Voronoi Trust Regions for Local Calibration Testing in Supervised Machine Learning Models

Markus Walker, Philipp S. Bien, 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, uojsw@student.kit.edu, uwe.hanebeck@kit.edu

***Abstract*—The assessment of prediction quality in machine learning models is crucial, particularly within specific regions of the input space, as black-box models do not perform equally well in every region. Common methods, such as mean squared error and calibration measures, are unable to assess local quality. To address this issue, we propose a novel approach based on Voronoi tessellation, which provides a visual and intuitive method for analyzing two-dimensional input spaces. Our method identifies regions in the input space, assesses the calibration of predictions within these regions, and is implemented for regression tasks in multi-input systems. The effectiveness of our approach is exemplified using Bayesian neural networks (BNNs) and shows that our proposed method provides a clearer understanding of the quality of predictions in different input space regions.**

***Index Terms*—Bayesian neural networks, uncertainty quantification, Voronoi tessellation, region merging, calibration testing.**

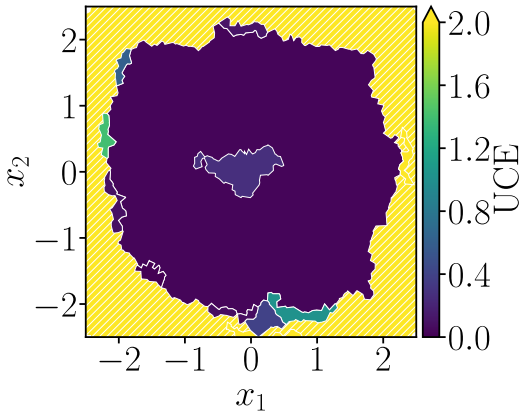


Fig. 1. Identified and tested Voronoi trust regions.

I. Introduction

White-box modeling is a theory-based method that involves analyzing the internal components of a system and utilizing relevant theorems and laws to construct a model under specific assumptions [1]. While this method can yield accurate results, it demands expertise across various domains for effective application [1]. For some complex system processes, white-box modeling presents a significant challenge, as it is difficult to include all relevant aspects [2].

As an alternative to white-box models, learned black-box models, such as neural networks, have gained popularity. For example, these models are employed in optimal control and state estimation, e.g., as a measurement model [1], [3]. However, classic neural networks provide only point estimates, making it unclear how reliable these models are and when they can be trusted. A better approach is to quantify the uncertainty of predictions, e.g., by using Bayesian neural networks (BNNs).

However, training BNNs can only be done approximately, using methods such as Markov Chain Monte Carlo (MCMC) [4] or Variational Inference (VI) [5], which introduce errors. Furthermore, the quality of predictions is also influenced by the lack of data and the choice of hyperparameters. Therefore, it is crucial to verify the trustworthiness of black-box models such as BNNs.

This work is 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.

The assessment of uncertainty estimates usually relies on metrics such as mean squared error, negative log-likelihood, or calibration measures like uncertainty calibration error (UCE) [6], quantile calibration error [7], and averaged normalized estimation error squared [8]. However, these metrics and calibration measures calculate a single score over all test data, without considering the input space location of data points. This means they do not account for regions sparsely covered or not covered at all by training data. Consequently, these regions may never produce appropriate predictive distributions, but this local behavior is not reflected in the single score of these measures.

To overcome this limitation, [9] introduced a new paradigm of asking *when*, or more specifically, in which regions of the input space the predictions are trustworthy. This is achieved by using a *reference model* to identify trustworthy regions in the input space of single-input systems. The reference model is a second model that has been trained for the same purpose in a different manner than the *base model* to be tested. The approach in [9] has been extended to multiple-input systems without using a reference model [10], [11]. However, there are currently no methods for multiple-input systems that incorporate a reference model.

This paper addresses this issue by using Voronoi tessellation to create small cells in the input space and performing region-merging using a reference model. By doing so, it provides

an illustrative perspective on local model quality, as shown in Fig. 1.

Contribution: In this paper, we introduce a novel approach based on Voronoi tessellation, providing an intuitive and visual method for analyzing two-dimensional input spaces. We merge Voronoi cells based on evaluated distance measures between predictions of a base model (e.g., using VI) and a reference model (e.g., using MCMC). Additionally, we assess the calibration of predictions within each identified Voronoi region, ensuring a comprehensive evaluation of model performance. Our method is implemented for regression tasks in multiple-input systems, demonstrating its effectiveness and applicability.

Notation: In this paper, underlined letters, e.g., $\underline{x}$, denote vectors, boldface letters, such as $\boldsymbol{x}$, represent random variables, while sets are represented as calligraphic letters, e.g., $\mathcal{D}$.

II. Related Work

A. Approximate Inference

Unlike classical neural networks with deterministic weights, BNN weights cannot be trained using standard backpropagation. Instead, learning weight distributions relies on fundamental approaches of approximate probabilistic inference, which are detailed in this subsection.

The MCMC method, initially proposed in [4], has emerged as a widely utilized approach for probabilistic inference, particularly in the context of training BNNs. MCMC approximates the weight posterior distribution by sampling from a Markov process. Despite its effectiveness, MCMC has a significant drawback: its high computational cost. This is due to the need to generate numerous samples, as exemplified by the Metropolis–Hastings algorithm [12]. To enhance its efficiency, several improvements have been developed, including Gibbs sampling [13], hybrid Monte Carlo [14], Hamiltonian Monte Carlo [15], and the No-U-Turn Sampler [16].

Another category of approximate inference algorithms for BNNs, collectively known as Variational Inference (VI), has become widely recognized as a promising approach to simplify complex learning tasks. This approach approximates the weight posterior with a more straightforward distribution, commonly referred to as the variational distribution, thereby reducing the complexity of the learning problem. The variational distribution, which is typically a normal distribution, is refined during the training by minimizing the empirical lower bound to the reverse Kullback–Leibler divergence using gradient descent [5]. Several advancements have been introduced to enhance scalability for larger architectures, such as using scaled gradients from random subsets of the training data to update the variational distribution, as implemented in Stochastic Variational Inference [17], or employing deterministic moment propagation instead of sampling in the forward pass [18]. To further reduce computational costs, [19], [20] suggested the dropout technique, which approximates the variational distribution [21].

Expectation Propagation (EP) [22] is a notable approach to the learning of weight posteriors, akin to VI. Both methods simplify the true posterior by approximating it with a more manageable distribution, but EP minimizes the forward Kullback–Leibler divergence, unlike VI, which targets the reverse. The application of EP in BNNs has garnered considerable interest, with prominent examples such as probabilistic backpropagation [23] and its extension [24].

Recent innovations, like Kalman Bayesian Neural Networks [25], leverage sequential Bayesian filtering within each layer, eliminating the need for explicit gradient computation and facilitating rapid sequential learning. As proposed in [26], the Bayesian perceptron represents the fundamental component of the Kalman Bayesian Neural Networks, offering a closed-form solution for the forward pass and weight updates that is equivalent to statistical linearization [10].

B. Calibration Measures

Evaluating prediction quality using calibration measures involves assessing how well predictive distributions match the actual data-generating process. However, this task is challenging due to the finite amount of samples in the test data set and the lack of ground truth for uncertainty estimates. Various methods exist for evaluating machine learning model predictions, such as calibration plots [27], which visually compare predicted and observed confidence levels.

For regression models, scoring rules are typically used to assess the quality of uncertainty estimates. Calibration measures like uncertainty calibration error (UCE) [6] and expected normalized calibration error [28] evaluate the discrepancy between predicted variances and observed mean squared error using binned test data $\mathcal{B}_l$. The UCE is the sum of errors over all bins and is defined by

$$UCE = \sum_{l=1}^{L} \frac{|\mathcal{B}_l|}{N_{\mathrm{Test}}} \left|\mathrm{MSE}(\mathcal{B}_l) - \mathrm{MV}(\mathcal{B}_l)\right| \quad ,$$

where $\mathrm{MSE}(\mathcal{B}_l)$ represents the mean squared error between the predicted means and the output data within the l-th bin, while $\mathrm{MV}(\mathcal{B}_l)$ denotes the mean variance of the predictions within the l-th bin. $|\mathcal{B}_l|$ is the number of test data points within the l-th bin, whereas N_{Test} is the total number of test data points. However, the UCE and expected normalized calibration error are restricted to univariate predictions. For normally distributed predictions with arbitrary dimensions, [7] introduced the quantile calibration error, which compares observed frequencies with selected quantile values of chi-squared distributed errors.

C. Trust Region Identification

To enhance single calibration scores, which are derived from calibration measures across entire test data sets, [9] introduced a novel two-step testing methodology for Bayesian models. This approach focuses on identifying regions within the input space that yield calibrated and trustworthy predictions. The methodology consists of two main phases:

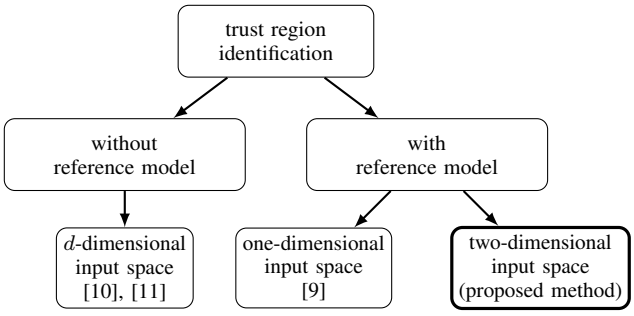


Fig. 2. Taxonomy of trust region identification methods.

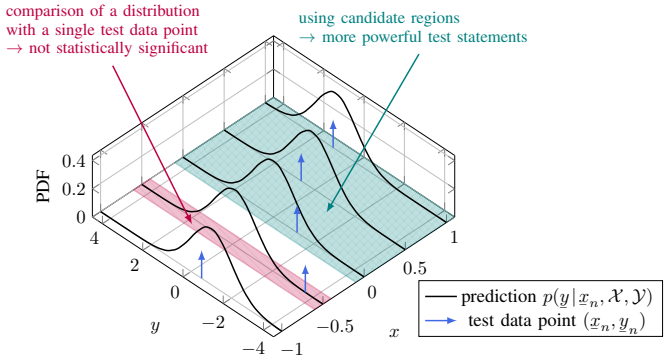


Fig. 3. Local calibration testing for a single-input, single-output system (adapted version from [11]). Within the purple-colored input space region, only one test point is used, which is not statistically significant. Adjacent predictions can be combined into candidate regions, as shown in the teal-colored region, to increase the effectiveness of the test statement.

1) The initial phase involves identifying regions within the input space where test data are present.
2) The subsequent phase evaluates how accurately the predictions align with the data-generating process within the identified regions, utilizing output test data, i.e., the calibration is assessed on a regional basis.

The main aspects of the first phase are the representation of regions and the means of identifying them. In [9], regions are defined as intervals, identified through the use of a reference model. The reference model is employed to obtain reference predictions. By using a distance measure for probability distributions, such as the Wasserstein distance [29], [30], the differences between the base predictions and the reference predictions can be obtained. The distances are used to merge adjacent predictions that exhibit similar distance values. However, these interval-based regions only work for single-input systems. As general models are not restricted to one-dimensional input spaces, [10] and [11] applied the two main phases from [9] to multiple-input systems. This is achieved by utilizing k-d trees [31] or ball trees [32] as candidate region identification methods. Therefore, the regions are given as hyperrectangles or hyperspheres. In contrast to [9], no reference model is employed in [10] and [11], thus avoiding the necessity to train a second model at an additional computational cost. A concise overview of the methodologies is presented in Fig. 2.

As outlined in [9], the second phase of the testing employs statistical tests such as the averaged normalized estimation error squared test [8] and the binomial test to evaluate the calibration of the identified candidate regions. If significant discrepancies are found between the data and the predictions, the candidate region is deemed untrustworthy and rejected by these statistical tests. However, arbitrary calibration measures can also be used to test candidate regions. For instance, the expected calibration error is employed for classification tasks in [10], while the UCE is utilized for regression tasks in [11].

III. Learning Setup

We consider a supervised learning setup where a real system generates N output realizations $\underline{y}_n \in \mathbb{R}^{d_y}$ when provided with deterministic inputs $\underline{x}_n \in \mathbb{R}^{d_x}$, for $n = 1, \ldots, N$. The outputs are typically noisy, and the exact mapping between the inputs and outputs is unknown.

We model the relationship between inputs and outputs using a feedforward BNN $\underline{\boldsymbol{y}} = f(\underline{x}, \underline{\boldsymbol{w}})$, where $\underline{\boldsymbol{y}}$ is the output represented as a random vector, and $\underline{\boldsymbol{w}}$ is the random weight vector containing all network weights. Here, d_w is the total number of (scalar) weights in the network, and $\Omega_{\underline{\boldsymbol{w}}} \subseteq \mathbb{R}^{d_w}$ is the sample space of the weights. Note that the BNN is not the real system generating the data, but a model that approximates the real system. However, we treat the BNN as the probabilistic model that generates the data, and we determine the likelihood function $p(\underline{y}\,|\,\underline{x}, \underline{w})$ for the BNN. For example, if there is no additional noise in the output, the likelihood function is simply given by the Dirac delta function $p(\underline{y}\,|\,\underline{x}, \underline{w}) = \delta(\underline{y} - f(\underline{x}, \underline{w}))$. Alternatively, if there is a (possibly unknown) noise density, the likelihood function can be modeled accordingly, e.g., as Gaussian likelihood. Even though $\underline{x}$ is deterministic, we can condition on $\underline{x}$ in our probabilistic model, where conditioning is denoted by "|". Additionally, we assume prior knowledge about the weight vector in the form of a prior distribution $p(\underline{w})$.

The posterior distribution over the weights $p(\underline{w}\,|\,\mathcal{X}, \mathcal{Y})$ and the predictive distribution $p(\underline{y}\,|\,\underline{x}, \mathcal{X}, \mathcal{Y})$ are then given by

$$p(\underline{w}\,|\,\mathcal{X}, \mathcal{Y}) = \frac{p(\mathcal{Y}\,|\,\mathcal{X}, \underline{w})\, p(\underline{w})}{p(\mathcal{Y}\,|\,\mathcal{X})} \quad , \text{ and}$$
$$p(\underline{y}\,|\,\underline{x}, \mathcal{X}, \mathcal{Y}) = \int_{\Omega_{\underline{\boldsymbol{w}}}} p(\underline{y}\,|\,\underline{x}, \underline{w})\, p(\underline{w}\,|\,\mathcal{X}, \mathcal{Y})\, \mathrm{d}\underline{w} \quad , \qquad (1)$$

where $p(\mathcal{Y}\,|\,\mathcal{X}, \underline{\boldsymbol{w}}) = \prod_{n=1}^{N} p(\underline{y}_n\,|\,\underline{x}_n, \underline{w})$ is the likelihood assuming independent $\underline{y}_n$ given deterministic $\underline{x}_n$, and $p(\mathcal{Y}\,|\,\mathcal{X}) = \int_{\Omega_{\underline{\boldsymbol{w}}}} p(\mathcal{Y}\,|\,\mathcal{X}, \underline{\boldsymbol{w}})\, p(\underline{w})\, \mathrm{d}\underline{w}$ is the normalization constant. The sets $\mathcal{X} = \{\underline{x}_1, \ldots, \underline{x}_N\}$ and $\mathcal{Y} = \{\underline{y}_1, \ldots, \underline{y}_N\}$ represent the given inputs and outputs in the training data set.

It should be noted that there is no available analytical solution for training or prediction. Therefore, in practice, approximation methods must be applied, as described in Sec. II-A.

IV. Test Problem and Key Idea

In this section, we begin by reviewing the test problem associated with Bayesian models, followed by the formulation

of our key idea for this paper. It should be noted that the test problem is identical to those discussed in [11].

A. Test Problem

Assessing the quality of predictions, alongside training and predicting, presents a significant challenge. In the majority of cases, only the finite test data set $\mathcal{D}_{\text{Test}} = \{(\underline{x}_n, \underline{y}_n)\}_{n=1}^{N_{\text{Test}}}$, can be employed for the assessment of prediction quality. Given a learned model, predictive distributions $p(\underline{y} \,|\, \underline{x}_n, \mathcal{X}, \mathcal{Y})$ are obtained for each test input $\underline{x}_n$ using (1). In practice, the true output distribution of the data-generating process, given an input $\underline{x}_n$, is typically unknown. Instead, only one realization $\underline{y}_n$ of the output is available for each input within the test data set.

In an ideal scenario, a method for assessing the alignment between a prediction and the data-generating process, namely whether the prediction is well calibrated, would be available for each prediction. Nevertheless, the evaluation of a prediction $p(\underline{y}|\underline{x}_n, \mathcal{X}, \mathcal{Y})$ derived from a specific data point $\underline{x}_n$, using a single output realization $\underline{y}_n$ lacks statistical significance. For instance, even when a one-dimensional output sample y_n is observed to exceed three standard deviations from the calculated mean value, as displayed in the purple-colored region ▢ in Fig. 3, the probability of this occurrence is not zero. However, it is unlikely that the sample originated from this distribution, although it is still possible.

B. Key Idea

To address this challenge, the methodology proposed in [9] entails not just examining single input–output pairs $(\underline{x}_n, \underline{y}_n)$ (which can be seen as Dirac distribution) for comparison with the prediction $p(\underline{y}|\underline{x}_n, \mathcal{X}, \mathcal{Y})$, but rather forming regions with adjacent predictions and data points, as shown in the teal-colored region ▢ in Fig. 3. Consequently, calibration measures can be employed at the local level within the input space, thus making it possible to make quality statements about specific regions within the input space. The general reference model-based approach, initially introduced in [9], relies on the following assumption:

Assumption 1. *When comparing a base model BNN^{base} (e.g., using VI) with a reference model BNN^{ref} (e.g., using MCMC) using their predictions $p^{\text{base}}(\underline{y}|\underline{x}_n, \mathcal{X}, \mathcal{Y})$ and a reference model $p^{\text{ref}}(\underline{y}|\underline{x}_n, \mathcal{X}, \mathcal{Y})$, it is assumed that their predictions will differ significantly in extrapolation. This difference, often due to a lack of data, can be detected using distance measures for probability distributions, thereby identifying regions where no training data are present.*

However, the method proposed in [9] is limited to single-input systems. To overcome this limitation, our key idea is to use the Voronoi tessellation of the input space, which is suitable for multiple-input systems. This involves calculating distances between the base and reference model predictions and merging Voronoi cells that exhibit similar distances into regions, as shown in Fig. 4. Then, as in [10], calibration measures can be used to test the local quality per region.

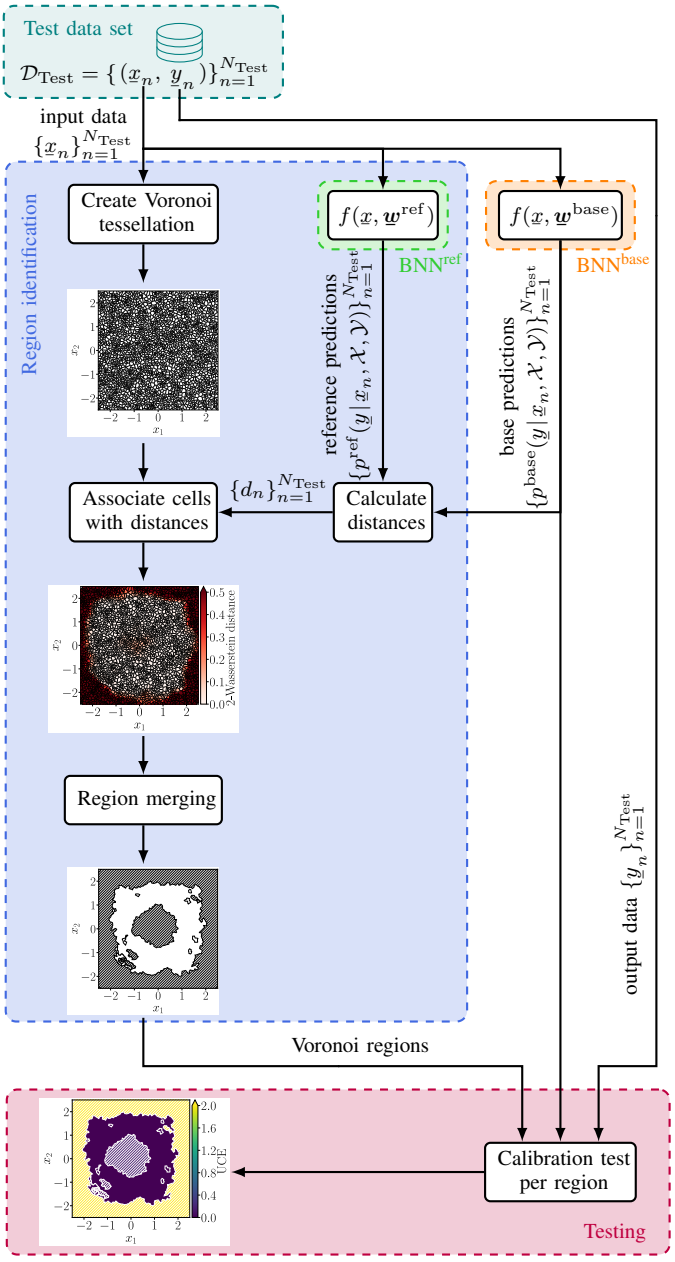


Fig. 4. Illustration of our proposed testing strategy. The strategy compares two models, a base model BNN$^{\text{base}}$ and a reference model BNN$^{\text{ref}}$, to identify regions in the input space using distance measures for probability distributions, the Voronoi tessellation, and a region merging method. It then uses calibration measures within the identified regions to assess their trustworthiness.

V. Region Merging Algorithm

Given the set of test input data points $\mathcal{X}_{\text{Test}} = \{\underline{x}_n\}_{n=1}^{N_{\text{Test}}}$, the Voronoi tessellation divides the input space into N_{Test} cells $\mathcal{V}(\underline{x}_n)$ such that each cell $\mathcal{V}(\underline{x}_n)$ consists of all points closer to $\underline{x}_n \in \mathcal{X}_{\text{Test}}$ than to any other point $\underline{x}_m \in \mathcal{X}_{\text{Test}}$ with $m \neq n$. Formally,

$$\mathcal{V}(\underline{x}_n) = \left\{ \underline{x} \in \mathbb{R}^{d_{\underline{x}}} \;\middle|\; \mathrm{d}^{\mathcal{V}}(\underline{x}, \underline{x}_n) < \mathrm{d}^{\mathcal{V}}(\underline{x}, \underline{x}_m), \;\; \forall m \neq n \right\}$$

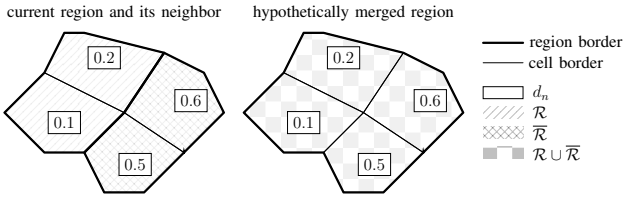


Fig. 5. Example of a Voronoi region merging step. Region $\mathcal{R}$ and its neighbor $\overline{\mathcal{R}}$ are considered for evaluating the merge predicate (2). The set of combined distances $\mathcal{W}_{\mathcal{R},\overline{\mathcal{R}}}$ is $\{0.1, 0.2, 0.5, 0.6\}$. Accordingly, both regions are merged if $\epsilon > 0.5$.

where $\mathrm{d}^{\mathcal{V}}(\cdot,\cdot)$ denotes a distance measure between two points, e.g., the Euclidean distance.

Based on the idea in [9], the distance between the base model predictions and the reference model predictions is used to merge Voronoi regions with similar distance values. Therefore, the distances between the base predictions and the reference predictions are given by

$$d_n = \mathrm{d}\big(p^{\mathrm{base}}(\underline{y}\,|\,\underline{x}_n, \mathcal{X}, \mathcal{Y}),\, p^{\mathrm{ref}}(\underline{y}\,|\,\underline{x}_n, \mathcal{X}, \mathcal{Y})\big), \forall \underline{x}_n \in \mathcal{X}_{\mathrm{Test}},$$

where $\mathrm{d}(\cdot,\cdot)$ is a distance between probability distributions, e.g., the 2-Wasserstein distance [33]. Therefore, the used information for the region merging is given by

$$\mathcal{I} = \{(\mathcal{V}(\underline{x}_n), d_n)\}_{n=1}^{N_{\mathrm{Test}}}$$

and assigns a distance between probability distributions to each Voronoi cell.

Using the Voronoi tessellation of the given input test points, the Voronoi region merging algorithm employs a region-growing approach. The idea of the algorithm is simple: adjacent Voronoi cells that have similar distances d_n are merged together to create larger regions. The region merging algorithm employs the concept of a Voronoi region, wherein a Voronoi region $\mathcal{R} = \{\mathcal{V}(\underline{x}_n)\}$ is defined as a set of adjacent Voronoi cells that together form a regional entity within the input space. A Voronoi cell $\mathcal{V}(\underline{x}_n)$ can only be contained in a single Voronoi region at a time, i.e., Voronoi regions are pair-wise disjoint: $\mathcal{R} \cap \overline{\mathcal{R}} = \emptyset$ for any two Voronoi regions $\mathcal{R}$ and $\overline{\mathcal{R}}$.

The Voronoi region merging algorithm operates in two steps, which will be explained in more detail below. Subsequently, the selection of its hyperparameters will be discussed.

A. Step 1: Merging Voronoi Cells

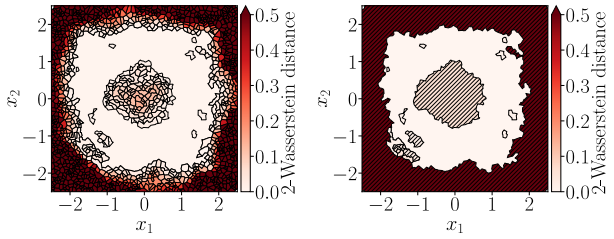
Initially, for each Voronoi cell $\mathcal{V}(\underline{x}_n)$, a Voronoi region is created, resulting in a total of N_{Test} Voronoi regions

$$\mathcal{R}_n = \{\mathcal{V}(\underline{x}_n)\}, \quad \forall n = 1, \ldots, N_{\mathrm{Test}} \quad .$$

The initial Voronoi regions are then sorted based on their respective distances d_n in ascending order. Subsequently, the Voronoi regions are iterated in their sorted order. For each region, the neighboring regions are evaluated using the merge predicate $\mathrm{Pred}(\mathcal{R}, \overline{\mathcal{R}})$, where $\mathcal{R}$ and $\overline{\mathcal{R}}$ are the current Voronoi region and its neighbor, respectively. If the merge predicate is satisfied, both regions are regarded as *similar* and are merged. The merge predicate is defined as

$$\begin{aligned} \mathrm{Pred}(\mathcal{R}, \overline{\mathcal{R}}) &= |\max(\mathcal{W}_{\mathcal{R},\overline{\mathcal{R}}}) - \min(\mathcal{W}_{\mathcal{R},\overline{\mathcal{R}}})| < \epsilon\,, \\ \mathcal{W}_{\mathcal{R},\overline{\mathcal{R}}} &= \{d_n \,|\, \mathcal{V}(\underline{x}_n) \in \mathcal{R} \cup \overline{\mathcal{R}}\} \quad , \end{aligned} \tag{2}$$

where $\mathcal{W}_{\mathcal{R},\overline{\mathcal{R}}}$ is the set of the combined distances corresponding to $\mathcal{R}$ and its neighbor $\overline{\mathcal{R}}$. I.e., two Voronoi regions will be merged if the min-max span of all distances within the *hypnotically* merged region $\mathcal{R} \cup \overline{\mathcal{R}}$ is lower than the threshold ϵ, as displayed in Fig. 5.



(a) Regions before step 2. (b) Regions after step 2.

Fig. 6. Illustration of an outlier merging process. The shaded areas indicate the merged adjacent outlier regions. The average model differences within regions are color-coded.

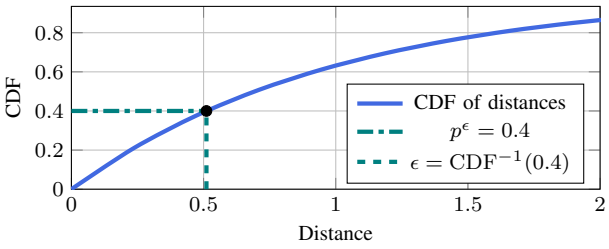
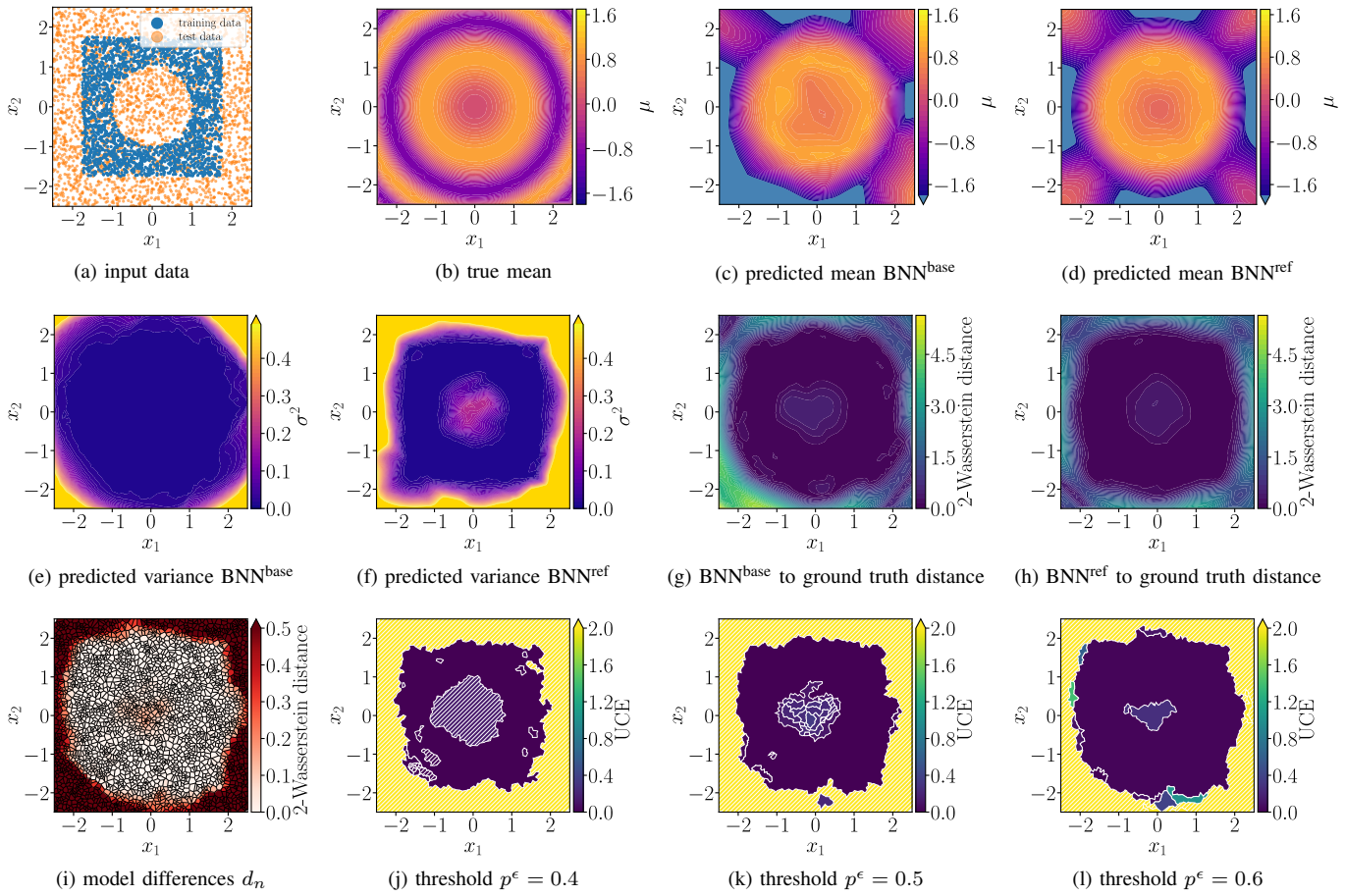


Fig. 7. Selection of the merge threshold value ϵ using the inverse cumulative distribution function of the distances d_n.

B. Step 2: Merging Outlier Regions

Following the completion of step 1, regions $\mathcal{R}$ that contain less than λ Voronoi cells, i.e., $|\mathcal{R}| < \lambda$, are classified as *outlier regions*. For these regions, the surrounding Voronoi regions exhibited *dissimilarities* with regard to their corresponding distances, thereby invalidating the merge predicate. However, adjacent outlier regions are merged, with the result of the merge predicate being *ignored*. The second step of the algorithm reduces the number of total regions, provided that the outlier regions are adjacent. For instance, if two adjacent outlier regions each contain a single Voronoi cell—that is if each region contains only one data point—then the two regions are merged. This is important since test measures using a single data point are not statistically significant, as discussed in Sec. IV-A. Moreover, the merging of outliers does not impact the regions of interest, which are the regions exhibiting similarities according to Assumption 1. An example of this step is shown in Fig. 6.



(a) input data (b) true mean (c) predicted mean BNN$^{\text{base}}$ (d) predicted mean BNN$^{\text{ref}}$

(e) predicted variance BNN$^{\text{base}}$ (f) predicted variance BNN$^{\text{ref}}$ (g) BNN$^{\text{base}}$ to ground truth distance (h) BNN$^{\text{ref}}$ to ground truth distance

(i) model differences d_n (j) threshold $p^\epsilon = 0.4$ (k) threshold $p^\epsilon = 0.5$ (l) threshold $p^\epsilon = 0.6$

Fig. 8. Results of the Nonlinear Regression Example. The training and test data used are displayed in (a), while the true mean of the data-generating process is shown in (b). The predicted means and variances by the base model BNN are shown in (c)–(f). The 2-Wasserstein distances between the true data-generating process are presented in (g) and (h), while the differences between the base and the reference model are displayed in (i). The results of our proposed testing method are shown in (j)–(l) using different thresholds p^ϵ and the UCE [6] as a calibration measure, with the outlier regions highlighted as shaded areas.

C. Selection of Threshold ϵ

The threshold value ϵ utilized in the merge predicate (2) is a hyperparameter within the Voronoi region merging algorithm. The resulting Voronoi regions are dependent upon the selected threshold, as a higher threshold directly increases the probability of two regions being merged. However, the selection of this parameter depends on the distribution of d_n. In order to ensure that the selection is independent of the distribution of the distances, the threshold is selected by

$$\epsilon = \text{CDF}^{-1}(p^\epsilon) \ ,$$

where $\text{CDF}^{-1}(\cdot)$ is the inverse cumulative distribution function of the distances, i.e., the quantile function of d_n. This enables the selection of a threshold value ϵ by specifying a probability $p^\epsilon \in [0, 1]$, and deriving the corresponding distance value from the cumulative density function. E.g., a probability of 0.4 would result in a threshold value that corresponds to a distance value that is larger than $40\,\%$ of all distances d_n, as illustrated in Fig. 7.

VI. Numerical Evaluation

A. Settings

The proposed testing method is demonstrated using the nonlinear regression example

$$\boldsymbol{y} = \sin\left(x_1^2 + x_2^2\right) + \boldsymbol{\epsilon} \ , \tag{3}$$

where 4500 training points and 3000 points are generated using $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 0.1)$. The training inputs $\underline{x}_n \in \mathcal{X}$ and the test inputs $\underline{x}_n \in \mathcal{X}_{\text{Test}}$ are each uniformly drawn from $[-1.75, 1.75] \times [-1.75, 1.75]$ and $[-2.5, 2.5] \times [-2.5, 2.5]$, respectively. To create a gap in the training data, $30\,\%$ of the data was removed, by creating a hole around the origin, as shown in Fig. 8a.

We use a fully connected feedforward network comprising a single hidden layer with 50 neurons and ReLU activation for the hidden layer, and linear activation for the output layer, which is trained using the No-U-Turn Sampler [16] and Stochastic Variational Inference [17] as reference model BNN$^{\text{ref}}$ and base model BNN$^{\text{base}}$, respectively.

In order to create Voronoi diagrams, the Euclidean distance is employed for $\text{d}^{\mathcal{V}}(\cdot, \cdot)$. To compare the base model predictions with the reference model predictions, the 2-Wasserstein distance between two normal distributions is utilized for $\text{d}(\cdot, \cdot)$.

The 2-Wasserstein distance between two normal distributions $p_1 = \mathcal{N}(\mu_1, \sigma_1^2)$ and $p_2 = \mathcal{N}(\mu_2, \sigma_2^2)$ is given by [33]

$$W_2(p_1, p_2) = (\mu_1 - \mu_2)^2 + (\sigma_1 - \sigma_2)^2 \quad . \tag{4}$$

The assessment of the calibration per identified region is done using the UCE [6]. Regions containing less than $\lambda = 10$ Voronoi cells are considered as outliers.

To evaluate if the predictions actually are good, we use the data-generating process (3) and calculate distances to the predictions of the respective model, using the 2-Wasserstein distance (4). It should be noted that this is not part of the proposed algorithm. The distances to the true stochastic process can be regarded as distances to the ground truth.

B. Results

The data, predictions, and test results of our proposed method are presented in Fig. 8. By comparing the 2-Wasserstein distance between the true data-generating process and the predictions, it can be seen that the base model has effectively learned the data-generating process, where training data are available, as evidenced by the near-zero 2-Wasserstein distances (Fig. 8g). In the outer areas ($|x_1|, |x_2| > 1.75$), where no training data are available, the predictions deviate significantly from the ground truth, considering the predicted mean (Fig. 8c) as well as the predicted variance (Fig. 8e). Around the origin, where training data are missing, the base model's mean deviates from the true mean. While the predictions around the origin are not accurate, they are less erroneous than in the outer regions, as evidenced by the 2-Wasserstein distances in Fig. 8g.

The same pattern can be seen (without the use of ground truth information) through our presented method, using the UCE per identified Voronoi region. For instance, in Figs. 8j and 8l, the outer regions are identified as badly calibrated according to their UCE scores. Furthermore, the region around the origin has lower UCE scores than the outer regions, which are also aligned with the ground truth distances.

In Figs. 8j to 8l, the results of our method are shown for different thresholds for the merge predicate. It can be seen that a threshold of $p^\epsilon = 0.4$, as illustrated in Fig. 8j, results in more conservative regions than a higher threshold, such as $p^\epsilon = 0.6$, shown in Fig. 8l. Here, conservatism implies that the regions are less extended towards areas where no training data are available. Moreover, the impact of the selected threshold is evident in the area surrounding the origin, where the base and reference models exhibit dissimilarities according to the merge predicate with $p^\epsilon = 0.4$, resulting in this region being identified as an outlier region. However, employing $p^\epsilon = 0.5$ results in multiple non-outlier regions in the central area.

C. Discussion

The proposed method builds upon the two-phase testing strategy introduced in [9]. Initially, the input space is partitioned into Voronoi cells. Then, adjacent Voronoi cells are merged into regions if they are similar with respect to distances between base and reference model predictions. Subsequently, these regions are tested using calibration measures, such as the UCE. It should be noted that the reference model and the base model can be swapped, which, in the method presented here, results in a change to the model that is tested in the second phase.

The region identification process is capable of dividing the input space into regions. In contrast to the hyperrectangular or hyperspherical regions described in [10], [11], the shape of the Voronoi regions is not predefined. A comparison of the 2-Wasserstein distance between the predictions and the known ground truth of the example in Fig. 8 reveals that our method is able to detect untrustworthy regions and calculate how well specific regions match the real stochastic process using the UCE. The hyperparameter, which represents the threshold of the merge predicate, can be employed to refine the region. Lower threshold values entail a more conservative approach, as regions must exhibit greater similarity in their distance values to be merged.

It should be noted that local testing in regions of the input space necessitates the utilization of a considerable quantity of testing data, as evidenced by the regression example, wherein approximately $\approx 50\,\%$ of the data is employed as test data. Nevertheless, this is a characteristic of region-based testing, which necessitates a substantial quantity of data to achieve a high-resolution partitioning of the input space.

However, building Voronoi tessellations explicitly in high-dimensional spaces requires complex algorithms [34]. Therefore, we recommend using it only for low-dimensional input spaces, such as two input dimensions, where our method provides an intuitive and visual way to gain information about the predictive capabilities and trustworthiness of black-box models such as BNNs. For larger input dimensions, the ball tree and k-d trees-based methods discussed in Sec. II-C are better suited.

VII. Conclusion

In conclusion, this paper presents a novel testing method for Bayesian models, such as Bayesian neural networks (BNNs), that initially employs Voronoi tessellation and region merging to identify input space regions. Subsequently, the local quality of predictions is evaluated using calibration measures for each identified region. A key contribution of our work is the provision of local calibration information, which is particularly valuable for ensuring the trustworthiness of models. The local calibration information offers deeper insight into the model's predictive capabilities within specific regions of the input space, thereby enhancing the trustworthiness of the models in question.

In the future, we want to leverage the developed method in the field of state estimation and optimal control. For instance, the incorporation of local calibration information in state estimation, with the objective of improving trust in black-box models, which can be employed as measurement models, represents a potential direction for further research.

References

[1] Y. Bai, B. Yan, C. Zhou, T. Su, and X. Jin, "State of Art on State Estimation: Kalman Filter Driven by Machine Learning," *Annual Reviews in Control*, vol. 56, p. 100909, Jan. 2023.

[2] Z. Cui, J. Dai, J. Sun, D. Li, L. Wang, and K. Wang, "Hybrid Methods Using Neural Network and Kalman Filter for the State of Charge Estimation of Lithium-Ion Battery," *Mathematical Problems in Engineering*, vol. 2022, no. 1, p. 9616124, 2022.

[3] W. He, N. Williard, C. Chen, and M. Pecht, "State of Charge Estimation for Li-ion Batteries Using Neural Network Modeling and Unscented Kalman Filter-Based Error Cancellation," *International Journal of Electrical Power & Energy Systems*, vol. 62, pp. 783–791, Nov. 2014.

[4] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[5] 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.

[6] M.-H. Laves, S. Ihler, J. F. Fast, L. A. Kahrs, and T. Ortmaier, "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] 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.

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

[9] M. Walker, M. Reith-Braun, P. Schichtel, M. Knaak, and U. D. Hanebeck, "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.

[10] M. Walker, H. Amirkhanian, M. F. Huber, and U. D. Hanebeck, "Trustworthy Bayesian Perceptrons," in *2024 27th International Conference on Information Fusion (FUSION)*, Venice, Italy, Jul. 2024, pp. 1–8.

[11] M. Walker and U. D. Hanebeck, "Multi-Scale Uncertainty Calibration Testing for Bayesian Neural Networks Using Ball Trees," in *2024 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, Plzeň, Czech Republic, Sep. 2024, pp. 1–7.

[12] W. K. Hastings, "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, Apr. 1970.

[13] 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, Nov. 1984.

[14] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," *Physics Letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[15] R. M. Neal, *Bayesian Learning for Neural Networks*, ser. Lecture Notes in Statistics. New York, NY: Springer, 1996, vol. 118.

[16] M. D. Homan and A. Gelman, "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1593–1623, Jan. 2014.

[17] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic Variational Inference," *Journal of Machine Learning Research*, 2013.

[18] A. Wu, S. Nowozin, E. Meeds, R. E. Turner, J. M. Hernández-Lobato, and A. L. Gaunt, "Deterministic Variational Inference for Robust Bayesian Neural Networks," in *International Conference on Learning Representations*, 2019.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[20] Y. Gal, J. Hron, and A. Kendall, "Concrete Dropout," in *Advances in neural information processing systems*, vol. 30, 2017.

[21] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, 2016, pp. 1050–1059.

[22] T. P. Minka, "A Family of Algorithms for Approximate Bayesian Inference," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[23] J. M. Hernández-Lobato and R. P. Adams, "Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, Lille, France, Jul. 2015, pp. 1861–1869.

[24] S. Ghosh, F. M. D. Fave, and J. Yedidia, "Assumed Density Filtering Methods for Learning Bayesian Neural Networks," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 1589–1595.

[25] 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.

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

[27] 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.

[28] D. Levi, L. Gispan, N. Giladi, and E. Fetaya, "Evaluating and Calibrating Uncertainty Prediction in Regression Tasks," *Sensors*, vol. 22, no. 15, 2022.

[29] L. V. Kantorovich, "Mathematical Methods of Organizing and Planning Production," *Management Science*, vol. 6, no. 4, pp. 366–422, 1960.

[30] L. N. Vaserstein, "Markov Processes over Denumerable Products of Spaces, Describing Large Systems of Automata," *Problemy Peredachi Informatsii*, vol. 5, no. 3, pp. 64–72, 1969.

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

[32] S. M. Omohundro, "Five Balltree Construction Algorithms," International Computer Science Institute, Tech. Rep. 89-063, Dec. 1989.

[33] G. Peyré and M. Cuturi, "Computational Optimal Transport: With Applications to Data Science," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[34] F. Aurenhammer, "Voronoi Diagrams—a Survey of a Fundamental Geometric Data Structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, Sep. 1991.