Proceedings of Machine Learning Research 230:1–30, 2024 Conformal and Probabilistic Prediction with Applications

Uncertainty Quantification for Metamodels

Martin Okánik MARTIN.OKANIK@TNO.COM
Athanasios Trantas THANASIS.TRANTAS@TNO.NL
Merijn Pepijn de Bakker MERIJN.DEBAKKER@PROTONMAIL.COM
Elena Lazovik ELENA.LAZOVIK@TNO.NL
Anna van Buerenplein 1, Den Haag, 2595 DA, The Netherlands

Editor: Simone Vantini, Matteo Fontana, Aldo Solari, Henrik Boström and Lars Carlsson

Abstract

In the realm of computational science, metamodels serve as indispensable tools for approximating complex systems, facilitating the exploration of scenarios where traditional modelling may prove computationally infeasible. However, the inherent uncertainties within these metamodels, particularly those driven by Machine Learning (ML), necessitate rigorous quantification to ensure reliability and robustness in decision-making processes. One alternative of obtaining uncertainty estimates is using ML models that have a native notion of uncertainty, such as the Bayesian Neural Networks (BNNs), however its repeated sampling necessary to approximate the output distribution is computationally demanding and might defeat the purpose of building metamodels in the first place. In datasets with multidimensional input space and a limited amount of training examples, error estimates provided by BNNs often have poor quality. This study explores alternative empirical approaches to uncertainty quantification, based on knowledge extraction from output space as opposed to input space. Leveraging patterns of magnitude of error committed by the metamodel in output space, we obtain significant improvement of adaptivity of prediction intervals, both over pure Conformal Prediction (CP) and BNNs. Our findings underscore the potential of integrating diverse uncertainty quantification methods to fortify reliability of metamodels, highlighting their robust and quantifiable confidence in model predictions.
Keywords: Uncertainty Quantification, Metamodels, Surrogating, Conformal Prediction, Modelling

1. Introduction

The emergence of Machine Learning (ML) and specifically Deep Learning (DL) has significantly advanced computational modelling, enabling analysis and prediction in diverse domains. Metamodels, also known as surrogate models, are computationally cheaper models designed to approximate the dominant features of a complex model. They are widely used whenever direct simulations are impractical. Their accuracy is crucial for decision-making, given the significant implications of prediction uncertainties stemming from data noise, model assumptions, and algorithmic limitations. Addressing this challenge requires robust uncertainty quantification (UQ) methodologies to provide a statistically rigorous foundation for model predictions.

UQ in metamodels involves identifying, characterizing, and managing uncertainties in inputs, parameters, and structure to ensure prediction confidence, risk understanding, and model refinement guidance. Two primary uncertainty types are aleatoric (inherent variability) and epistemic (model and parameter uncertainty). Aleatoric uncertainty is quantifiable

© 2024 M. Okánik, A. Trantas, M.P.d. Bakker & E. Lazovik.

using statistical methods, reflecting system randomness. Epistemic uncertainty, stemming from unknown model or parameter structures, is more challenging to estimate. This requires techniques like Bayesian inference, sensitivity analysis and ensemble methods (Abdar et al., 2021).

A case of a robust UQ framework that guarantees valid prediction intervals without precise data distribution assumptions is Conformal Prediction (CP) (Vovk et al., 2005). CP's model-agnostic nature suits integration with DL models, known for complex, nonlinear relationships and overfitting tendencies.

DL models, with their hierarchical structures and high-dimensional data processing, have transformed image recognition, language processing, and predictive analytics (Goodfellow et al., 2016). However, their "black-box" nature and sensitivity to data quality necessitate effective UQ. Techniques like Multi-Layered Perceptrons (MLPs), Bayesian Neural Networks (BNNs), and ensemble methods show promise but often require intricate modifications to model architecture or training.

Integrating CP with DL models offers a novel approach to UQ, leveraging CP's distribution-free, model-agnostic attributes for reliable prediction intervals. This integration enhances ML model interpretability and trustworthiness, paving the way for research in adaptive UQ methods.

The next section discusses UQ in metamodels and how our study addresses evolving field demands, section 4 describes the methodology, section 5 the results, and section 6 discusses and summarizes these.

2. Related Work

This section introduces the aim and background of the two main axes of this study.

2.1. Surrogate models

Historically, surrogate models were often derived from simplifying fundamental equations governing a phenomenon. Over time data-driven methods gained prominence, e.g. Gaussian Process Regression (GPR) (Sacks et al., 1989). Ensemble methods like random forests (Breiman, 2001), and boosting (Zhou et al., 2002) were introduced in early 2000's.

Modern DL models are gaining more and more traction in surrogate modelling (SM) thanks to the increase of computational capabilities in recent years, and an explosive growth of available training data. Such surrogates are often demanding to train, while still having significantly smaller execution time than their target models.

The usecase (GRASSMIND) under study, introduced in section 3, is an agent-based model (ABM). Surrogating ABMs is not a new idea. (van der Hoog, 2019) proposed to use ANNs as computational emulators of entire ABMs. A particular use of ML-based surrogates is sensitivity analysis of ABMs (Ten Broeke et al., 2021). Their direct use in such setting is hindered by relatively long inference time and high dimensionality of most ABMs, since it requires extensive scanning of input space. (Perumal and van Zyl, 2020) and (Lamperti et al., 2018) performed parameter calibration of ABMs using various ML-based SMs. A relatively comprehensive analysis of ML SM for ABMs was provided by (Angione et al., 2022). Surrogates used were of various complexities, ranging from linear regression,

through trees, gradient-boosted trees, nearest neighbours, GPRs, SVMs, ANNs. Their results suggest that ANNs and boosted trees significantly outperform other algorithms.

Several challenges persist in SM. Curse of dimensionality (Altman and Krzywinski, 2018) in input space limits both the computational savings, and in some cases also accuracy via overfitting, (Marrel et al., 2008). Lack of quality and quantity of training data is an obstacle that even the best surrogating algorithm cannot overcome, and is made worse by higher dimensionality of input space. On the other hand, too much data in regions of input parameter space that do not see significant deviations creates redundancy and makes SM infeasible because of a prohibitive training cost. (Constantine et al., 2014) delved into adaptive sampling strategies to address challenges of non-uniform sensitivity in parameter spaces. Modern adaptive sampling algorithms such as (fuzzy) LOLA-Voronoi (Crombecq et al., 2011), (van der Herten et al., 2015), CV-Voronoi (Xu et al., 2014), (Xu et al., 2014) (multi-task), can significantly reduce the computational cost of surrogate training.

A specific challenge in SM involves providing an affordable and flexible UQ technique.

2.2. Uncertainty quantification

The quest for robust UQ in ML, particularly within DL paradigms, has been a focal point of research, given its critical importance in ensuring the reliability and interpretability of model predictions. Early works primarily focused on Bayesian approaches, which introduce probabilistic interpretations of model parameters to quantify uncertainty. Notable among these is the work by (MacKay, 1992a,b) and (Neal, 1996), who pioneered the use of BNNs to provide principled UQ by integrating over model weights.

Ensemble methods (Dietterich, 2000) estimate model uncertainty through the variability in predictions across multiple models trained on the same task. This approach, while computationally more intensive, offers a practical way to capture both epistemic and aleatoric uncertainties without significant alterations to the underlying model architecture.

CP introduced by (Vovk et al., 2005), provides a distribution-free and model-agnostic UQ framework. Its ability to generate prediction intervals with a guaranteed coverage probability, as further elaborated by (Shafer and Vovk, 2008), has made it an attractive option for complementing traditional ML/DL models.

Recent advancements have seen the integration of CP with DL to address the limitations of conventional UQ methods. (Papadopoulos et al., 2011) demonstrated the applicability of CP in providing reliable prediction intervals for ANN predictions, highlighting its potential in enhancing model trustworthiness. This line of research has been extended by works such as (Angelopoulos et al., 2020), who explored adaptive CP methods tailored for complex data distributions typical of DL applications.

Exploration of CP in the context of invariant and adaptive UQ, as suggested by (Johansson et al., 2018), opens new avenues for research. By shifting the focus from input space to output space correlations, these studies propose innovative approaches to UQ that better align with the dynamic and complex nature of deep learning model predictions.

Recently, more and more emphasis is given on adaptivity of UQ, i.e. making the width of prediction intervals follow the difficulty of data points. Several methods were developed which modify loss functions of ML models, e.g. Nix and Weigend (1994), Pearce et al. (2018), Tagasovska and Lopez-Paz (2018), Kuleshov et al. (2018). A substantial recent

development was (Kivaranovic et al., 2020) which proposed a modified loss function for an ANN, inspired by the standard quantile loss. Even more recently, (Feldman et al., 2021) proposed orthogonal quantile regression. This involves a correction term to the loss function that promotes independence between estimated interval length and an indicator of miscoverage.

2.3. Motivation and aim of this study

In SM inference time is a crucial consideration, since its drastic reduction is the main reason of surrogating in the first place. ANNs and boosted trees rank among the most capable function emulators to serve as base of SMs at a still acceptable inference time. Their shortcoming is that as a result of their deterministic nature they cannot produce prediction intervals, while numerous much more computationally expensive algorithms can.

The primary aim of this study is to provide a computationally cheap empirical UQ method that enhances adaptivity of prediction intervals based solely on information from (surrogate) output space. This method reflects the needs of SM: it is usable for already trained and running surrogates with no need of their modification, and its decoupling from the many-dimensional input space makes it suitable for real-life datasets with limited size.

A secondary aim of this study is to surrogate GRASSMIND, an ABM that simulates the collective dynamics of plant growth as a function of ecological, biochemical and geometrical variables. The GRASSMIND dataset, further described in section 3, will be used as the model dataset in all of this study. Ideas on UQ presented here originally emerged from the need to provide a way of cheaply generating valid and approximately adaptive prediction intervals on top of a cheap deterministic metamodel of GRASSMIND.

3. Usecase Overview

3.1. BioDT Project

The BioDT project exemplifies the EU's commitment to utilizing Digital Twin technology for biodiversity preservation (BioDT, 2022). It integrates existing technologies to model species-environment interactions comprehensively, aiming to improve biodiversity monitoring and predictive capacities (Trantas et al., 2023). This initiative also supports the establishment of large-scale infrastructures to address biodiversity challenges, aligning with key environmental strategies such as the European Biodiversity Strategy 2030 and DestinE initiative (Nativi et al., 2021).

3.2. GRASSMIND

The model used in this study is GRASSMIND, an environmental ABM that simulates grasslands and combines biogeochemical cycles with biodiversity (Taubert et al., 2012). It includes additional submodels for plant-soil feedbacks, management and climate change. It also describes the competition for sunlight in the form of plant geometry and density on a given patch of land and the population dynamics (growth, senesence, death).

3.3. Dataset

Running a GRASSMIND simulation requires the GRASSMIND executable to use a parameter file containing values for a large number of model parameters. A GRASSMIND run is associated with some additional files, such as setting files for climate and land management strategies. After each run, GRASSMIND writes the model simulation results to a file. This means that the surrogating procedure requires code for the writing and reading of several files iteratively.

For generating the dataset for this study, the GRASSMIND model is run stochastically (seed of 50) for four plant functional types. After each run, the results of the four plant functional types are summed, while there is no aggregation over the stochastic runs. A Latin Hypercube Sampler picks the random parameter settings. This implies that for each LHS set of parameters, the model is run 50 times.

The total size of such dataset is 12500. For the purposes of UQ, in particular the resolution of variations of coverage in various subsets of the test set, a large test set is needed. This is the reason why a train-test split ratio of 1:1 was adopted. Sensitivity analyses to different split seeds were conducted, and no significant differences were found.

Tasks, labelled by index i ranging from 0 to 11, have two distinct groups. Tasks $i \in \{0, 1\}$ are the "spikiest" (i.e. longer tail of their distribution). These will be referred to as *group A*. The remaining tasks $i \in \{2, 3, 4, ..., 11\}$ have fewer outliers and will turn out as easier to predict. They will be referred to as *group B*. Figure 1 illustrates the difference between groups A and B in a (logarithmic) histogram, while all the task distributions can be found in Appendix D. All output tasks, as well as all input values, are standardized to zero mean and unit standard deviation. Group A has smaller median absolute values on both sides of zero, but only a slowly decaying tail at higher values roughly above 5.

4. Methodology

This section contains the methodology behind the two main components of this research. First, the ML-based surrogate models used to fit GRASSMIND are introduced, along with the criteria based on which they are assessed. After that follows the empirical UQ technique we developed to obtain adaptive width of prediction intervals to accompany mean model predictions, including how the degree of such adaptivity was diagnosed and improved. The notation used throughout this work is described in Appendix A.

The division of this paper into "Surrogate models" and "Uncertainty quantification" is not arbitrary. Surrogating GRASSMIND is the less important result. However in the second part, the comparision of adaptivity of the different surrogates serves as a case study for our proposed UQ scheme. This is in particular the MLP with our proposed UQ scheme, and the BNN with its native intervals.

4.1. Surrogate models

4.1.1. Surrogate model architectures

Several ML algorithms were tested, with three found to be the most promising and were selected for further development in surrogating GRASSMIND:

1. **Multi-layer perceptron (MLP)**; i.e. the simplest ANN, is the baseline algorithm in this study, that has been shown to work well on non-sequential multi-dimensional problems like ours (Baum, 1988).
2. **Bayesian Neural Network (BNN)** was chosen as a well-established representative of probabilistic neural networks (PNNs) (Jospin et al., 2022a). For UQ purposes, BNN will serve as an important reference point for testing our empirical approach.
3. **Gradient-boosted trees (XGB)** was chosen not only because of the promising performance metrics, but also because of its fundamentally different nature compared to the ANNs, providing a useful independent reference (Friedman, 2001).

Unlike the MLP and XGB models, a BNN prediction includes an uncertainty estimate. This is achieved using Monte Carlo methods (Metropolis and Ulam, 1949), by sampling the distributions of the learnt BNN parameters, in order to produce a sample of predicted values, from which the properties of the underlying output distribution can be estimated. In this work, $N_{\mathrm{MC}} = 1000$ draws are used to estimate this distribution. In practice only the 90% confidence interval is used, which is estimated as the quantile interval [0.05, 0.95].

A multitude of models were trained based on these algorithms, and their performance is later illustrated with some selected examples. These are formed from a series of three MLPs named small (s), medium (m), large (l), and three corresponding BNNs derived from these. Derived in this case means that they have the same structure of neural layers. Note that the notation "MLP-s" for MLP-small should not be confused with the abbreviated plural "MLPs". Using notation $N_{(0)} - N_{(1)} - ... - N_{(L-1)}$ where the N values represent the number of neurons in the L hidden layers, the s, m, l architectures are as follows: 64-32, 64-128-64, 128-256-512-256-128. Input and output layers always have sizes 16 and 12, corresponding to the number of features and tasks. For XGB, which is only used as reference, only one model configuration is reported.

As follows from section 1 and 2, the main trade-off to consider in SM is the one between fit quality and computational cost. The desired position on this imaginary scale depends on the particular use case, and no definite value is provided in this study, since at this point concrete user specifications for a surrogate of GRASSMIND do not exist. The two components of this trade-off will be addressed in the following two sections.

4.1.2. Performance metrics

To quantify the degree of alignment of the (mean) surrogate model output and the target data, four standard metrics were used: R^2, RMSE, MAE, MAPE, see details in Appendix C. In the case of BNNs, the predictive coverage of the 90% confidence interval returned by the BNN was tracked as well. This coverage represents the fraction of test set points that landed within their respective confidence interval. All the equations for metrics are applied separately for each task. For each metric, the most important indicator is the arithmetic mean of the 12-dimensional vector of task-wise values.

4.1.3. Computational cost

Computational cost is influenced by surrogate model parameters: batch size, learning rate, number of epochs (training time) and the algorithm and architecture (both training and

inference time). It is the *inference* time that is of main interest here rather than the *training* time. This is because in the intended future use case, the surrogate model is primarily used in the offline setting, only trained once and then applied many times to new data from the same data generating process (i.e. GRASSMIND). Extension to the online case would cause the training time to also enter these considerations. Training and inference times are evaluated both on Central Processing Unit (CPU) and Graphics Processing Unit (GPU) parts of a mid-range cluster and more details can be found in Appendix F.

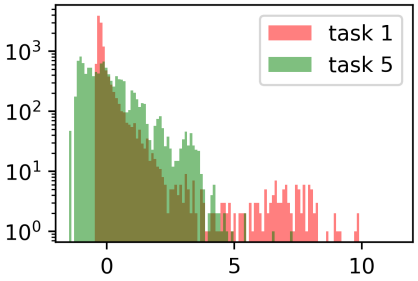


Figure 1: log-histogram of y of two example tasks from the dataset (section 3.3.

name	**dim**(p)	$f(y;p)$
unity	0	1
pmed	2	$1 + Ay_* + By_*^2$
p3med	3	$1 + Ay_* + By_*^2 + Cy_*^3$
ppmed	3	$(1 + Ay_* + By_*^2)^C$
pcmed	3	$1 + Ay_* + By_*^C$
plogmed	3	$1 + Ay_* + B\log(y_* + C)$

Table 1: Transformations used for scaling of prediction intervals based on output space in section 4.2.5.

4.2. Uncertainty quantification

The main aim of this study is to present an alternative method of generating partially adaptive prediction intervals on top of a (potentially deterministic) surrogate by using a combination of conformal prediction and leveraging information in output space of the surrogate.

4.2.1. Conformal Prediction

CP is placed at in the UQ pipeline to ensure marginal validity, in the sense that coverage of prediction intervals is equal to the desired coverage $1-\alpha$, α being the desired error rate, fixed to 10% throughout the study. Our test set left from surrogating has calibration:validation split of 4000:2250, i.e. 4000 data points are used for building the CP. (Vovk et al., 2005) provides a very robust treatment of various aspects of CP. For a quick first exposure, one can refer to (Angelopoulos and Bates, 2021), whose notation and ideas are used throughout this work. The parts relevant to this study are mentioned in Appendix B.

4.2.2. conformalising existing intervals

CP works by evaluating adequacy of existing prediction intervals, and making a correction in the direction based on the dominant sign in the score function distribution. However it does not *produce* intervals when they do not exist, this needs to be provided by another algorithm. In our study there are two options:

1. Surrogate only predicts the mean (MLP, XGB): in this case a first guess of error interval is generated in an arbitrary way, initially with no adaptivity (half-width δ

scalar). In our case we somewhat adapted the first-guess δ in a short algorithm involving successive multiplications/divisions to have a coverage within $\pm 10\%$ of $1-\alpha$.

2. Surrogate produces a distribution of outputs (BNN): in this case an interval is constructed from appropriate quantiles of this distribution, and is taken as the initial estimate. The interval is centered at the median of the output distribution and has a coverage of $1-\alpha$, to approximate the desired quantile interval $[\hat{t}_{\alpha/2}(x), \hat{t}_{1-\alpha/2}(x)]$.

In either case, estimates of prediction intervals can be expected to have a poor marginal coverage, that is why they are calibrated by CP. We used the score function

$$s(x, y) = \max\{\hat{t}_{\alpha/2}(x) - y, y - \hat{t}_{1-\alpha/2}(x)\}, \tag{1}$$

to build conformalised intervals (see Appendix B for details).

4.2.3. Adaptivity and conditional coverage

Marginal validity guarantee is in itself not enough to produce a *useful* prediction interval. *Conditional* validity requires that the coverage is not below $1-\alpha$ for any split of the dataset - be it random or purposeful. In mathematical terms:

$$\mathbb{P}(y \in C(x) \mid x) \geq 1 - \alpha, \tag{2}$$

for every x, y in the validation set. This ideal state can be approached by making the prediction intervals *adaptive* i.e. varying in width according to difficulty of the data point. Naturally, scalar intervals generated on top of an MLP by the method described in section 4.2.2 do not provide any adaptivity, and even when CP ensures a marginal validity with coverage $1-\alpha$, they cannot be expected to be conditionally valid. Stochastic algorithms like BNNs provide some degree of adaptivity since their intervals vary in size, although this is usually not enough by itself to approach conditional validity.

4.2.4. Evaluating adaptivity

Section 4.2.5 will describe how we leveraged the information from output space to introduce (MLP) or improve (BNN) adaptivity. First, tools are needed to quantify adaptivity, and how it varies across output space. This is introduced here. Let us denote the outputs of the i-th task as y (i.e. $y \equiv Y_i$). Our proposed approach is summarised in Algorithm 1.

The result of the third step (vectorized and applied through all tasks) is a $m \times N_b$ matrix C of coverage values of i-th task in b-th bin, which is then made into an m-dimensional vector of task-wise mean miscoverage values in step 4. Miscoverage in this case means a (possibly non-linear) metric describing the departure of coverage from its desired value. The scalar y_{median} denotes the median y-value of the given task. The dataset is normalized to zero mean and unit standard deviation, so the mean is zero, but not the median, since the y-distribution is strongly asymmetric. Specifically for the most spiky tasks $i = 0, 1$, the spikes only come in positive direction, making the median of y negative. It is postulated that this median behaviour is the default most easily learnt by the model, hence the sorting by $|y - y_{\text{median}}|$. This is an empirical design choice for this study. Similar approach based on y_{mean} instead of y_{median} was also attempted, with only minor difference in outcomes.

For each task $i = 0 \ldots (m-1)$ of the dataset separately:

1. Sort data points in ascending order by $|y - y_{\text{median}}|$
2. Divide into N_b bins of equal size $\frac{n}{N_b}$ datapoints per bin (or nearest integer)
3. Compute coverage separately for each bin
4. Compute a metric of average departure of coverage values of all bins from the desired coverage $1 - \alpha$

Algorithm 1: Evaluation of the coverage matrix

For the fourth step, we propose the following metric of miscoverage:

$$\text{RMSCM}_i = \sqrt{\frac{1}{N_b} \sum_{b=1}^{n_b} \xi_{ib} \left(C_{ib} - (1-\alpha)\right)^2}, \tag{3}$$

where the acronym stands for Root Mean-Squared Conditional Miscoverage. Its mathematical form resembles the formula for mean square distance (MSD). In principle, it is a modified MSD-like quantity where the point of reference is the desired coverage $1-\alpha$. Intuitively, RMSCM is related to the absolute "area" between a "curve" connecting the coverage values in particular bins, and a horizontal line at $1 - \alpha$, provided this "area" underwent a r.m.s. transformation. Another choice of metric could be based on a non-transformed area, but the square inside the sum in RMSCM ensures higher sensitivity to larger deviations, which we deem to be a desirable design feature.

For consistency with criterion from Equation 2, the matrix ξ_{ib} should take values:

$$\xi_{ib} = \begin{cases} 1, & C_{ib} < (1-\alpha) \\ 0, & C_{ib} > (1-\alpha) \end{cases} \tag{4}$$

However, for practical reasons we are not only interested in not going *below* $1 - \alpha$ (i.e. too small intervals), but also seek prediction intervals as small as possible, i.e. not going *above* $1 - \alpha$ either. Hence, we modify our criterion for conditional coverage by setting

$$\xi_{ib} = 1 \quad \forall C_{ib}. \tag{5}$$

This causes a slight departure from the established definition of conditional coverage provided in Equation 2, conceptually it amounts to replacing the $\geq$ sign by the $=$ sign, thus making the restriction stronger. This fulfils the stated goal of "usefulness" better than an interval which is forced to be above the threshold, but is too large most of the time.

The validation set of size 2250 was divided into 30 bins of size 75 each (step 2 of Algorithm 1), and coverage was calculated separately for each such bin. Such (mis)coverage curves will be shown in the result section. Equal size of coverage bins and relative scarcity of major peaks means that most of the interesting variation in $|y - y_{\text{med}}|$ is contained in the last few bins. Equal size of bins is still retained, because it is the most generally applicable algorithmic solution, and it allows for an easy visual interpretation when reading off a

graph. Also, all the data points are by themselves equally important as an objective for optimisation, and this setting reflects that.

It should be noted that making plots involving the coverage against some measure of output space is not an idea unique to this work, related plots can be seen in (Kivaranovic et al., 2020) or in (Feldman et al., 2021). However in this work, they are not only indicators, but in a way targets for optimizing adaptivity, since they visually express RMSCM.

4.2.5. Optimising adaptivity

With a metric of miscoverage in place (Equation 3), it is now possible to formulate an optimisation problem. Consider a general transformation function f scaling the prior prediction interval before applying CP. Let us have a model prediction for the i-th task y, and prior interval of half-width δ. The objective is to find the functional shape of f that minimizes RMSCM. f takes into account the value y itself, and some chosen parameter vector p, so that the transformed prior interval goes from $y \pm \delta$ to $y \pm f(y;p)\delta$. The coordinate actually used will not be y itself but a scaled version:

$$y_* \equiv \frac{|y - y_{\text{med}}|}{\langle |y| \rangle}, \tag{6}$$

where $\langle\rangle$ denotes the standard mean. This choice of y_* is motivated by an assumption that the model is making larger errors far from its median behaviour (c.f. MAE in Figure 5).

Transformations considered in this work are listed in Table 1. We opted for a very crude way of optimisation, that consisted of an exhaustive grid search through the parameter space defined by p, with typically 10-20 points in each dimension. This is justified a posteriori in section 5.

Below is a brief recapitulation of the algorithm. For each task separately:

1. Take $y \pm \delta$, y from model output, δ produced in any way applicable (c.f. 4.2.2)
2. Search for optimal parameter vector p in any way applicable (e.g. grid search)
3. From now on, use the optimal p for all predictions associated with this task

Algorithm 2: optimisation of the parameters of a transformation

It should be emphasised that for the same data-generating process (i.e. task) the optimal p only needs to be found once, provided that the calibration dataset is of appropriate size to represent the statistical distribution of peaks in sufficient detail.

4.2.6. Summary of research strategy

We produce surrogates of the mean GRASSMIND output using various implementations of MLPs, XGBs and BNNs, in the last case taking into account only the mean value of a BNN. The criteria against which we compare the surrogates are:

1. Overall picture provided by performance metrics, with main focus on RMSE
2. Inference time, to some extent training time

For UQ, we follow algorithm 2 to optimize the parameter vector p of transformations that introduce adaptivity into artificially generated prediction intervals of constant width. This serves to build a UQ framework to be used with deterministic surrogate models that only produce point predictions. We evaluate its usefulness by comparing it to adaptivity of prediction intervals obtained from a BNN:

1. MLP + guess-interval + transformation + CP
2. BNN + CP

here the notation "A + B" means "B sequentially follows A" in the pipeline. Our criteria against which the two schemes are compared are:

1. RMSCM (primary, since adaptivity is the goal)
2. Inference time (secondary, still much less than surrogate for mean)

5. Results

In this section the results of of SM and UQ are presented in two separate subsections.

5.1. Surrogate models

5.1.1. Performance metrics

The performance metrics introduced in section 4.1.2 can be seen in Table 2.

score	R^2	**RMSE**	**MAE**	**MAPE**	**covg. 90%**
MLP-s	0.911	0.271	0.128	1.033	-
MLP-m	0.911	0.271	0.128	1.033	-
MLP-l	**0.921**	**0.246**	0.100	0.651	-
BNN-s	0.912	0.268	0.131	**0.925**	0.648
BNN-m	0.904	0.267	0.114	1.108	0.895
BNN-l	0.906	0.262	0.108	1.043	0.893
XGB	**0.923**	0.307	**0.083**	**0.519**	-

Table 2: Performance metrics for the seven models mentioned in the text (last column only meaningful for BNNs, since remaining algorithms only output point predictions).

The two important metrics for further discussion are RMSE and MAE. RMSE is more sensitive to a smaller number of large deviations compared to MAE, and since networks are somewhat better for the peaks, their RMSE is still better.

Note that R^2 values are not a good discriminator among models. MAPE shows very similar patterns to MAE, the only difference is that XGB performance is somewhat more enhanced compared to the network-based models. This is because it is able to fit the small y-values near perfectly, while networks still have non-negligible errors there. This is emphasized when relative errors are considered instead of absolute errors, hence the difference between XGB and networks is more pronounced between RMSE and MAPE compared to RMSE and MAE. For BNNs, the last column represents the (predictive) coverage of the 90%

confidence interval estimate, i.e. which proportion of test samples lies within the estimated [0.05, 0.95] quantiles of the BNN output distribution.

Also note that all these values are averaged across tasks, and might not be representative of any task separately. As an example, for BNN-large, RMSE (mean value 0.26) reaches 0.56 and 0.63 for tasks 0 and 1 of group A, while it ranges from 0.13-0.20 for all but two of the ten tasks from group B, the exceptions being 0.30, 0.37 for tasks 2 and 3.

5.1.2. Computational cost

We quantify the computational cost by the time it takes to evaluate 10^6 data points, taken per data point. This number 80 times larger than the whole dataset was obtained by stacking and shuffling it together 80 times, to get a larger total execution time which ensures that model execution is the dominant process that contributes to the time estimate. The device used was a GPU and more details can be found in Appendix F. These inference times for the seven models are given in Table 3. Note that for BNNs, this means evaluation of just one sample from the distribution. The real inference time would be larger by a factor N_{MC} representing the number of Monte Carlo samples taken from the distribution, in our case this was $N_{\mathrm{MC}} = 1000$. A single evaluation of XGB is much more expensive, since trees were not properly optimised for GPU, on a CPU it had the same order of magnitude as the ANNs. The main purpose of XGB in this study was to use it as a baseline for performance metrics, without expecting it to be actually deployed, hence the inference time is not considered important.

model	CPU (ns)	GPU (ns)
MLP-s	198	0.2
BNN-s	181	0.2
MLP-m	312	0.3
BNN-m	427	1.2
MLP-l	2160	0.7
BNN-l	1820	1.7
XGB	258	232

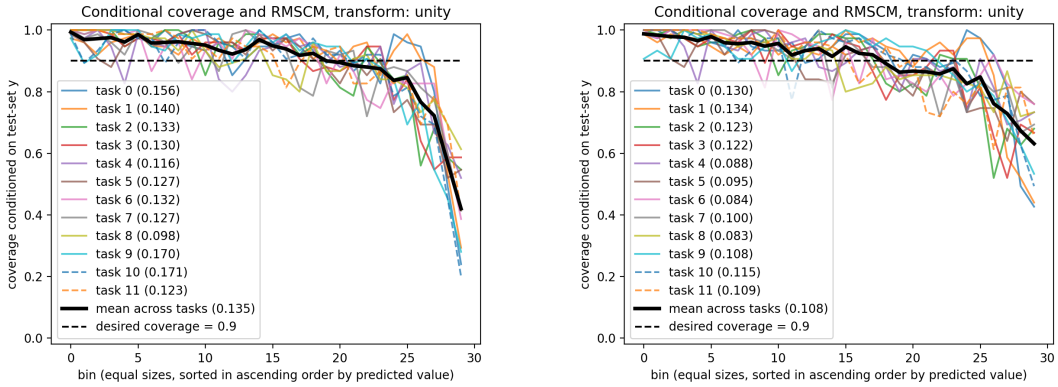
Table 3: CPU and GPU inference times (ns/datapoint).

transform	MLP-s	BNN-l
unity	0.135	0.108
plogmed	0.064	0.069
pcmedP	0.068	0.077

Table 4: RMSCM values for two models with different transformations.

5.2. Uncertainty quantification

This part displays how well our UQ approaches work. Examples given here are built on MLP-s and BNN-l, more specific, the cheapest MLP and the most expensive BNN. MLP is attributed an arbitrary scalar prediction interval, which is conformalised for correct marginal coverage, and transformations such as those in Table 1 are applied. For the BNN, the main interest is placed on how well it can perform with no transformations, just CP. The main question is, whether MLP with a suitable transformation can outperform much more expensive BNN, in terms of the adaptivity of produced prediction intervals.



(a) (b)

Figure 2: Conditional coverage for "MLP + guess-interval + CP" (left) and "BNN + CP" (right). The 12 tasks are plotted as solid or dashed thin coloured curves, while their average is the thick black curve. Bin 0 represents median, and bin 30 represents points furthest away from the median. Values in parentheses in the legend are task-wise RMSCM values.

5.2.1. Bare conformal prediction for MLP and BNN

Figure 2 (a) shows the dependence of coverage on size bin after a scalar initial guess of prediction interval was conformalised to have the desired marginal coverage $1 - \alpha = 0.9$; following our proposed method outlined in section 4.2.4. The trend of decreased coverage to the right is consistent with the observation that model makes systematically larger errors when it outputs large values, and the size of prediction interval needs to take this into account - but in its present form does not, and the aim here is to ensure this.

It should be noted that in Figure 2 and all subsequent similar figures, the thick black line (labeled "mean across tasks") represents the arithmetic mean of the 12 coloured task-wise curves, however the neighbouring RMSCM number in parentheses in the legend (in this case 0.135) does not correspond to this curve directly, but rather to the average of the separate task-wise RMSCM values shown in other entries in the legend. In other words, deviation of the mean is not equal to the mean of deviations, they are merely shown on the same line in the legend to save space. It should also be emphasized that there are only 75 samples in the individual bins, so estimate of coverage (a probability) is very noisy, and this is reflected in the discordant colourful curves corresponding to individual tasks in Figure 2. Taking average across tasks naturally suppresses this, but it is only a side effect, there is no reason for coverage values of different tasks in a particular bin to come from the same probability distribution which could be meaningfully averaged. The main of highlighting the task-mean curve is to show the collective tendency of the dataset as a whole in one curve instead of twelve, and it should not be attributed any literal meaning by itself.

BNNs should in principle be able to learn (some of) the factors in input space that make their predictions uncertain. One can repeat the same procedure for the conformalised output of a BNN, see Figure 2 on the right.

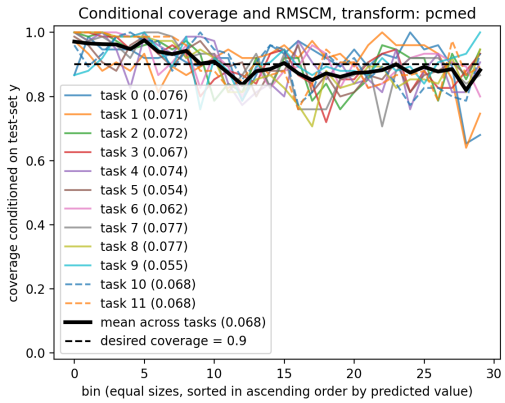


Figure 3: Conditional coverage for "MLP + guess-interval + transformation + CP" for the 12 tasks plotted as solid or dashed thin coloured curves, while their average is the thick black curve. Bin 0 represents median, and bin 30 represents points furthest away from the median. Values in parentheses in the legend are task-wise RMSCM values.

The task-mean RMSCM moderately decreased by going from MLP to BNN Figure 2 (b). While for "MLP + guess-interval + CP" it is 0.135, for "BNN + CP" it is 0.108 insignificantly lower value was reported with BNN of similar architecture with a simple ReLU activation function, namely 0.105. Qualitatively, it can be seen that the decrease of the coverage in high bin values (> 25) is made slower with BNN, which accounts for most of the decrease in RMSCM between MLP and BNN. Note that the improvement is quite modest for the two tasks in group A, only going from (0.156, 0.140) to (0.130, 0.134).

5.2.2. Conformal prediction with transformations

Our proposed strategy of leveraging correlations in output space as opposed to input space as described in section 4.2.5 is an alternative to using PNNs to achieve adaptive prediction intervals. It is implemented by extending and squeezing the prediction interval half-width δ based on a parametric function $f(y; p)$ of the mean prediction y. Performance of the setup "MLP + guess-interval + transformation + CP" is shown in Figure 3, to be contrasted with Figure 2 (top). Task-mean RMSCM fell by a half from 0.135 to 0.068, in an example when using the transformation with variable-power median-centered polynomial (pcmed). While this does not ensure perfect conditional coverage, the task-mean coverage stays always between 80% and 90% in all regions of the y-space. The same is true *nearly* all of the time for the task-wise coverage, which is naturally more spread than its mean. In particular the two most challenging tasks from group A, namely 0 and 1 now see almost as good RMSCM as the overall average, that is (0.076, 0.071). Transformation named "pcmed" shown in Figure 3 was one of the best performing transformations. Overall, each member of a 3-parameter family of transformations from Table 1 had a final RMSCM in range [0.64, 0.79], depending on particular surrogate model and transformation used. 2-parameter family, represented only by a simple quadratic fit, was significantly worse, reaching RMSCM values around 0.9 with different models - still better than the non-transformed BNN, and unlike

the BNN it completely prevented the steep decline of coverage in the last few bins (compare bins 25-30 in Figure 2 and Figure 3).

Seeking adaptivity by exploiting information included in both input *and* output spaces by combining BNNs (as opposed to MLPs) with transformations was also tried, but did not yield better results. In fact the RMSCM was slightly worse in such configuration. Values for the transformations are seen in Table 4.

6. Discussion and conclusion

6.1. Surrogate models

Conclusions of this study indicate that GRASSMIND can be cheaply surrogated with multiple accessible and widely used ML algorithms. Accuracy of such surrogate is rather low for some prediction tasks (broadly group A) and very high for the remaining tasks (broadly group B). Tasks in group A (i =0, 1) are also the spikiest, as was noted already from their histogram in Figure 1. It is notable that all algorithms have more-or-less similar ability (or lack thereof) to fit selected individually studied data points, especially the peaks in tasks 0 and 1, which are generally the hardest and least well fitted data in the entire dataset. This suggests that the remaining "unfitable" y is due to irreducible noise in target model (GRASSMIND) itself, and these variations are inherently unlearnable by the surrogate. This means that further improvement of the surrogate is unlikely to yield tangible benefits.

In this first phase, both deterministic (MLP, XGB) and probabilistic (BNN) algorithms were used, however the stochastic nature of BNNs was only leveraged indirectly through their capacity to potentially improve the mean prediction, see (Jospin et al., 2022b).

6.2. Uncertainty quantification

The second phase of this study introduced a method of generating adaptive prediction intervals, which only requires fitting a transformation function of a small number of parameters to capture the approximate relationship between the surrogate output and typical magnitude of committed error. This approach has several advantages:

1. **Computational resources**: error prediction on a new data point is as computationally cheap as evaluating the transformation function, which has only several parameters and can often be a simple polynomial. The computational expense of this procedure is usually insignificant compared to an ANN. Adding this on top of a surrogate thus provides UQ almost for free.

2. **Alternative to PNNs**: in this dataset, this approach outperforms a relatively large BNN in terms of generating valid prediction intervals. As an alternative to an expensive BNN which needs to be evaluated at least $\sim 10^2$ times to obtain good statistics, one can use a small MLP and leverage the correlations in output space to adapt the width of a scalar initial guess of the prediction interval.

3. **Error agnosticity**: one does not need to know the source of model error, the method naturally incorporates both the aleatoric and epistemic uncertainties, provided they are correlated with y.

4. **Input space agnosticity**: researcher does not need to know (or let a machine learn) anything about the input space in order to construct prediction intervals. This study used GRASSMIND as a black-box model, with no such knowledge.

5. **Model agnosticity**: it can be used on top of any surrogate model, even one not based on ML. All that is required is the paired datasets containing (i) output of the model and (ii) the test dataset, assumed as the ground truth (i.e. the output of the to-be-surrogated model).

6. **Model preservation**: it can be added on top of an already existing surrogate, with no need to modify loss function, re-calculate weights etc. It preserves a deterministic surrogate for prediction of the expected outcome exactly as it is, with no concern of changing these predictions after incorporating this UQ scheme. This is important for backward compatibility.

There are still some concerns and open questions remaining, that are addressed here:

1. **Generalisability**: this is the most obvious concern - how frequent or rare is it that a dataset fitted by a model posseses a strong-enough correlation between model error and model output, for it to be able to significantly improve conditional validity of the model? One could certainly collect a lot of datasets from various domains with their respective ML fits to argue either way, so we do not believe providing exhaustive examples would be a satisfactory justification. There are however some *fundamental* reasons why a reasonable generalisability can be expected. First, scientific models are often based on a theory that has some limits of validity, and the further the input values deviate from some inner range, the worse approximation the theory provides. This can easily have a mapping to the output space, when some parts of it correspond to the parts of input space of a good degree of approximation of reality, and some of a bad degree of approximation. A second reason is that many scientific models are constructed on empirical observations of the world, and outliers due to e.g. wrong measurements can also be expected to posses some y-dependence (e.g. big in magnitude). A third and fourth reason have nothing to do with the target model, but the surrogate model. When a surrogate is e.g. slightly underfitted, this also tends to show a y-dependent pattern of error. Also, large peaks are often infrequent and cannot be learnt so well, further increasing the error for certain values of y.

2. **Why not just look into *input* space**: Methods looking at output space will necessarily be more restricted than methods based on input space (e.g. BNNs), since a given output can be caused by multiple input values, and most often not vice versa. This fact blurs the overall relationship between y and data point difficulty compared to X. In many cases, such as ours, this principal restriction is more than counteracted by the ease of extracting information from output space. Knowledge extraction from output space can (in fact *needs to*) be done one output dimension at a time, which constitutes a one-dimensional problem. Most of interesting variance can thus be extracted very easily, with no need of advanced algorithms. This is in contrast to finding more complete patterns in many-dimensional input space, a task which is fundamentally not separable, and one often does not have enough data to fit a

probabilistic model (e.g. a BNN) whose intervals would fully reflect the difficulty of different regions of the complicated many-dimensional input space.

3. **When to use**: it is typically not possible to say *a priori* whether this method is useful for a given task, since it depends not only on the dataset, but also on how well the trained surrogate model fits particular properties of the dataset. A simple diagnostic such as a plot of the dependence of model MAE on a suitably chosen y-related metric (in our case y_* as in Equation 6) can provide a quick insight whether this approach should be attempted.

4. **Quality requirements for the design and fit of parameters of the transformation function**: transformations with vastly different functional shape lead to broadly similar aggregate RMSCM value (0.64-0.79), as well as comparable distribution among bins, i.e. the shape of the miscoverage curve. This suggests that they are all broadly able to characterise that part of variation of data point difficulty which is dependent on y_*, and the remaining RMSCM (let us say below ~ 0.6) cannot be lowered by even a better fitting function that works with y_*, and likely any other variable constructed from y only.

5. **Transformation before or after CP**: both approaches were tried, and produced broadly similar results. Describing best practices in designing this type of UQ pipeline are a task for future research.

6. **BNN + transform**: combining BNNs with transformations has not improved the results as quantified by the RMSCM. This is a surprising result, and should be investigated further.

Acknowledgements

This study has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101057437 (BioDT project, https://doi.org/10.3030/101057437). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

References

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2021.05.008. URL https://www.sciencedirect.com/science/article/pii/S1566253521001081.

Naomi Altman and Martin Krzywinski. The curse (s) of dimensionality. *Nat Methods*, 15 (6):399–400, 2018.

Anastasios Angelopoulos, Stephen Bates, Jitendra Malik, and Michael I Jordan. Uncertainty sets for image classifiers using conformal prediction. *arXiv preprint arXiv:2009.14193*, 2020.

Anastasios N Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

Claudio Angione, Eric Silverman, and Elisabeth Yaneske. Using machine learning as a surrogate model for agent-based simulations. *Plos one*, 17(2):e0263150, 2022.

Eric B Baum. On the capabilities of multilayer perceptrons. *Journal of Complexity*, 4(3): 193–215, 1988. ISSN 0885-064X. doi: https://doi.org/10.1016/0885-064X(88)90020-9. URL https://www.sciencedirect.com/science/article/pii/0885064X88900209.

BioDT. Biodiversity digital twin. https://biodt.eu/, 2022. Accessed: 15-1-2024.

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

Paul G Constantine, Eric Dow, and Qiqi Wang. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4): A1500–A1524, 2014.

Karel Crombecq, Dirk Gorissen, Dirk Deschrijver, and Tom Dhaene. A novel hybrid sequential design strategy for global surrogate modeling of computer experiments. *SIAM Journal on Scientific Computing*, 33(4):1948–1974, 2011.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

Shai Feldman, Stephen Bates, and Yaniv Romano. Improving conditional coverage via orthogonal quantile regression. *Advances in neural information processing systems*, 34: 2060–2071, 2021.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Ulf Johansson, Henrik Linusson, Tuve Löfström, and Henrik Boström. Interpretable regression trees using conformal prediction. *Expert Systems with Applications*, 97:394–404, 2018. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2017.12.041. URL https://www.sciencedirect.com/science/article/pii/S0957417417308588.

Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022a. doi: 10.1109/MCI.2022.3155327.

Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022b.

Danijel Kivaranovic, Kory D Johnson, and Hannes Leeb. Adaptive, distribution-free prediction intervals for deep networks. In *International Conference on Artificial Intelligence and Statistics*, pages 4346–4356. PMLR, 2020.

Ranganath Krishnan, Pi Esposito, and Mahesh Subedar. Bayesian-torch: Bayesian neural network layers for uncertainty estimation, January 2022. URL https://doi.org/10.5281/zenodo.5908307.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International conference on machine learning*, pages 2796–2804. PMLR, 2018.

Francesco Lamperti, Andrea Roventini, and Amir Sani. Agent-based model calibration using machine learning surrogates. *Journal of Economic Dynamics and Control*, 90:366–389, 2018.

P. Langley. Crafting papers on machine learning. In Pat Langley, editor, *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992a.

David John Cameron MacKay. *Bayesian methods for adaptive models*. California Institute of Technology, 1992b.

Amandine Marrel, Bertrand Iooss, François Van Dorpe, and Elena Volkova. An efficient methodology for modeling complex computer codes with gaussian processes. *Computational Statistics & Data Analysis*, 52(10):4731–4744, 2008.

Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

Stefano Nativi, Paolo Mazzetti, and Max Craglia. Digital ecosystems for developing digital twins of the earth: The destination earth case. *Remote Sensing*, 13(11), 2021. ISSN 2072-4292. doi: 10.3390/rs13112119. URL https://www.mdpi.com/2072-4292/13/11/2119.

Radford M. Neal. Priors for infinite networks. *Bayesian learning for neural networks*, pages 29–53, 1996.

David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

Harris Papadopoulos, Vladimir Vovk, and Alexander Gammerman. Regression conformal prediction with nearest neighbours. *Journal of Artificial Intelligence Research*, 40:815–840, 2011.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Tim Pearce, Alexandra Brintrup, Mohamed Zaki, and Andy Neely. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4075–4084. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/pearce18a.html.

Rylan Perumal and Terence L van Zyl. Surrogate assisted methods for the parameterisation of agent-based models. In *2020 7th International conference on soft computing & machine intelligence (ISCMI)*, pages 78–82. IEEE, 2020.

Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and Analysis of Computer Experiments. *Statistical Science*, 4(4):409 – 423, 1989. doi: 10.1214/ss/1177012413. URL https://doi.org/10.1214/ss/1177012413.

Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

Natasa Tagasovska and David Lopez-Paz. Frequentist uncertainty estimates for deep learning. *arXiv preprint arXiv:1811.00908*, 2018.

Franziska Taubert, Karin Frank, and Andreas Huth. A review of grassland models in the biofuel context. *Ecological Modelling*, 245:84–93, 2012.

Guus Ten Broeke, George van Voorn, Arend Ligtenberg, and Jaap Molenaar. The use of surrogate models to analyse agent-based models. *Journal of Artificial Societies and Social Simulation*, 24(2), 2021.

Athanasios Trantas, Ruduan Plug, Paolo Pileggi, and Elena Lazovik. Digital twin challenges in biodiversity modelling. *Ecological Informatics*, page 102357, 2023.

Joachim van der Herten, Ivo Couckuyt, Dirk Deschrijver, and Tom Dhaene. A fuzzy hybrid sequential design strategy for global surrogate modeling of high-dimensional computer experiments. *SIAM Journal on Scientific Computing*, 37(2):A1020–A1039, 2015.

Sander van der Hoog. Surrogate modelling in (and of) agent-based models: A prospectus. *Computational Economics*, 53(3):1245–1263, 2019.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

Shengli Xu, Haitao Liu, Xiaofang Wang, and Xiaomo Jiang. A robust error-pursuing sequential sampling approach for global metamodeling based on voronoi diagram and cross validation. *Journal of Mechanical Design*, 136(7):071009, 2014.

Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.

Appendix A. Notation

Notation in this article will be as follows: X and Y are the 16- and 12-dimensional input and output datasets represented as matrices, with the vector $y \equiv Y_i$ denoting output from the i-th task. More broadly, y will be used to mean the output variable, not always in reference to a particular array object. Lowercase n defines the number of data points in the given set, which depends on context (train/test/calibration/validation). Lowercase m is the number of output dimensions (tasks), $m = 12$ throughout this study. Index i is reserved for tasks, ranges from 0 to 11. Uppercase N is reserved for quantities that do not express number of data points, e.g. number of neural layers, number of bins, or number of samples drawn from a distribution. L is used for the number of hidden layers. $M(X)$ denotes the surrogate model output when being fed an input matrix X. α is always the desired error rate for UQ purposes, the desired coverage $1 - \alpha = 0.9$ in all results that we show here. δ is interval half-width along y.

Appendix B. Quick summary of conformal prediction

Conformal prediction (Vovk et al., 2005) was used in this study to produce prediction intervals to accompany the point predictions, or to calibrate the confidence intervals inferred from the models which do produce them (BNNs). CP calibrates intervals by using a separate unseen *calibration set* on which it evaluates a *score function*, which is a heuristic measure of model uncertainty. Using the model and the calibration data, one can construct a prediction interval $C(x)$ that is *valid* in the sense that:

$$1 - \alpha \leq \mathbb{P}(y \in C(x)) \leq 1 - \alpha + \frac{1}{n+1} \tag{7}$$

where y is the true value of a point x in the validation set (unseen by both the model and the CP), $1-\alpha$ is the desired coverage (α being the accepted error rate), and n the number of calibration examples. This property is called *marginal coverage* or "being marginally valid", and is guaranteed to hold globally, as long as the choice of score function is suitable. Loosely speaking, suitability means being able to rank different test examples according to their difficulty, e.g. using a metric derived from the error committed by the model.

Let us briefly introduce the procedure and the corresponding notation. Let there be an initial estimate of the prediction interval labeled as $[\hat{t}_{\alpha/2}(x), \hat{t}_{1-\alpha/2}(x)]$. This notation hints at the fact that it is constructed from symmetric quantile estimates that enclose an interval with coverage $1-\alpha$. Such interval could have been obtained by any method, and can in general be expected to be rather poor. CP is used to find an additive correction necessary to make this interval marginally valid. For this, a score function $s(x, y)$ is used, which expresses the directed deviation of the true label y from the nearest lower/upper boundary:

$$s(x, y) = \max\{\hat{t}_{\alpha/2}(x) - y, y - \hat{t}_{1-\alpha/2}(x)\}. \tag{8}$$

This functional form allows to express a correction needed to widen/compress the prior prediction interval if the latter turns out to provide lower/higher coverage than $1-\alpha$. The proper amount of correction is estimated as $\hat{q}$, the $1-\alpha$ quantile of s across the n data points of the calibration set. Here, it is expressed in its full form, with a statistical correction for finite-sample effects applied to the desired coverage $1-\alpha$:

$$\hat{q} = \text{Quantile}\left(s_1, ..., s_n; \frac{\lceil (n+1)(1-\alpha) \rceil}{n}\right). \tag{9}$$

Then the conformalised prediction interval is

$$C(x) = [\hat{t}_{\alpha/2}(x) - \hat{q}, \hat{t}_{1-\alpha/2}(x) + \hat{q}]. \tag{10}$$

Appendix C. Performance metrics

This appendix lists detailed formulas for the performance metrics used for the intercomparision of surrogate models in the main text.

1. Coefficient of determination (R^2) expresses the fraction of variance in the dataset explained by the model.

$$(R^2)_i = 1 - \frac{\sum_{j=0}^{n-1}(Y_{ij} - \hat{Y}_{ij})^2}{\sum_{j=0}^{n-1}(Y_{ij} - \bar{y}_i)^2} \tag{11}$$

   Here the index i $(0...m-1)$ labels the task, j $(0...n-1)$ the data point, Y ($\hat{Y}$) is the $m \times n$ matrix of ground truth (predicted) output values, and $\bar{y}$ is the m-dimensional vector of task-averages .

2. Root mean square error (RMSE)

$$(\text{RMSE})_i = \sqrt{\frac{1}{n}\sum_{j=0}^{n-1}(Y_{ij} - \hat{Y}_{ij})^2} \tag{12}$$

3. Mean absolute error (MAE): unlike RMSE, the cost is linear with respect to the error magnitude.

$$(\text{MAE})_i = \frac{1}{n}\sum_{j=0}^{n-1}|Y_{ij} - \hat{Y}_{ij}| \tag{13}$$

4. Mean absolute percentage error (MAPE) is a variant of MAPE when the error is considered in proportion to the magnitude of the true value.

$$(\text{MAPE})_i = \frac{1}{n}\sum_{j=0}^{n-1}\left|\frac{Y_{ij} - \hat{Y}_{ij}}{Y_{ij}}\right| \tag{14}$$

   An additional metric that applies only to probabilistic models, in this case BNNs, is the predictive coverage of the 90% confidence interval. This is the interval constructed from [0.05, 0.95] quantiles of the true output distribution, estimated as corresponding quantiles from $N_{\text{MC}} = 1000$ Monte Carlo samples of the BNN. It is another indication of how well the mean prediction reproduces the ground truth, but corrected for the amount of uncertainty that the model identifies in the given region of input space.

Appendix D. Task distributions

In analogy to Figure 1, Figure 4 shows the (logarithmic) distributions of all 12 tasks, tasks in group A being coloured green, and those in group B being coloured red.

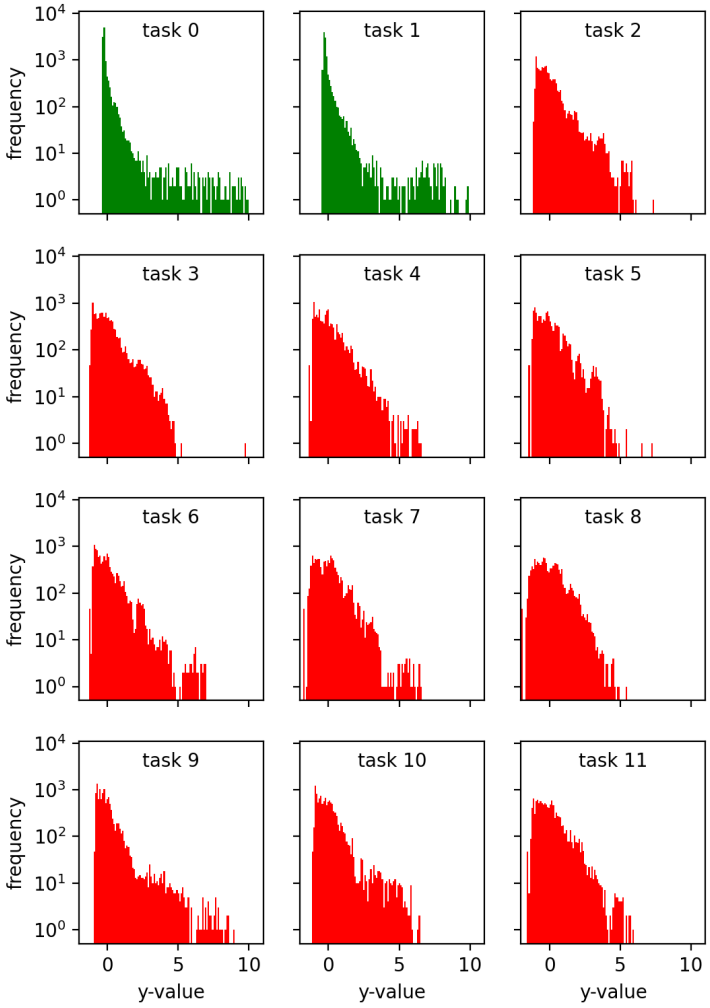
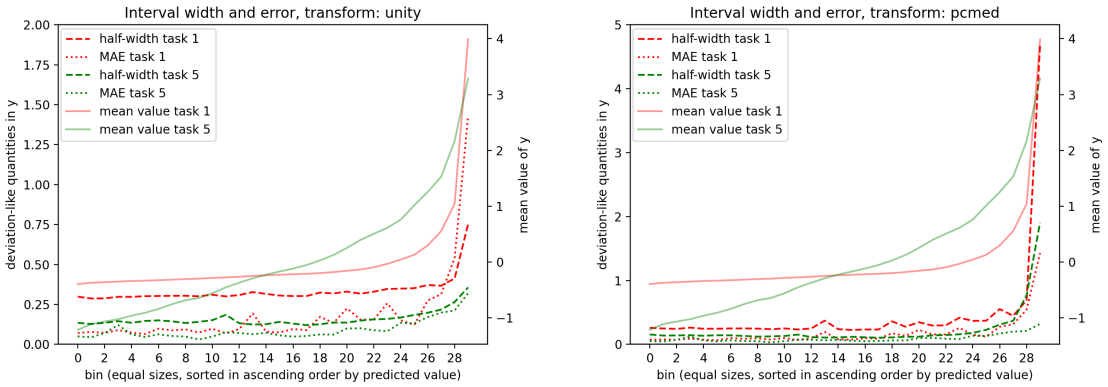


Figure 4: Histogram of logarithmic frequency of all 12 tasks. Tasks in group A (B), are coloured red (green). Width of the bin is 0.1.

Appendix E. Supporting visualisations

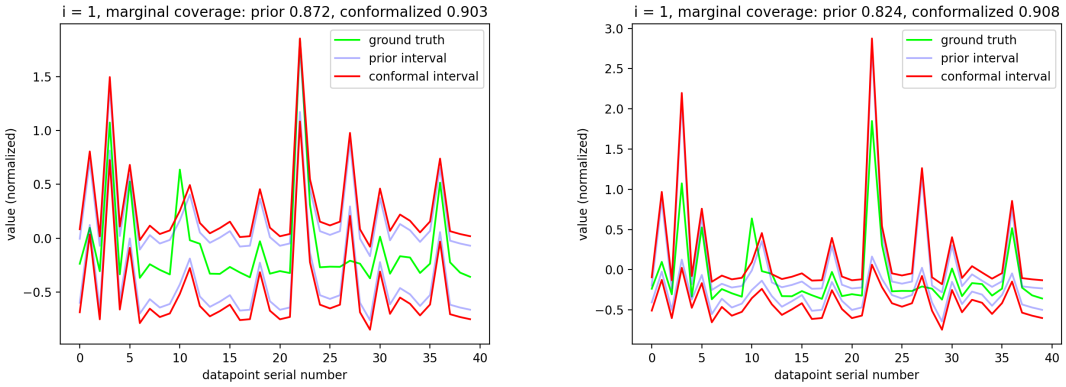
There is an alternative, less direct and precise, but potentially more intuitive way of visualizing our results. This relies on plotting the width of prediction interval as a function of y_* or its percentile (expressed with the bin value 0-30), instead of relying on the more abstract RMSCM. When supplemented with the parallel information of how big the surrogate errors are for a given y_* bin, this gives a qualitative indication of how well covered data tends to be in a given bin. Intuitively, when mean absolute error is smaller than the mean interval width, the coverage is large and vice-versa. Figure 5 shows this for a non-transformed and a transformed BNN. It is seen that a non-transformed BNN, the magnitude of the error increases faster than the interval half-width, reaching the same magnitude for the case of task 5, and significantly exceeding it for task 1. The transformation (pcmed) ensures that the interval half-width curve remains above the error curve, which is consistent with the result that coverage does not decay in the last few bins in Figure 3.



(a) (b)

Figure 5: Comparative plots of the mean prediction interval half-width (dashed) with the mean absolute error (dotted), per bin in y_*. The supplementary axis on the right of the figures is used for indicating the mean value of y in the given bin (full). Task 1 (red) is representative of group A, task 2 (green) is representative of group B. The left figure (a) shows the little adaptivity achieved by an un-transformed BNN, the right figure (b) shows the more substantial adaptivity achieved by a transformation (pcmed), in this case on the same BNN.

A further look at figure Figure 5 can be used to motivate the functional shape of $y_*(y)$ in Equation 6. This equation takes the median value y_{med} as reference. Median is chosen as opposed to mean, since for datasets with a small number of very large peaks that only extend in one direction, the mean is not representative of the usual behaviour - this is the case of tasks in group A. In theory, the y-space around $\langle y \rangle$ could even be empty. The reason why a departure from median (mean, or any other statistical element) is even considered in the numerator of Equation 6 is the assumption that the target model (GRASSMIND) is at its most precise near the median, and its surroundings are in some sense its standard regime of operation. This is a big assumption, whose adequacy can be quickly checked after



(a) (b)

Figure 6: Plot of the true values y (green) with the prediction interval (blue: before CP, red: after CP) for 40 data points for task 1, based on an MLP surrogate model. Above: no transformation, below: pcmed. Data point labels on x-axis have no further meaning, their sequence is random and points are connected only for better visualization, this is not a time-series data.

having fitted the surrogate, examining the dependence of its error on y_*, and potentially trying various ways of assembling y_* from y. Any noise in the target model translates to an irreducible error in the surrogate, and this can be easily quantified as a function of y_*.

Figure 6 shows a graphic example of what the interval prediction (built on top of an MLP with a transform) looks like directly in y-space. For the upper figure, there is no transformation applied, only a scalar correction is applied by the CP to ascertain marginal coverage near the desired value $1-\alpha = 0.9$. No information about the difficulty of individual data points is included. For the lower figure, the initial guessed scalar interval (same as in upper figure, not shown) undergoes a transformation, in this case pcmed, interval after transformation is shown in blue. It is then also fed to the CP algorithm to ensure marginal coverage, interval after CP shown in red. The result is an expansion of error bars for data points with larger y_*. Through the significant correlation of y_* and the absolute surrogate error $|y - M(X)|$, this means that more difficult data points see their error bars on average extended, significantly improving the adaptivity.

Appendix F. Implementation Details

Three types of ML algorithms were used in this study: multi-layer perceptrons (MLPs), Bayesian neural networks (BNNs), and gradient-boosted trees (XGBs). This appendix provides additional details on their implementation.

MLPs and BNNs were built using the PyTorch package (Paszke et al., 2019). BNN was built on top of a "skeleton" MLP architecture using the package bayesian-torch (Krishnan et al., 2022). Following training parameters were used in all networks:

parameter	value
activation	Leaky ReLU
optimiser	Adam
batch size	128
learning rate	10^{-3}
loss function	mean squared error
weight decay	0
torch seed	42

Table 5: Parameters common across all neural networks (MLPs, BNNs).

Optimal parameters were found manually, by a coarse trial-and-error sampling of the parameter space, which describes the architecture (activations and the number and sizes of hidden layers) and the learning process (learning rate, epochs, batch size, optimiser, loss function, weight decay) and in case of BNNs the probabilistic parameters (number of Monte Carlo samples of the output distribution, and the parameters of the prior distribution). We could afford this ad-hoc approach to tuning mainly because (i) the optima in parameter space were broad and largely insensitive to perturbations and (ii) since this is a proof-of-concept study, chasing sub-percentage improvements in key metrics is not needed.

Particular models for which results were shown are in Table 6, which includes their architecture and training lengths.

In case of BNNs, there are further parameters, the ones used are listed in Table 7.

Boosted trees (XGBs) were implemented using the package XGBoost (Chen and Guestrin, 2016). Parameters of the model used in this study are given in table 8.

model	architecture	epochs
MLP-s	64-32	1000
MLP-m	64-128-64	1000
MLP-l	128-256-512-256-128	1000
BNN-s	64-32	500
BNN-m	64-128-64	500
BNN-l	128-256-512-256-128	400

Table 6: Architectures and training epochs for the specific model configurations quoted in this study. Architecture follows the notation $N_{(0)} - N_{(1)} - ... - N_{(L-1)}$ where the N values represent the number of neurons in the L hidden layers. Smaller number of epochs for BNNs is due to more significant overfitting at 1000.

parameter	**value**
variational inference	reparametrization
μ_{prior}	0
σ_{prior}	1
$\mu_{\text{posterior, init}}$	0
$\rho_{\text{posterior, init}}$	-3
N_{MC}	1000

Table 7: Parameters specific for the BNNs.

parameter	**value**	**notes**
n_estimators	100	number of boosting rounds or decision trees to be built
max_depth	10	overfitting prevention
eta	0.3	learning rate in XGBoost nomenclature
subsample	1	fraction of the training data to be randomly sampled for each boosting round
colsample _bytree	0.5	same as above, but for features
random seed	1234	

Table 8: Parameters specific for the XGB.

The cluster's specifications used for experimentation can be found below:

- CPU: Dual AMD EPYC 7543, 32-core
- RAM: 1TB DDR4 3200MHz rECC
- GPU: 4x NVidia 3090 with 24 GB VRAM