between the true data generating process and the tested regions are shown in Fig. 3, where the candidate regions are calculated using a maximum leaf size of $M = 40$ and ANEES is used as a statistical test with a significance level of $\alpha = 0.01$ to compare the predictions with the test data.

The plots in Figs. 3a to 3c show that the predictions converge to the true data generating process in areas where training data is available, which is confirmed in Fig. 3d by the 1-Wasserstein distance between the predictions and the true process. The same conclusion can be drawn, without the use of information about the data-generating process, from the results of our presented test method using the ANEES test. Here, 250 and 1000 test points are used in Fig. 3e and Fig. 3f, respectively. Note that the input space is resolved more finely when more test data is available. To further investigate the effect of more training and test data, Fig. 4 shows the 1-Wasserstein distance and the tested regions for 1000 and 5000 test and training samples. The experiments reveal that despite the increase in the amount of training data, there are still local differences in the quality of the predictions. However, our proposed test method can identify these differences and reject candidate regions with untrustworthy predictions.

B. Binary Classification

As a second example, we consider a binary classification problem, where the data is again generated over the two-dimensional input space $\underline{x} \in \mathbb{R}^2$. The two classes are assigned according to

$$y = \begin{cases} 1 & , \; g(\tilde{x}_1) < x_2 \\ 0 & , \; \text{otherwise} \end{cases} ,$$

where classes are separated by a quadratic polynomial $g(\tilde{x}_1) = 0.5 \cdot \tilde{x}_1^2 - \tilde{x}_1 - 0.5$ with $\tilde{x}_1 = x_1 + \epsilon$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 0.05)$. Note that ϵ is used to simulate a diffuse boundary between both classes. For training and testing, we generate 30 and 50 random samples uniformly from $[-1, 1] \times [-1, 1]$ and $[-1.5, 1.5] \times [-1.5, 1.5]$, respectively. Again, the prior is initialized with an identity covariance matrix, and the mean is sampled from a standard normal distribution. Fig. 5 shows the predictions and evaluated regions using ECE (1) without dividing test data into bins, and again using a maximum leaf size of $M = 40$.

The comparison of the mean predictions with the true separation in Fig. 5a reveals incorrect classifications in the upper left and lower right corners. Such results are expected since a single perceptron can only represent a linear separation between binary classes, and therefore cannot learn a quadratic polynomial boundary. Fig. 5b illustrates that the variance of the prediction increases in the areas of misclassification. This can be seen as reasonable, as the model itself provides a statement about the increased uncertainty in the prediction in the case of erroneous classification. Again, the input space is resolved more finely by candidate regions when more test data is used. However, candidate regions that lead to misclassification are

recognized by larger error values for both 250 and 1000 test points.

C. Discussion

By using the 1-Wasserstein distance between the predictions and the known ground truth, the results show that statistical linearization leads to good predictions with near-zero distance values. Furthermore, our proposed testing method, which does not use exact ground truth information but uses test data points, is able to first locate appropriate regions and then test data in these regions. The region identification itself is able to divide the input space into regions of a sufficient number of data points. Furthermore, the ANEES test statistic makes it possible to make binary decisions for each region in the input space, whereby regions with poor ANEES values according to the critical values are rejected and thus become untrusted regions.

VIII. Conclusion

This paper has shown that the recursive Bayesian weight estimation of the well-known Bayesian perceptron is equivalent to statistical linearization. Since our proposed statistical linearization can similarly be applied to multilayer neural networks, BNNs can be represented as a linear state space model, thus opening up weight estimation to the powerful existing research in this area. Furthermore, our proposed testing method for multi-input systems, which is not restricted to Bayesian perceptrons, assesses prediction quality by testing in input space regions and is the next step toward the trustworthy application of Bayesian models. Future research directions include statistical linearization of BNNs and testing methods for multi-input, multi-output systems for regression and classification tasks. Furthermore, the information about the trustworthiness of the regions enables the systematic use of active learning to learn specifically in the regions that promise more information gain.

References

[1] X. Wu *et al.*, "Uncertainty-Guided Active Reinforcement Learning with Bayesian Neural Networks," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 5751–5757.

[2] F. Cursi *et al.*, "Bayesian Neural Network Modeling and Hierarchical MPC for a Tendon-Driven Surgical Robot With Uncertainty Minimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2642–2649, Apr. 2021.

[3] N. Metropolis *et al.*, "Equation of State Calculations by Fast Computing Machines," *J. Chem. Phys.*, vol. 21, no. 6, pp. 1087–1092, 1953.

[4] A. Graves, "Practical variational inference for neural networks," in *NIPS'11: Proceedings of the 24th International Conference on Neural Information Processing Systems*, 2011, pp. 2348–2356.

[5] K. Watanabe and S. G. Tzafestas, "Learning algorithms for neural networks with the Kalman filters," *Journal of Intelligent and Robotic Systems*, vol. 3, no. 4, pp. 305–319, 1990.

[6] M.-H. Laves *et al.*, "Well-Calibrated Regression Uncertainty in Medical Imaging with Deep Learning," in *Proceedings of the Third Conference on Medical Imaging with Deep Learning*, vol. 121, 2020, pp. 393–412.

[7] M. F. Huber, "Bayesian Perceptron: Towards fully Bayesian Neural Networks," in *59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 3179–3186.

[8] M. Walker *et al.*, "Identifying Trust Regions of Bayesian Neural Networks," in *2023 IEEE Symposium Sensor Data Fusion and International Conference on Multisensor Fusion and Integration (SDF-MFI)*, Bonn, Germany, Nov. 2023, pp. 1–8.

[9] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, Apr. 1970.

[10] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 721–741, 1984.

[11] S. Duane *et al.*, "Hybrid monte carlo," *Physics letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[12] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[13] M. D. Hoffman, A. Gelman *et al.*, "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1593–1623, 2014.

[14] A. Graves, "Practical variational inference for neural networks," *Advances in neural information processing systems*, vol. 24, 2011.

[15] M. D. Hoffman *et al.*, "Stochastic variational inference," *Journal of Machine Learning Research*, 2013.

[16] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[17] T. P. Minka, "A family of algorithms for approximate Bayesian inference," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[18] T. Heskes and O. Zoeter, "Extended version: Expectation propagation for approximate inference in dynamic bayesian networks," University of Nijmegen, Tech. Rep., 2003.

[19] T. P. Minka, "Power EP," Microsoft Research, Tech. Rep. MSR-TR-2004-149, 2004.

[20] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *International conference on machine learning*. PMLR, 2015, pp. 1861–1869.

[21] P. Wagner, X. Wu, and M. F. Huber, "Kalman Bayesian Neural Networks for Closed-form Online Learning," in *37th AAAI Conference on Artificial Intelligence*, 2023.

[22] J.-A. Goulet, L. H. Nguyen, and S. Amiri, "Tractable approximate Gaussian inference for Bayesian neural networks," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 11 374–11 396, Jan. 2021.

[23] M. H. DeGroot and S. E. Fienberg, "The Comparison and Evaluation of Forecasters," *Journal of the Royal Statistical Society. Series D (The Statistician)*, vol. 32, no. 1/2, pp. 12–22, 1983.

[24] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," in *Proceedings of the 35th international conference on machine learning*, vol. 80, 2018, pp. 2796–2804.

[25] M. Pakdaman Naeini, G. Cooper, and M. Hauskrecht, "Obtaining Well Calibrated Probabilities Using Bayesian Binning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, Feb. 2015.

[26] D. Levi *et al.*, "Evaluating and Calibrating Uncertainty Prediction in Regression Tasks," *Sensors*, vol. 22, no. 15, 2022.

[27] F. Küppers, J. Schneider, and A. Haselhoff, "Parametric and Multivariate Uncertainty Calibration for Regression and Object Detection," in *Computer Vision – ECCV 2022 Workshops*, 2023, vol. 13805, pp. 426–442.

[28] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., 2001.

[29] S. Särkkä and L. Svensson, *Bayesian filtering and smoothing*. Cambridge university press, 2023, vol. 17.

[30] S. S. Haykin, Ed., *Kalman filtering and neural networks*, ser. Adaptive and learning systems for signal processing, communications, and control. New York: Wiley, 2001.

[31] P. Kaminski, A. Bryson, and S. Schmidt, "Discrete square root filtering: A survey of current techniques," *IEEE Transactions on automatic control*, vol. 16, no. 6, pp. 727–736, 1971.

[32] S. J. Julier and J. K. Uhlmann, "Using covariance intersection for SLAM," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 3–20, 2007.

[33] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Springer US, 1992.

[34] S. Maneewongvatana and D. Mount, "It's Okay to Be Skinny, If Your Friends Are Fat," *Center for Geometric Computing 4th Annual Workshop on Computational Geometry*, 01 2000.

[35] D. Gabor, "Theory of communication. Part 1: The analysis of information," *Journal of the Institution of Electrical Engineers-part III: radio and communication engineering*, vol. 93, no. 26, pp. 429–441, 1946.