Multi-Scale Uncertainty Calibration Testing for Bayesian Neural Networks Using Ball Trees

Markus Walker and Uwe D. Hanebeck

***Abstract*— Bayesian neural networks (BNNs) offer an elegant and promising approach to quantifying the uncertainty of neural network predictions by providing predictive distributions. Although the potential of BNNs is considerable, established BNN training methods often result in inaccurate uncertainty estimation and local differences in quality depending on the considered input space region. To assess the efficacy of Bayesian models such as BNNs and gain insights into their predictive capabilities in distinct input space regions, we introduce a novel methodology that utilizes ball trees as a space partitioning data structure. Our approach enables the assessment of the predictive quality within specific regions of the input space across multiple scales in the input space, utilizing all nodes provided by the ball tree structure. Furthermore, our method allows the combination of results across different scales.**

***Index Terms*— Bayesian neural networks, uncertainty quantification, space-partitioning, ball trees, calibration testing.**

I. Introduction

Bayesian neural networks (BNNs) have gained widespread use in various fields, such as reinforcement learning [1] and model predictive control [2], due to their ability to quantify uncertainty. This characteristic extends the capabilities of classical neural networks by providing predictive distributions, which have shown significant promise. In real-world applications where uncertainties are inherent, predictive distributions offer a valuable solution. They permit the estimation of the confidence of a network's predictions and the implementation of corrective measures if necessary.

Nevertheless, despite the sophisticated approach BNNs adopt to address uncertainty quantification, their training process can only be approximated. In practice, approximation methods such as Markov Chain Monte Carlo (MCMC) [3], Variational Inference (VI) [4], Expectation Propagation (EP) [5], or Kalman filtering techniques [6] are typically employed for the training of BNNs. However, the use of approximation methods makes BNNs sensitive to inaccuracies, which necessitates the testing of predictions.

Moreover, the testing process is challenging due to the limited number of available test samples and the unknown nature of the data-generating process. Consequently, there are test measures such as the mean squared error, negative log-likelihood, or the uncertainty calibration error (UCE) [7] which utilize the sampled test data to provide insights into the predictive capabilities. Nevertheless, due to the presence of sources of error such as model assumptions, approximate inference, and a finite number of training data points, the quality of predictions varies depending on the input values [8]. E.g., regions of the input space that are densely covered by training data are more likely to approximate the data-generating process accurately than sparsely covered or uncovered regions.

To address this issue, [8] proposed a two-step 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 or calibration measures. This strategy was initially designed for single-input systems and later extended to multi-input systems [9]. However, the prediction quality is only evaluated on a single scale, that is, the size of the identified regions, without consideration of how the quality may vary over different scales.

This is where our paper introduces a paradigm shift, namely the interest in how quality evolves over multiple scales. We propose a multi-scale approach for evaluating BNNs, similar to viewing a landscape from multiple altitudes, each offering a unique perspective. By using ball trees [10], an efficient space partitioning data structure, we can explore the input space at various scales, providing a more holistic understanding of the prediction quality.

Contribution: In this paper, we introduce a novel approach that employs ball trees for spatial partitioning of the input space of BNNs. This method enables the identification of candidate regions where the quality of predictions is evaluated within specific regions of the input space using any chosen test metrics. Furthermore, the structure of ball trees enables multi-scale evaluation. Our approach is demonstrated through regression and binary classification examples.

Notation: In this paper, underlined letters, e.g., $\underline{x}$, denote vectors and boldface letters, such as $\underline{\boldsymbol{x}}$, represent random variables.

II. Related Work

A. Approximate Inference

In contrast to classical neural networks with deterministic weights, the weights of BNNs cannot be trained using standard backpropagation. Instead, the methods for learning weight distributions are based on the fundamental approaches of approximate probabilistic inference, which are explained in this subsection.

The MCMC approach, initially proposed in [3], has emerged as a highly effective and widely utilized approach for probabilistic inference, particularly in the context of training

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.

Markus Walker and Uwe D. Hanebeck are with the Intelligent Sensor-Actuator-Systems Laboratory (ISAS), Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Germany (e-mail: markus.walker@kit.edu; uwe.hanebeck@kit.edu).

BNNs. MCMC operates by approximating the posterior distribution through sampling from a Markov process. Despite its effectiveness, a notable drawback of MCMC is its high computational cost. This is due to the generation of numerous samples, as seen in the Metropolis–Hastings algorithm [11]. To improve its efficacy, a number of enhancements have been made, including Gibbs sampling [12], hybrid Monte Carlo [13], Hamiltonian Monte Carlo [14], and its extension, the No-U-Turn Sampler [15], which reduces hyperparameters.

In the realm of BNNs, Variational Inference (VI) [16] reformulates the complex learning problem into a tractable optimization task by approximating the weight posterior with a simpler distribution, typically a normal distribution, i.e., the variational distribution. During training, the variational distribution is updated by minimizing the empirical lower bound to the reverse Kullback–Leibler divergence using gradient descent. A number of improvements have been proposed, including the enhancement of scalability for larger architectures through the utilization of scaled gradients derived from random subsets of the training data to update the variational distribution, as performed in Stochastic Variational Inference [17] or the usage of deterministic moment propagation instead of sampling in the forward pass [18].

Expectation Propagation (EP) [5] represents a powerful approach for learning weight posteriors, comparable to VI. EP also approximates the true posterior distribution with a simpler one. However, in contrast to VI, EP minimizes the forward Kullback–Leibler divergence, whereas VI minimizes the reverse Kullback–Leibler divergence. The implementation of EP in the context of BNNs has been the subject of significant attention, with notable examples including probabilistic backpropagation [19] and its extension as proposed in [20].

Other recent approaches, such as the Kalman Bayesian Neural Networks (KBNN) [21] and Tractable Approximate Gaussian Inference [22], are primarily based on sequential Bayesian filtering in each layer, as originally proposed in [6]. This avoids explicit gradient computation during training, allowing them to be used for fast sequential learning. In order to achieve this, the Bayesian perceptron was proposed in [23], which forms the core building block of the KBNN and provides a closed-form solution for the forward pass and weight update. Furthermore, in [9], it was proven that this method is equivalent to the statistical linearization of perceptrons.

The stated training methods are based on fundamental principles of probabilistic inference, yet they are often computationally intensive and more complex to implement than conventional training of neural networks. Therefore, some methods intend to provide simpler implementations and computationally cheaper approximations of the weight posterior, such as the dropout technique proposed by [24], which is as an approximation of VI [25], or bootstrap posterior ensemble methods of different models [26], [27], [28].

B. Calibration Measures

The assessment of prediction quality through the use of calibration measures is based on the evaluation of how well the predictive distributions reflect the actual data-generating process. However, given that only a limited number of samples of the data-generating process are available in the test data set and that there is no available ground truth for the uncertainty estimates, the evaluation of calibration is a challenging task. There exists a multitude of approaches for the assessment of the predictions of machine learning models. E.g., calibration plots can be employed for both classification [29] and regression [30], enabling a visual comparison of the predicted and observed confidence levels across all test data.

For classification models, the expected calibration error (ECE) [31] is frequently employed for assessing calibration, whereby the discrepancy between the predicted confidence of the model and its accuracy is quantified utilizing binned test data. The ECE is given by

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

where $|B_l|$ is the number of test data points within the l-th bin, N_{Test} is the number of all test data points, and $\frac{|B_l|}{N_{\text{Test}}}$ is the empirical probability of test data falling into the l-th bin. The absolute difference between the rate of correct classifications $\text{acc}(B_l)$, which is the accuracy, and the average predicted probability score per bin $\text{conf}(B_l)$ is calculated and then summed over all L bins, weighted by the empirical probability per bin $\frac{|B_l|}{N_{\text{Test}}}$.

For regression models, the assessment of uncertainty estimate quality typically employs scoring rules. For univariate normal distribution predictions, calibration measures, such as the uncertainty calibration error (UCE) [7] and the expected normalized calibration error [32], are used to assess the discrepancy between predicted variances and observed mean squared error by leveraging the relationship between the two. This is also done by using bins similar to those used in the ECE. The UCE is given by

$$UCE = \sum_{l=1}^{L} \frac{|B_l|}{N_{\text{Test}}} |\text{MSE}(B_l) - \text{MV}(B_l)| \tag{2}$$

where $\text{MSE}(B_l)$ is the mean squared error between the predictions and the output data in the l-th bin, and $\text{MV}(B_l)$ is the mean variance of the predictions in the l-th bin. To measure calibration for arbitrary dimensional normally distributed predictions proposed the quantile calibration error, which compares the observed frequencies and the desired quantile values of the chi-squared distributed errors. In the case of normally distributed predictions with arbitrary dimensionality, [33] proposed the quantile calibration error to assess calibration by comparing the observed frequencies per quantile and the chi-squared distributed error quantiles.

C. Trust Region Identification

To extend single calibration scores, typically provided by calibration measures overall test data, to trust regions for Bayesian models, [8] proposed a two-step testing procedure. This procedure involves identifying input space regions that lead to calibrated and trustworthy predictions and comprises two principal steps:

1) The identification of candidate regions where information is present, i.e., input space regions where test data are available.
2) The measurement of calibration within these candidate regions using output test data.

In addition, for the special case of single-input, single-output systems, a variant of the two-step testing strategy was presented and demonstrated. The initial step of candidate region identification involves using a second model as a reference model, assuming that the predictions of the approximate model and the reference model will differ for the same test input if not enough information is available during training or approximate inference methods have led to errors. To assess the differences between predictions of both models for each input, the 1-Wasserstein distance is used, and if the distance exceeds a certain threshold, the one-dimensional input space is split, resulting in candidate regions represented by intervals.

As general models are not constrained to one-dimensional input spaces, [9] extends the candidate region identification from [8] to multiple-input systems. This is achieved by utilizing k-d trees [34] as a candidate region identification method, whereby hyperplanes split the input space along orthogonal axes to partition the input space into hyperrectangular candidate regions. In contrast to [8], no reference model is employed, thus avoiding the necessity to train a second model at an additional computational cost.

The second step of the testing strategy proposed by [8], namely the testing of identified candidate regions, is implemented using statistical tests for regression tasks, such as the binomial test and the averaged normalized estimation error squared test [35]. In cases where there are significant discrepancies between the data and the predictions, the candidate region is rejected by statistical tests and therefore deemed untrustworthy. The binomial test enables the testing of specific confidence intervals and makes no assumptions about the distribution of predictions and data. Consequently, it is categorized as a nonparametric test. E.g., the binomial test can be utilized to verify whether the $95\,\%$ confidence interval of the predictions encompasses $95\,\%$ of the output test data. In the case of normally distributed predictions, the parametric averaged normalized estimation error squared test [35], which is based on the average of the squared Mahalanobis distances between the test data and the predictions, is employed to verify whether the data are consistent with the predicted distribution. However, arbitrary calibration measures can test candidate regions, as demonstrated in [9], which employed the ECE for classification tasks.

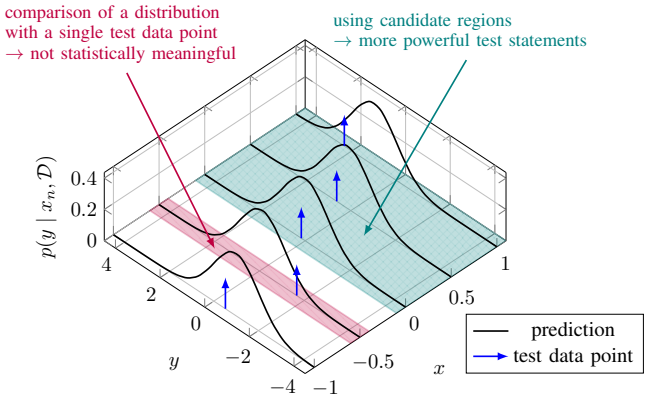


Fig. 1: Illustration of local calibration testing for a single-input, single-output system. In the purple-colored input space region, only a test point is utilized, which is not statistically meaningful. To enhance the test statement's efficacy, adjacent predictions can be combined to form candidate regions, as illustrated in the teal-colored input space region.

III. Learning Setup

A supervised learning setup with a feedforward BNN with L layers is considered. The feedforward BNN is represented by $\underline{\boldsymbol{y}} = f\,(\underline{x}, \underline{\boldsymbol{w}})$, where $\underline{x}$ is the $d_{\underline{x}}$-dimensional input, $\underline{\boldsymbol{w}}$ is the weight vector containing all network weights as random variables, and $\underline{\boldsymbol{y}}$ is the $d_{\underline{y}}$-dimensional output represented as random vector. The prior distribution of the weight vector is given by $p(\underline{w})$. The training dataset, denoted by $\mathcal{D} = \{(\underline{x}_n, \underline{y}_n)\}_{n=1}^{N}$, comprises N independent and identically distributed pairs. Each pair is represented by an input vector $\underline{x}_n \in \mathbb{R}^{d_{\underline{x}}}$ and its respective output $\underline{y}_n \in \mathbb{R}^{d_{\underline{y}}}$. In order to train the BNN, the weight posterior 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 $p(\mathcal{Y} \mid \mathcal{X})$ is the normalization constant and $p(\mathcal{Y} \mid \mathcal{X}, \underline{w})$ is the likelihood. The inputs and outputs of the training set $\mathcal{D}$, are denoted by $\mathcal{X} = \{\underline{x}_1, \ldots, \underline{x}_N\}$ and $\mathcal{Y} = \{\underline{y}_1, \ldots, \underline{y}_N\}$ respectively. Given a deterministic input $\underline{x}$, the prediction $p(\underline{y} \mid \underline{x}, \mathcal{D})$ is obtained by

$$p(\underline{y} \mid \underline{x}, \mathcal{D}) = \int_{\Omega_{\underline{\boldsymbol{w}}}} p(\underline{y} \mid \underline{x}, \underline{w})\, p(\underline{w} \mid \mathcal{D})\, \mathrm{d}\underline{w} \ , \qquad (3)$$

with the learned posterior distribution $p(\underline{w} \mid \mathcal{D})$. However, there is no analytical solution for either training or prediction. Consequently, approximation methods must be employed in practice, as outlined in Sec. II-A.

IV. Test Problem and Key Idea

In addition to training and predicting with BNNs, assessing the quality of the predictions is a challenging task. Typically, the only information available is the test data set $\mathcal{D}_{\text{Test}} = \{(\underline{x}_n, \underline{y}_n)\}_{n=1}^{N_{\text{Test}}}$. Each test input $\underline{x}_n$ results in a predictive distribution $p(\underline{y} \mid \underline{x}_n, \mathcal{D})$ according to (3), as shown in Fig. 1. The true distribution $p(\underline{y} \mid \underline{x})$ is typically unknown and only one output sample $\underline{y}_n$ is available for each input $\underline{x}_n$. Consequently, the distance between the true distribution

$p(\underline{y} \mid \underline{x})$, and the predicted distribution $p(\underline{y} \mid \underline{x}, \mathcal{D})$, cannot be calculated.

Ideally, for each prediction there would be a method for assessing whether the prediction aligns with the data-generating process, that is, whether it is well calibrated. However, evaluating a prediction $p(\underline{y}_n \mid \underline{x}_n, \mathcal{D})$ from a specific input $\underline{x}_n$ using a single output sample $\underline{y}_n$ is not statistically meaningful. E.g., even if a one-dimensional output sample y_n is greater than six standard deviations from the mean value, the probability is not zero and the sample could be from this distribution, although the probability is very low.

Accordingly, the concept proposed in [8] is to not solely consider individual input-output pairs $(\underline{x}_n, \underline{y}_n)$ for comparison with the prediction $p(\underline{y} \mid \underline{x}_n, \mathcal{D})$. Instead, the approach involves forming candidate regions with adjacent predictions, in which multiple test points can be used to generate statements regarding specific regions in the input space. This approach addresses the limitation of calibration measures such as the UCE [7], which were unable to make statements about specific regions in the input space.

However, defining the size of a region results in a trade-off between the spatial resolution of the candidate regions and the power of the statistical test results. E.g., if a candidate region is large, the test result will be less precise with regard to the input space position. Conversely, if the region is too small, there may be an insufficient number of test data points to make a meaningful statement regarding the region. This trade-off can be compared to the uncertainty principle in the short-time Fourier transform, where there is a trade-off between time and frequency resolution [36].

To circumvent the necessity for a fixed region size, our key idea is to consider multiple scales in the input space. This entails utilizing regions of different sizes and evaluating the calibration per region size. This can be regarded as an analogy to the wavelet transform, which enables analysis across multiple time and frequency resolutions [37].

V. Candidate Region Identification

In order to efficiently manage multiple input dimensions and provide compact region representations across various scales, we propose the use of ball trees [10]. This proposed scheme is depicted in Fig. 2.

A ball tree is a space partitioning data structure where partitions are represented as hyperspheres [10]. Each hypersphere has a center point $\underline{x}_\mathrm{c}$ and a radius r, and these hyperspheres are organized in a binary tree structure. The mean of all input test data defines the root node in the ball tree, and the maximum distance from the mean defines the radius. To create the tree structure, each node is split into two child nodes if the number of data points per node exceeds the user-defined maximum leaf node size $M \in \mathbb{N}_{>0}$. In the context of testing, the maximum leaf node size M can be used to ensure that the minimum number of data points per resulting leaf node partition, which are regarded as candidate regions, is met. For balanced trees, the actual leaf node size falls within the range of $M/2 \leq m \leq M$. Thus, the value M should be

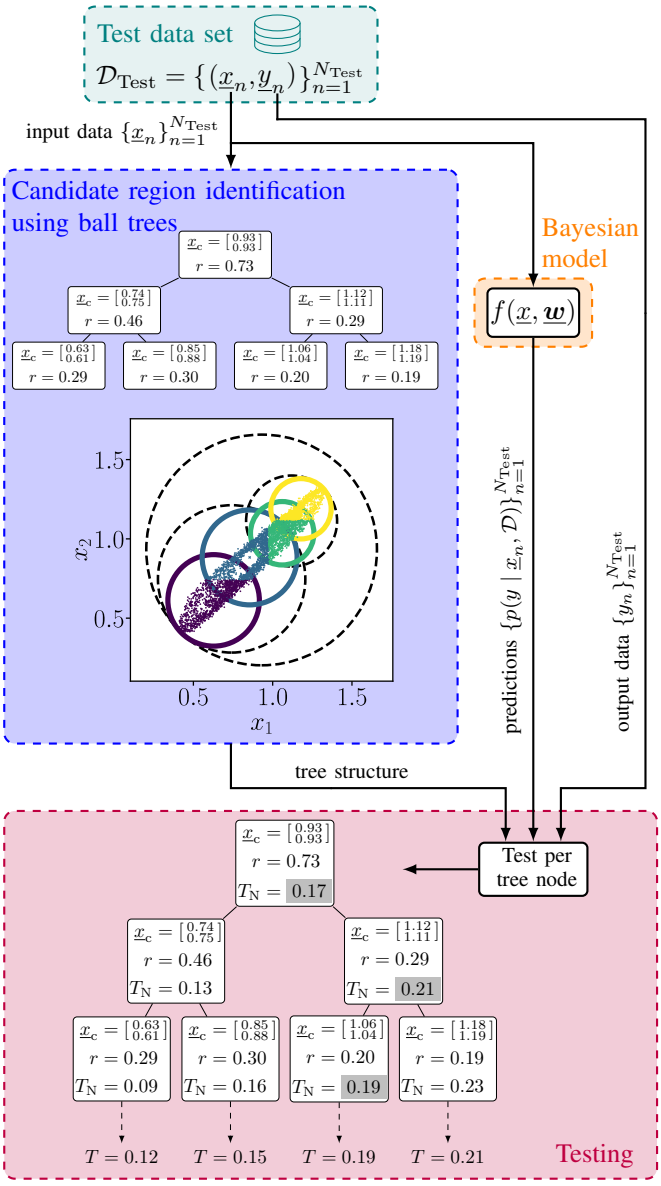
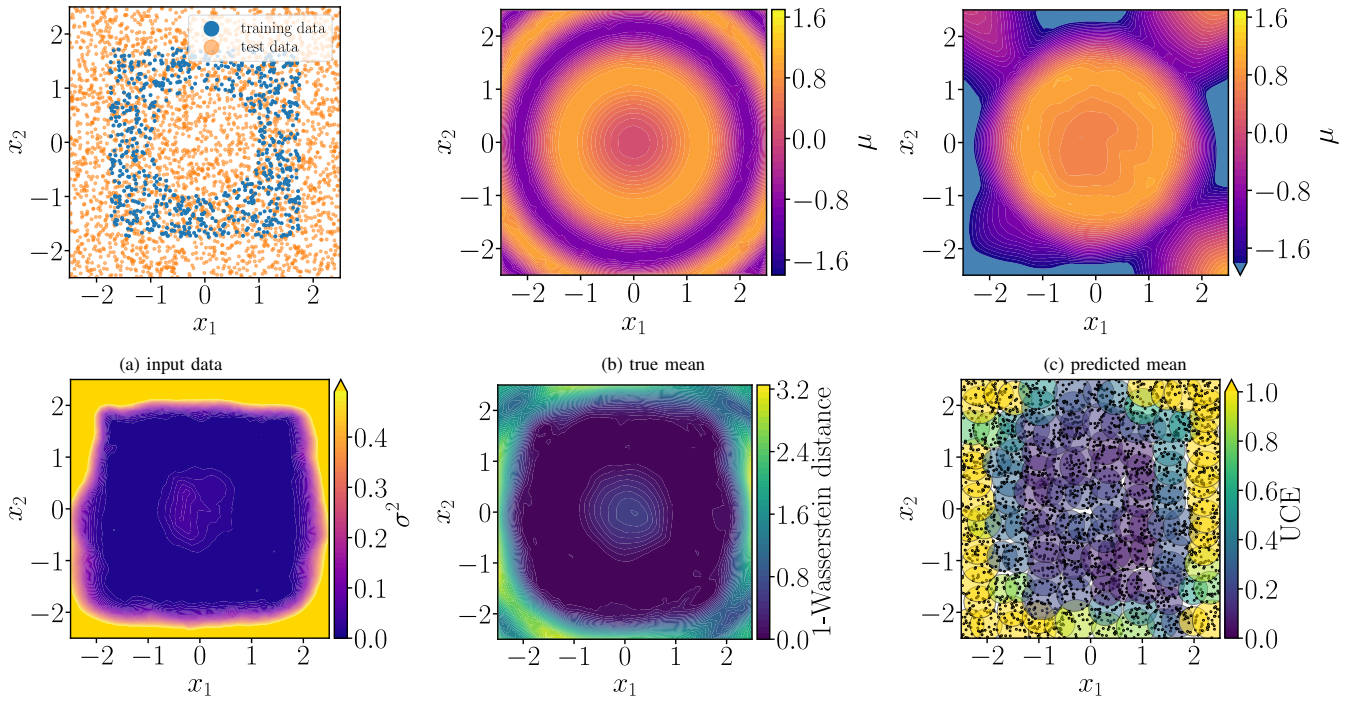


Fig. 2: Illustration of our proposed twofold scheme for testing the quality of predictions from Bayesian models. The first step consists of partitioning the input space using ball trees, and the second step assesses the quality of the predictions per node using a test metric. The combined test statistic T is calculated according to (4) using all statistical values on a root-to-leaf path, which is highlighted in gray as an example of a path.

set to be twice as much as the minimum required number of data points in order to obtain meaningful test results.

The process of splitting into child nodes is accomplished by a hyperplane that serves to separate the data belonging to the parent node into two subsets. After splitting, each child node is represented as the mean of a subset as a center point $\underline{x}_\mathrm{c}$, and the maximal distance of all points in a subset from the center point defines the radius r. E.g., when using the k-d tree construction algorithm [10], the splitting hyperplanes are selected to orthogonally intersect the main axis with the widest spread in data.



(a) input data (b) true mean (c) predicted mean

(d) predicted variance (e) 1-Wasserstein distance (f) combined test statistics

Fig. 3: Results of the Nonlinear Regression Example. The test and training data used in this example are displayed in (a), while the true mean of the data-generating process is shown in (b). It should be noted that the true variance remains constant at 0.1 and is not displayed. The distributions predicted by the BNN are shown in (c) and (d). The 1-Wasserstein distances between the true data-generating process are presented in (e), while the results of our proposed testing method are shown in (f) using the UCE as a calibration measure.

Regardless of the specific construction algorithm employed, the fundamental concept remains consistent: a tree structure is generated, and all hyperspheres on a path from the root to the leaf represent multiple scales per candidate region that can be evaluated.

VI. Multi-Scale Testing

Given the partitioned input space represented by the tree nodes, which are hyperspheres, the predictions and test data points within those regions are compared e.g., by calibration measures such as those described in Sec. II-B. The combined test statistics over multiple scales are obtained by

$$T = \sum_{k=1}^{K} w_{\mathrm{N}}(r_k) \cdot T_{\mathrm{N}}(\underline{x}_{\mathrm{c},k}, r_k) \ , \tag{4}$$

where $w_{\mathrm{N}}(r_k) > 0, \forall k$ represents the scale-dependent weight and is normalized such that $\sum_{k=1}^{K} w_{\mathrm{N}}(r_k) = 1$. The test statistic $T_{\mathrm{N}}(\underline{x}_{\mathrm{c},k}, r_k)$ is that of a single node on the root-to-leaf-node path. In this context, the weights determine the scales that should be prioritized for evaluating the quality of predictions. E.g., if the objective is to concentrate on local test statistics, one could select the unnormalized weights as $\tilde{w}_{\mathrm{N}}(r_k) = 1/r_k$, which is similar to the weighting function employed in localized cumulative distribution applications [38].

As a matter of fact, evaluating the root node only leads back to classical calibration measures that use the entire testing data set, as described in Sec. II-B. Therefore, our proposed method can be considered a generalized version of classical calibration testing.

VII. Experiments

We now demonstrate our proposed testing method on a nonlinear regression example and a binary classification example.

A. Nonlinear Regression

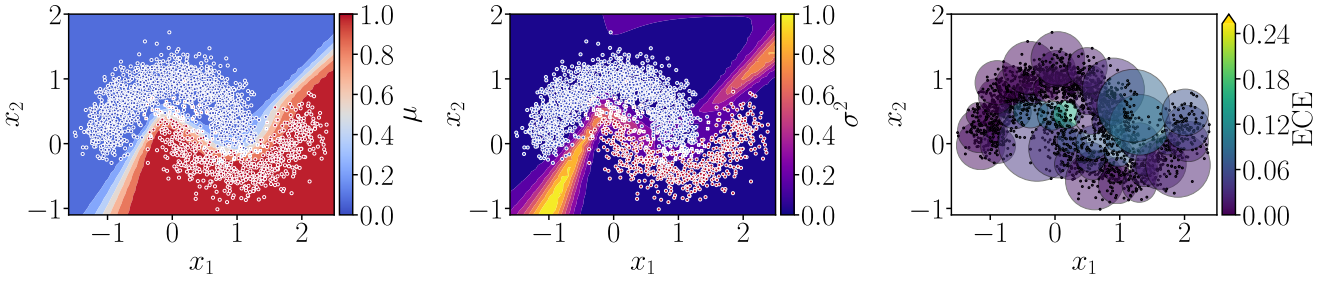
In the first experiment, we generate 1500 training points and 3000 test points from

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

with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 0.1)$, and $\underline{x} \in \mathbb{R}^2$. The training inputs $\underline{x}_n \in \mathcal{X}_{\mathrm{Train}}$ and the test inputs $\underline{x}_n \in \mathcal{X}_{\mathrm{Test}}$ are drawn uniformly from the two-dimensional input space $[-1.75, 1.75] \times [-1.75, 1.75]$ and $[-2.5, 2.5] \times [-2.5, 2.5]$, respectively. To simulate a gap in the training data, 30 % of the data around $\underline{x}^\top = [0, 0]$ is removed, as illustrated in Fig. 3a.

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 [15].

The predictions, the test results of our proposed method, and the 1-Wasserstein distance between the true data-generating process and the predictive distributions are presented in Fig. 3. The 1-Wasserstein distance between the normally distributed outputs of the true data-generating process $p(y \mid x_n) = \mathcal{N}\left(\mu_{\mathrm{True}}, \sigma^2_{\mathrm{True}}\right)$ and the normally distributed predictions $p(y \mid x_n, \mathcal{D}) = \mathcal{N}\left(\mu_{\mathrm{Pred}}, \sigma^2_{\mathrm{Pred}}\right)$ is



(a) predicted mean (b) predicted variance (c) combined test statistics

Fig. 4: Binary classification results with predicted mean (a) and variance (b), where dots mark the true classification of input test data and the incorrectly classified points are highlighted as crosses. The true labels are color-coded, with class label 0 displayed in blue and class label 1 in red. The combined test statistics per leaf node are displayed in (c).

given by [39], [40]

$$W_1 = |\mu_{\text{Dist}}| \left(1 - 2\Phi_{\mathcal{N}}\left(-\frac{|\mu_{\text{Dist}}|}{|\sigma_{\text{Dist}}|}\right)\right) + |\sigma_{\text{Dist}}| \sqrt{\frac{2}{\pi}} \exp\left(-\frac{\mu_{\text{Dist}}^2}{2\sigma_{\text{Dist}}^2}\right) ,$$

with $\mu_{\text{Dist}} = \mu_{\text{True}} - \mu_{\text{Pred}}$, $\sigma_{\text{Dist}}^2 = (\sigma_{\text{True}} - \sigma_{\text{Pred}})^2$, and standard normal cumulative distribution function $\Phi_{\mathcal{N}}(.)$. The ball tree was constructed using the k-d tree construction algorithm [10] and a maximum leaf size of $M = 40$. To assess the quality of predictions within each hypersphere, namely the nodes of the ball tree, the UCE (2), without the binning scheme, is employed as a calibration measure to compare the predictions with the test data. The combined test statistic is calculated using the unnormalized weights $\tilde{w}_{\text{N}}(r_k) = 1/r_k$.

By comparing the 1-Wasserstein distance between the true data-generating process and the predictions, it can be seen that the model has effectively learned the data-generating process, where training data is available, as evidenced by the near-zero 1-Wasserstein distances. In the outer regions ($|x_1|, |x_2| > 1.75$), where no training data is available, the predictions deviate significantly from the ground truth, which is correctly determined by our method. In the center, there is a slight increase in the variances around the origin where data is missing. While the predictions are not accurate, they are less erroneous than in the outer regions, as evidenced by the 1-Wasserstein distances. This finding is consistent with the results of our method, which indicate that the central region has lower combined statistics than the outer regions.

B. Binary Classification

As a second example, we consider a binary classification problem, where the data is generated over the two-dimensional input space $\underline{x} \in \mathbb{R}^2$ from the moon data set [41]. The data set is imbalanced, with 2000 data points belonging to class 0 and 1000 data points belonging to class 1. The noise in the data is set to $\sigma = 0.2$. The test data comprises 75 % of the data set.

The KBNN [21] is employed with two fully connected hidden layers, each comprising five neurons. The hidden units utilize a ReLU activation function, while the output layer employs a sigmoid activation function. Fig. 4 illustrates the input space and the corresponding local calibration, evaluated using the ECE (1), without dividing the test data into bins, in the combined test statistics (4). Again, the k-d tree construction is employed, with the maximum leaf node size set to $M = 40$. This results in a tree depth of eight, i.e., eight node statistic values are used to obtain one weighted statistic of a leaf node.

The comparison of the mean predictions with the true labels in Fig. 4a reveals incorrect classifications where both moons intersect. At the predicted separation between both classes, variance is increasing as illustrated in Fig. 4b. However, variance is also almost zero in the upper left and lower right corners where no data was available during training and testing. Consequently, in those regions, the predictions are considerably more certain than anticipated due to the lack of data. The combined statistical data is presented Fig. 4c. It can be observed that the combined statistic values along the separation of both classes are higher than in the region around $\underline{x}^\top = [0, 1]$, for example. This reflects the incorrect classifications that occur along the separation of both classes, thereby accurately reflecting classification errors in the combined statistic.

C. Discussion

The proposed method utilizing test data points employs a two-step testing strategy. Initially, the input space is partitioned to identify regions. Subsequently, these regions are subjected to testing. The region identification process is capable of dividing the input space into regions of varying scales, each characterized by a radius of a hypersphere. Furthermore, the method is capable of combining results across different scales, thereby providing insights into the local quality of predictions. A comparison of the 1-Wasserstein distance between the predictions and the known ground truth of the synthetic example in Fig. 3 and the true classification labels in Fig. 4 demonstrates that the proposed method is capable of identifying input-space regions that result in miscalibrated predictions.

Note that local testing in regions of the input space requires a larger amount of testing data than standard testing procedures. This can be seen, e.g., in the second experiment, where 75 % of the data is employed as test data. However,

this is an inherent feature of the region-based methodology, which requires substantial quantities of data to achieve high-resolution analysis within an input space.

VIII. Conclusion

In conclusion, this paper presents a novel testing method for multi-input systems that assesses prediction quality by testing in specific regions of the input space. This method is versatile, allowing for the combination of different scales of evaluation using an arbitrary calibration measure. A key contribution of our work is the provision of local calibration information, which is particularly useful when trustworthy models are of interest. This local calibration information provides a detailed understanding of the model's performance and trustworthiness in different regions of the input space.

Looking ahead, there are several promising directions for future research. One such direction is the use of local quality of predictions in applications such as state estimation and optimal control. Additionally, refining predictions in active learning settings presents an exciting area of investigation.

References

[1] Y. Gal, R. McAllister, and C. E. Rasmussen, "Improving PILCO with Bayesian neural network dynamics models," in *Data-efficient machine learning workshop, ICML*, Apr. 2016.

[2] F. Fiedler and S. Lucia, "Model predictive control with neural network system model and Bayesian last layer trust regions," in *2022 IEEE 17th International Conference on Control & Automation (ICCA)*, Naples, Italy, Jun. 2022, pp. 141–147.

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

[6] 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.

[7] 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.

[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] ——, "Trustworthy Bayesian Perceptrons," in *Proceedings of the 27th International Conference on Information Fusion (Fusion 2024)*, Venice, Italy, July 2024.

[10] S. M. Omohundro, "Five balltree construction algorithms," International Computer Science Institute, Tech. Rep. 89-063, Dec. 1989.

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

[12] 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.

[13] S. Duane *et al.*, "Hybrid Monte Carlo," *Physics Letters B*, vol. 195, no. 2, pp. 216–222, 1987.

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

[15] 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.

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

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

[18] A. Wu *et al.*, "Deterministic Variational Inference for Robust Bayesian Neural Networks," in *International Conference on Learning Representations*, 2019.

[19] 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.

[20] 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.

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

[24] N. Srivastava *et al.*, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[25] 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.

[26] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6405–6416.

[27] I. Harris, "Predictive Fit for Natural Exponential Functions," *Biometrika*, vol. 76, pp. 675–684, 1989.

[28] T. Fushiki, F. Komaki, and K. Aihara, "Nonparametric bootstrap prediction," *Bernoulli*, vol. 11, no. 2, pp. 293–307, 2005.

[29] 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.

[30] 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.

[31] 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.

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

[33] 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.

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

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

[36] 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.

[37] A. Cohen, *Numerical analysis of wavelet methods*, ser. Studies in mathematics and its applications. Elsevier,, 2003, vol. 32.

[38] U. D. Hanebeck, "Optimal Reduction of Multivariate Dirac Mixture Densities," *at - Automatisierungstechnik*, vol. 63, no. 4, pp. 265–278, Apr. 2015.

[39] M. Tsagris, C. Beneki, and H. Hassani, "On the Folded Normal Distribution," *Mathematics*, vol. 2, no. 1, pp. 12–28, Mar. 2014.

[40] S. Chhachhi and F. Teng, "On the 1-Wasserstein Distance between Location-Scale Distributions and the Effect of Differential Privacy," Apr. 2023, arXiv preprint arXiv:2304.14869.

[41] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *The Journal of Machine Learning Research*, vol. 12, no. null, pp. 2825–2830, Nov. 2011.