UCL

UCL CDT DIS

UCL CDT DIS Note

1st May 2020

UK Atomic Energy Authority

Surrogate Modelling of the Tritium Breeding Ratio

Graham Van Goffrier[a] and Petr Mánek[a]

[a]University College London

The tritium breeding ratio (TBR) is an essential quantity for the design of modern and next-generation Tokamak nuclear fusion reactors. Representing the ratio between tritium fuel generated in breeding blankets and fuel consumed during reactor runtime, the TBR depends on reactor geometry and material properties in a complex manner. In this work, we explored the training of surrogate models to produce a cheap but high-quality approximation for a Monte Carlo TBR model in use at the UK Atomic Energy Authority. We investigated possibilities for dimensional reduction of its feature space, reviewed 9 families of surrogate models for potential applicability, and performed hyperparameter optimisation. Here we present the performance and scaling properties of these models, the fastest of which, an artificial neural network, demonstrated $R^2 = 0.985$ and a mean prediction time of 0.898 µs, representing a relative speedup of $8 \cdot 10^6$ with respect to the expensive MC model. We further present a novel adaptive sampling algorithm, Quality-Adaptive Surrogate Sampling, capable of interfacing with any of the individually studied surrogates. Our preliminary testing on a toy TBR theory has demonstrated the efficacy of this algorithm for accelerating the surrogate modelling process.

Contents

1 Introduction **3**
1.1 Problem Description 3

2 Data Exploration **5**
2.1 Expensive Model Description 5
2.2 Dataset Generation 6
2.3 Dimensionality Reduction 7
2.3.1 Principal Component Analysis 7
2.3.2 Variogram Computations 8
2.3.3 Autoencoders 8

3 Methodology **9**
3.1 Metrics 10
3.2 Decoupled Sampling 11
3.2.1 Experiments 12
3.3 Adaptive Sampling 13

4 Results **14**
4.1 Results of Decoupled Sampling 14
4.1.1 Hyperparameter Tuning 14
4.1.2 Scaling Benchmark 15
4.1.3 Model Comparison 16
4.2 Results of Adaptive Sampling 17

5 Conclusion **19**

Appendices **21**

A Online Materials Overview **21**

B Detailed Results **21**

1 Introduction

The analysis of massive datasets has become a necessary component of virtually all technical fields, as well as the social and humanistic sciences, in recent years. Given that rapid improvements in sensing and processing hardware have gone hand in hand with the data explosion, it is unsurprising that software for the generation and interpretation of this data has also attained a new frontier in complexity. In particular, simulation procedures such as Monte Carlo (MC) event generation can perform physics predictions even for theoretical regimes which are not analytically tractable. The bottleneck for such procedures, as is often the case, lies in the computational time and power which they necessitate.

Surrogate models, or metamodels, can resolve this limitation by replacing a resource-expensive procedure with a much cheaper approximation [1]. They are especially useful in applications where numerous evaluations of an expensive procedure are required over the same or similar domains, e.g. in the parameter optimisation of a theoretical model. The term "metamodel" proves especially meaningful in this case, when the surrogate model approximates a computational process which is itself a model for a (perhaps unknown) physical process [2]. There exists a spectrum between "physical" surrogates which are constructed with some contextual knowledge in hand, and "empirical" surrogates which are derived purely from samples of the underlying expensive model.

In this project, in coordination with the UK Atomic Energy Authority (UKAEA), we sought to develop a surrogate model for the tritium breeding ratio (TBR) in a Tokamak nuclear fusion reactor. Our expensive model was an MC-based neutronics simulation, *Paramak*[1], which returns a prediction of the TBR for a given configuration of a spherical Tokamak. We took an empirical approach to the construction of this surrogate, and no results described here are explicitly dependent on prior physics knowledge.

For the remainder of Section 1, we will define the TBR and set the context of this work within the goals of the UKAEA. In Section 2 we will describe our datasets generated from the expensive model for training and validation purposes, and the dimensionality reduction methods employed to develop our understanding of the parameter domain. In Section 3 we will present our methodologies for the comparison testing of a wide variety of surrogate modelling techniques, as well as a novel adaptive sampling procedure suited to this application. After delivering the results of these approaches in Section 4, we will give our final conclusions and recommendations for further work.

1.1 Problem Description

Nuclear fusion technology relies on the production and containment of an extremely hot and dense plasma. In this environment, by design similar to that of a star, hydrogen atoms attain energies sufficient to overcome their usual electrostatic repulsion and fuse to form helium [3]. Early prototype reactors made use of the deuterium (^{2}H, or D) isotope of hydrogen in order to achieve fusion under more accessible conditions, but lead to limited success. The current frontier generation of fusion reactors, such as the Joint European Torus (JET) and the under-construction International Thermonuclear Experimental Reactor (ITER), make use of tritium

[1] Provided by collaborator Jonathan Shimwell, at UKAEA.

(^{3}H, or T) fuel for further efficiency gain. Experimentation at JET dating back to 1997 [4] has made significant headway in validating deuterium-tritium (D-T) operations and constraining the technology which will be employed in ITER in a scaled-up form.

However, tritium is much less readily available as a fuel source than deuterium. While at least one deuterium atom occurs for every 5000 molecules of naturally-sourced water, and may be easily distilled, tritium is extremely rare in nature. It may be produced indirectly through irradiation of heavy water (D_2O) during nuclear fission, but only at very low rates which could never sustain industrial-scale fusion power.

Instead, modern D-T reactors rely on tritium breeding blankets, specialised layers of material which partially line the reactor and produce tritium upon neutron bombardment, e.g. by

$$^1_0\mathrm{n} + {}^6_3\mathrm{Li} \longrightarrow {}^3_1\mathrm{T} + {}^4_2\mathrm{He} \tag{1}$$

$$^1_0\mathrm{n} + {}^7_3\mathrm{Li} \longrightarrow {}^3_1\mathrm{T} + {}^4_2\mathrm{He} + {}^1_0\mathrm{n} \tag{2}$$

where T represents tritium and ^{7}Li, ^{6}Li are the more and less frequently occurring isotopes of lithium, respectively. ^{6}Li has the greatest tritium breeding cross-section of all tested isotopes [3], but due to magnetohydrodynamic instability of liquid lithium in the reactor environment, a variety of solid lithium compounds are preferred.

The TBR is defined as the ratio between tritium generation in the breeding blanket per unit time and tritium fuel consumption in the reactor. The MC neutronics simulations previously mentioned therefore must account for both the internal plasma dynamics of the fusion reactor and the resultant interactions of neutrons with breeding blanket materials. Neutron paths are traced through a CAD model (e.g. Figure 1) of a reactor with modifiable geometry.

The input parameters of the computationally-expensive TBR model therefore fall into two classes. Continuous parameters, including material thicknesses and packing ratios, describe the geometry of a given reactor configuration. Discrete categorical parameters further specify all relevant material sections, including coolants, armours, and neutron multipliers. One notable exception is the enrichment ratio, a continuous parameter denoting the presence of ^{6}Li. Our challenge, put simply, was to produce a fast TBR function which takes these same input parameters and approximates the MC TBR model with the greatest achievable regression performance.

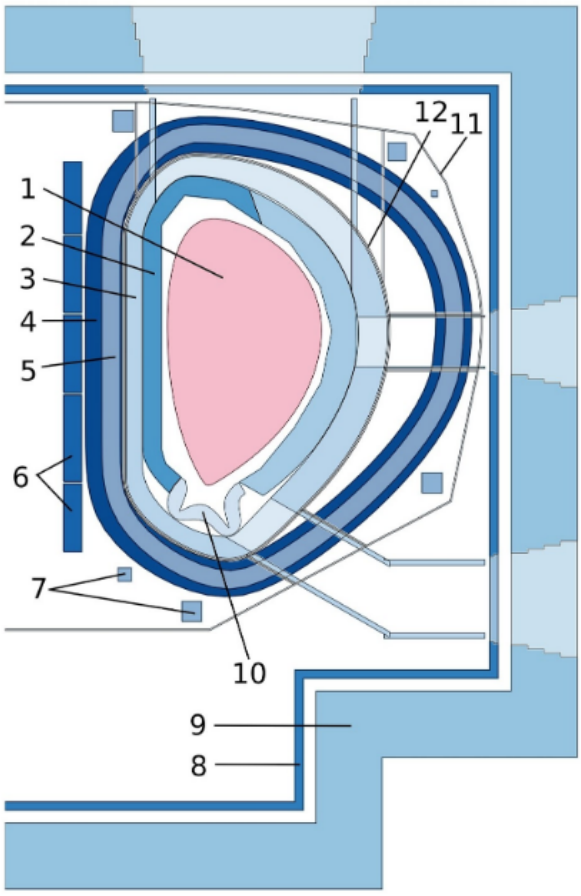


Figure 1: Typical single-null reactor configuration as specified by BLUEPRINT [5]: 1 — plasma, 2 — breeding blankets

2 Data Exploration

The initial step of our work is the study of the existing MC TBR model and its behaviour. Following the examination of its features (simulation parameters), we present efficient means of evaluating this model on large sets of points in high-performance computing (HPC) environment, preprocessing techniques designed to adapt collected datasets for surrogate modelling, and our attempts at feature space reduction to achieve the lowest possible number of dimensions.

2.1 Expensive Model Description

The expensive MC TBR model is fundamentally a Monte Carlo simulation based on the OpenMC framework [6]. As input the software expects 18 parameters, discrete and continuous, that are fully listed in Table 1. During evaluation, which usually takes units of seconds, a fixed number of neutron events is generated, and the results are given in terms of the mean and the standard deviation of the TBR aggregated over the simulated run. The former of these two we accept to be the output TBR value that is subject to approximation.

	Parameter Name	Acronym	Type	Domain
Blanket	Breeder fraction†	BBF	Continuous	$[0, 1]$
	Breeder ^{6}Li enrichment fraction	BBLEF	Continuous	$[0, 1]$
	Breeder material	BBM	Discrete	$\{Li_2TiO_3, Li_4SiO_4\}$
	Breeder packing fraction	BBPF	Continuous	$[0, 1]$
	Coolant fraction†	BCF	Continuous	$[0, 1]$
	Coolant material	BCM	Discrete	$\{D_2O, H_2O, He\}$
	Multiplier fraction†	BMF	Continuous	$[0, 1]$
	Multiplier material	BMM	Discrete	$\{Be, Be_{12}Ti\}$
	Multiplier packing fraction	BMPF	Continuous	$[0, 1]$
	Structural fraction†	BSF	Continuous	$[0, 1]$
	Structural material	BSM	Discrete	$\{SiC, eurofer\}$
	Thickness	BT	Continuous	$[0, 500]$
First wall	Armour fraction‡	FAF	Continuous	$[0, 1]$
	Coolant fraction‡	FCF	Continuous	$[0, 1]$
	Coolant material	FCM	Discrete	$\{D_2O, H_2O, He\}$
	Structural fraction‡	FSF	Continuous	$[0, 1]$
	Structural material	FSM	Discrete	$\{SiC, eurofer\}$
	Thickness	FT	Continuous	$[0, 20]$

Table 1: Input parameters supplied to the MC TBR simulation in alphabetical order. Groups of fractions marked†‡ are independently required to sum to one.

In the following sections, we often reference TBR points or samples. These are simply vectors in the feature space generated by Cartesian product of domains of all features—parameters from Table 1.

Since most surrogate models that we employ assume overall continuous numerical inputs, we take steps to unify our feature interface in order to attain this property. In particular, we transform discrete features by embedding each such feature using standard one-hot encoding. This option is available to us since discrete domains that generate our feature space are finite in cardinality and

relatively small in size. And while it helps us towards unification, this step comes at the expense of increasing the dimensionality of the feature space. This is further discussed in Section 2.3.

2.2 Dataset Generation

In our work, we deliberately make no assumptions about the internal properties of the MC TBR simulation, effectively treating it as a black box model. This limits our means of studying its behaviour to inspection of its outputs at various points in the feature space. We therefore require sufficiently large and representative quantities of samples to ensure that surrogates can be trained to approximate the MC TBR model accurately.

With a grid search in such a high-dimensional domain clearly intractable, we selected uniform pseudo-random sampling[2] to generate large amounts of feature configurations that we consider to be independent and unbiased. For evaluation of the expensive MC TBR model, we utilise parallelisation offered by the HPC infrastructure available at UCL computing facilities. To this end, we designed and implemented the Approximate TBR Evaluator—a Python software package capable of sequential evaluation of the multi-threaded OpenMC simulation on batches of previously generated points in the feature space. Having deployed ATE at the UCL Hypatia cluster[3], we completed three data generation runs that are summarised in Table 2.

#	Samples	Batch division	t_{run}	$\bar{t}_{\text{eval.}}$ [s]	Description
0	100 000	100×1000	2 days, 23 h	7.88 ± 2.75	Testing run using old MC TBR version.
1	500 000	500×1000	13 days, 20 h	7.78 ± 2.81	Fully uniform sampling in the entire domain.
2	400 000	400×1000	10 days	7.94 ± 2.60	Mixed sampling, discrete features fixed.

Table 2: Parameters of sampling runs. Here, t_{run} denotes the total run time (including waiting in the processing queue), and $\bar{t}_{\text{eval.}}$ is the mean evaluation time of the MC TBR model (per single sampled point).

Skipping run zero, which was performed using an older, fundamentally different version of the MC TBR software, and was thus treated as a technical proof-of-concept, we generated a total of 900 000 samples in two runs. While the first run featured fully uniform sampling of the unrestricted feature space, the second run used a more elaborate strategy. Interested in further study of relationships between discrete and continuous features, we selected four assignments of discrete features (listed in Table 3) and fixed them for all points, effectively slicing the feature space into four corresponding subspaces. In order to achieve comparability, all such *slices* use the same samples for the values of continuous features.

Since some surrogate modelling methods applied in this work are not scale-invariant or perform suboptimally with arbitrarily scaled inputs, all obtained TBR samples (features and TBR values) were standardised prior to further use. In this commonly used statistical procedure, features and regression outputs are independently scaled and offset to attain zero mean and unit variance.

[2] Continuous and discrete parameters are drawn from a corresponding uniform distribution over their domain, as defined in Table 1. For repeatability, each run uses a seed equal to its number.

[3] The Hypatia RCIF partition is comprised of 4 homogeneous nodes. Each node is installed with 376 GB RAM and 40 Intel® Xeon® Gold 6148 CPUs of clock frequency 2.40 GHz.

	Discrete feature assignment					
Batches	BBM	BCM	BMM	BSM	FCM	FSM
0-99	Li_4SiO_4	H_2O	$Be_{12}Ti$	eurofer	H_2O	eurofer
100-199	Li_4SiO_4	He	$Be_{12}Ti$	eurofer	H_2O	eurofer
200-299	Li_4SiO_4	H_2O	$Be_{12}Ti$	eurofer	He	eurofer
300-399	Li_4SiO_4	He	$Be_{12}Ti$	eurofer	He	eurofer

Table 3: Selected discrete feature assignments corresponding to slices in run 2.

2.3 Dimensionality Reduction

Model training over high-dimensional parameter spaces (illustrated in Figure 2) may be improved in many aspects by carefully reducing the number of variables used to describe the space. For many applications, feature selection strategies succeed in identifying a sufficiently representative subset of the original input variables. However, all given variables were assumed to be physically relevant to the MC TBR model. On the other hand, feature extraction methods aim to identify a transformation of the parameter space which decreases dimensionality; even if no individual parameter is separable from the space, some linear combinations of parameters or nonlinear functions of parameters may be.

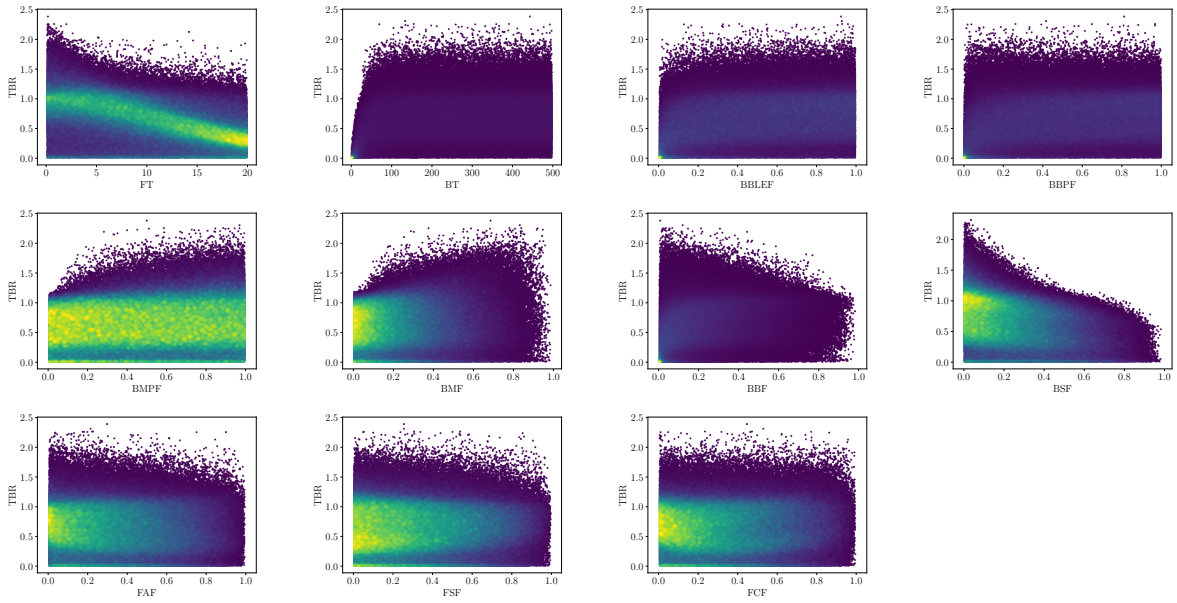


Figure 2: Marginalised dependence of TBR on the choice of continuous features in 500 000 points generated in run 1. Points are coloured by density.

2.3.1 Principal Component Analysis

To pursue linear feature extraction, principal component analysis (PCA) [7] was performed via SciKit Learn [8] on a set of 300 000 uniform samples of the MC TBR model.

Figure 3 shows the resultant cumulative variance of the 11 principal components. The similar share of variance among all features reveals irreducibility of the TBR model by linear methods.

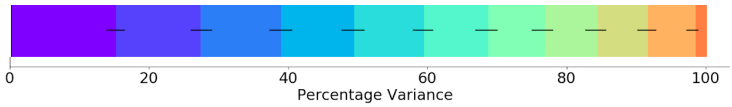


Figure 3: Cumulative variance for optimal features identified by PCA

2.3.2 Variogram Computations

Kriging is a geostatistical surrogate modelling technique which relies on correlation functions over distance (lag) in the feature space [9]. Although kriging performed poorly for our use case due to high dimensionality, these correlation measures gave insight into similarities between discrete-parameter slices of the data.

Figure 4 shows the Matheron semivariance [10] for three discrete slices with coolant material varied, but all other discrete parameters fixed. Fits [11] to the Matérn covariance model confirmed numerically that the coolant material is the discrete parameter with the greatest distinguishability in the MC TBR model.

2.3.3 Autoencoders

Autoencoders [12] are a family of approaches to dimensionality reduction driven by artificial neural networks (ANNs). Faced with a broad selection of alternatives, we opted for a conventional autoencoder architecture with a single hidden layer. While it follows that the input and output layers of such network are sized to accommodate the analysed dataset, the hidden layer, also called *the bottleneck*, allows for variable number of neurons that represent a smaller subspace. By scanning over a range of bottleneck widths and investigating relative changes in the validation loss, we assess the potential for dimensional reduction.

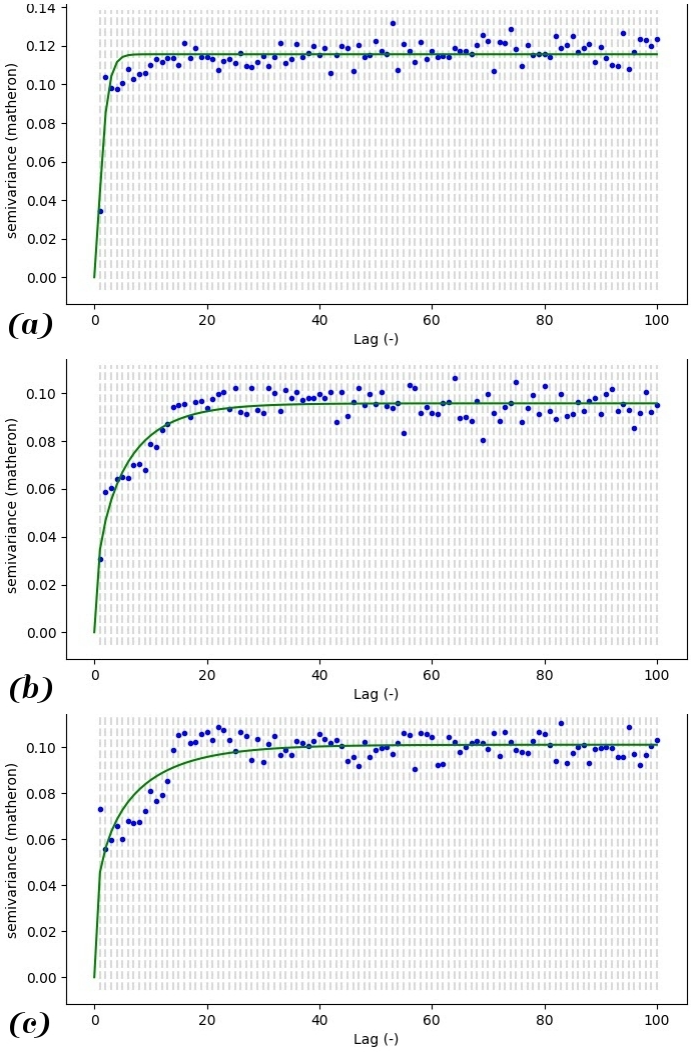


Figure 4: Semivariograms for MC TBR data with coolant materials: (a) He, (b) H_2O, (c) D_2O

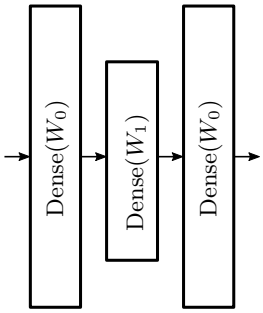


Figure 5: Autoencoder with input width W_0 and bottleneck width W_1. Here, Dense(N) denotes a fully-connected layer of N neurons.

In particular, we consider two equally-sized sets[4] of samples: (a) a subset of data obtained from run 1 and (b) a subset of a single slice obtained from run 2. Our expectation was that while the former case would provide meaningful insights into correlations within the feature space, the latter would validate our autoencoder implementation by analysing a set of points that are trivially reducible in dimensionality due to a number of fixed discrete features.

The results of both experiments are shown in Figure 6. Consistent with our motivation, in each plot we can clearly identify a constant plateau of low error in the region of large dimensionality followed by a point, from which a steep increase is observed. We consider this *critical point* to mark the largest viable dimensional reduction without significant information loss. With this approach we find that the autoencoder was able to reduce the datasets into a subspace of 18 dimensions in the first case and 10 dimensions in the second case.

Confirming our expectation that in the latter, trivial case the autoencoder should achieve greater dimensional reduction, we are inclined to believe that our implementation is indeed operating as intended. However, we must also conclude that in both examined cases this method failed to produce a reduction that would prove superior to a naïve approach.[5] This is consistent with the previous results obtained by PCA and kriging.

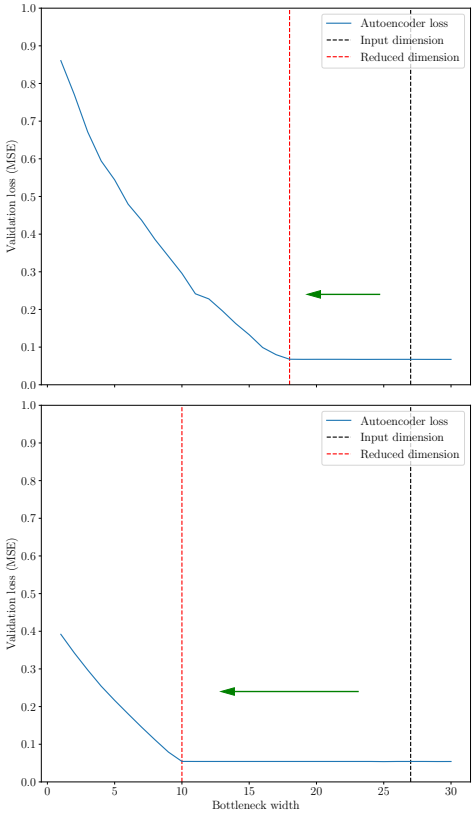


Figure 6: Autoencoder loss scan on the full feature space (top) and a single slice (bottom). Dimensional reduction is indicated by a green arrow.

3 Methodology

Assuming that data is appropriately treated to eliminate redundant features, we proceed to propose surrogate models and criteria used for their evaluation. The task all presented surrogates strive to solve can be formulated using the language of conventional regression problems. In this section, we focus on interpretation in the scheme of supervised learning with decoupled and adaptive sampling.

Labeling the expensive MC TBR model $f(x)$, a surrogate is a mapping $\hat{f}(x)$ that yields similar images as $f(x)$. In other words, $f(x)$ and $\hat{f}(x)$ minimise a selected dissimilarity metric. Furthermore, in order to be considered *viable*, surrogates are required to achieve expected evaluation time lower than that of $f(x)$.

[4] Each set contained 100 000 samples from batches 0-99 of the corresponding runs.

[5] In both tested cases we can trivially eliminate 6 dimensions due to overdetermination of one-hot-encoded categorical features, and 2 dimensions corresponding to sum-to-one constraints. Furthermore, in the single slice case we may omit 6 additional dimensions due to fixed feature assignment.

In the decoupled sampling approach that is further described in Section 3.2, we first gather a sufficiently large training set of samples $\mathcal{T} = \left\{\left(x^{(i)}, f\left(x^{(i)}\right)\right)\right\}_{i=1}^{N}$ to describe the behaviour of $f(x)$ across its domain. Depending on a specific model family and appropriate choice of its hyperparameters, surrogate models $\hat{f}(x)$ are trained to minimise the empirical risk $R_{\text{emp.}}$ with respect to $\mathcal{T}$ and a model-specific loss function $\mathcal{L}$, where the empirical risk is defined as $R_{\text{emp.}}(\hat{f} \mid \mathcal{T}, \mathcal{L}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\left(\hat{f}(x^{(i)}), f(x^{(i)})\right)$.

The adaptive sampling approach that we characterise in Section 3.3 can be viewed as a more general problem. Rather than fixing the training set $\mathcal{T}$ for the entire duration of training, multiple sets $\{\mathcal{T}_k\}_{k=0}^{K}$ are used, such that $\mathcal{T}_{k-1} \subset \mathcal{T}_k$ for all $k > 1$. The first set $\mathcal{T}_0$ is initialised randomly to provide a *burn-in*, and is repeatedly extended in epochs, whereby each epoch trains a new surrogate $\hat{f}_k(x)$ on $\mathcal{T}_k$ using the supervised learning procedure, evaluates its performance, and forms a new set $\mathcal{T}_{k+1}$ by adding more samples to $\mathcal{T}_k$. This permits the learning algorithm to condition the selection of new samples in $\mathcal{T}_{k+1}$ on the evaluation results of $\hat{f}_k(x)$ in order to maximise improvement of surrogate performance over complex regions within the feature space.

3.1 Metrics

Aiming to provide objective comparison of a diverse set of surrogate model families, we define a multitude of metrics to be tracked during experiments. Following the motivation of this work, two desirable properties of surrogates arise: (a) their capability to approximate the TBR MC model well and (b) their prediction time. An ideal surrogate would maximise the former while minimising the latter.

Table 4 provides an exhaustive listing and description of metrics recorded in our experiments. For regression performance analysis, we include a selection of absolute metrics to assess the approximation capability of surrogates, and set practical bounds on the expected uncertainty of their predictions. In addition, we also track relative measures that are better-suited for comparison between this work and others, as they maintain invariance with respect to the selected domain and image space. For complexity analysis, surrogates are assessed in terms of wall time[6] elapsed during training and prediction. This is motivated by common practical use cases of our work, where models are trained and used as drop-in replacements for the expensive MC TBR model. Since training set sizes remain to be determined, all times are reported per a single datapoint. Even though some surrogates support acceleration by means of parallelisation, sequential processing of samples was ensured to achieve comparability between considered models.[7]

To prevent undesirable bias in results due to training set selection, all metrics are collected in the scheme of k-fold cross-validation with a conventional choice of $k = 5$. In this setting, a sample set is uniformly divided into 5 disjoint folds, each of which is used as a test set for models trained on the remaining 4.[8] Having repeated the same experiment for each such run, the overall value of individual metrics is reported in terms of their mean and standard deviation over all folds.

[6] Real time elapsed during computation measured by a chronometer, here by means of the Python `time` package.

[7] The only exception to this are artificial neural networks, which require considerable amount of processing power for training on conventional CPU architectures.

[8] Unless explicitly stated otherwise, we use 1 fold for testing and the 4 remaining folds for training. This gives 80% to 20% train-test ratio.

Regression performance metrics	Mathematical formulation / description	Ideal value [units]
Mean absolute error (MAE)	$\sum_{i=1}^{N} \lvert y^{(i)} - \hat{y}^{(i)} \rvert / N$	0 [TBR]
Standard error of regression S	$\mathrm{StdDev}_{i=1}^{N} \left\{ \lvert y^{(i)} - \hat{y}^{(i)} \rvert \right\}$	0 [TBR]
Coefficient of determination R^2	$1 - \sum_{i=1}^{N} \left(y^{(i)} - \hat{y}^{(i)} \right)^2 / \sum_{i=1}^{N} \left(y^{(i)} - \overline{y} \right)^2$	1 [rel.]
Adjusted R^2	$1 - (1 - R^2)(N-1)/(N-P-1)$	1 [rel.]
Complexity metrics		
Mean training time $\overline{t}_{\mathrm{trn.}}$	(wall training time of $\hat{f}(x)$)$/N_0$	0 [ms]
Mean prediction time $\overline{t}_{\mathrm{pred.}}$	(wall prediction time of $\hat{f}(x)$)$/N$	0 [ms]
Relative speedup ω	(wall evaluation time of $f(x)$)$/(N\overline{t}_{\mathrm{pred.}})$	$\to \infty$ [rel.]

Table 4: Metrics recorded in supervised learning experiments. In formulations, we work with a training set of size N_0 and a test set of size N, TBR values $y^{(i)} = f(x^{(i)})$ and $\hat{y}^{(i)} = \hat{f}(x^{(i)})$ denote images of the ith testing sample in the expensive model and the surrogate respectively. Furthermore, the mean $\overline{y} = \sum_{i=1}^{N} y^{(i)}/N$ and P is the number of input features.

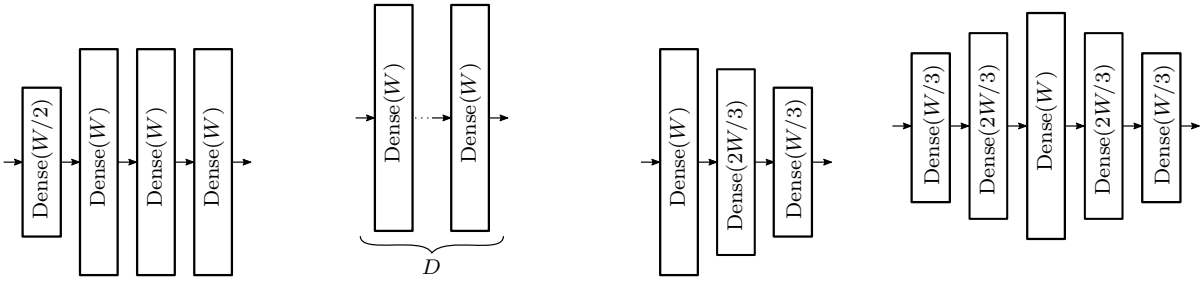
3.2 Decoupled Sampling

In our experiments, we evaluate and compare surrogates in effort to optimise against metrics described in Section 3.1. To attain meaningful and practically usable results, we require a sufficiently large and diverse pool of surrogate families to review. This is described by the listing in Table 5. The presented selection of models includes basic techniques suitable for linear regression enhanced by the kernel trick or dimension lifting, methods driven by decision trees, instance-based learning models, ensemble regressors, randomised algorithms, artificial neural networks and mathematical approaches developed specifically for the purposes of surrogate modelling. For each of these families, a state-of-the-art implementation was selected and adapted to operate with TBR samples.

Surrogate	Acronym	Implementation	Hyperparameters
Support vector machines [13]	SVM	SciKit Learn [8]	3
Gradient boosted trees [14–16]	GBT	SciKit Learn	11
Extremely randomised trees [17]	ERT	SciKit Learn	7
AdaBoosted decision trees [18]	ABT	SciKit Learn	3
Gaussian process regression [19]	GPR	SciKit Learn	2
k nearest neighbours	KNN	SciKit Learn	3
Artificial neural networks	ANN	Keras (TensorFlow) [20]	2
Inverse distance weighing [21]	IDW	SMT [22]	1
Radial basis functions	RBF	SMT	3

Table 5: Considered surrogate model families.

While some of the presented model families are clearly determined by an explicit choice of a learning algorithm, others may vary considerably depending on hyperparameter configuration. A good example of the latter group are artificial neural networks, which in addition to conventional hyperparameters enable to control network architecture through selection from various parametric graph templates (illustrated in Figure 7). This allows us to realise a simplistic network architecture search during the hyperparameter tuning procedure.



(a) 1h3f(W) (b) df(D, W) (c) 3pyramid(W) (d) 5diamond(W)

Figure 7: Selected parametric neural network architectures. All layers except the last use ReLU activation. Prediction information flow is indicated by arrows.

3.2.1 Experiments

The presented surrogate candidates are evaluated in four experimental cases:

1. Hyperparameter tuning in a simplified domain.
2. Hyperparameter tuning in full domain.
3. Scaling benchmark.
4. Model comparison.

The aim of the initial experiments is to use a relatively small subset of collected TBR samples to determine hyperparameters of considered surrogates. Since this process requires learning the behaviour of an unknown, possibly expensive mapping – here a function that assigns cross-validated metrics to a point in the hyperparameter domain – it in many aspects mirrors the primary task of this work with the notable extension of added utility to optimise. In order to avoid undesirable exponential slowdown in exhaustive searches of a possibly high-dimensional parameter space, Bayesian optimisation [23] is employed as a standard hyperparameter tuning algorithm. We set its objective to maximise R^2 and perform 1000 iterations.[9]

In the first experiment, efforts are made to maximise the possibility of success in surrogates that are prone to suboptimal performance in discontinuous spaces. This follows the notion that, if desired, performance of such models may be replicated by training separate instances to model each continuous subregion of the domain independently. To this end, data are limited to a single slice from run 2, and discrete features are completely withheld from evaluated surrogates. This is repeated for each of the four available slices to investigate variance in behaviour under different discrete feature assignments. The second experiment conventionally measures surrogate performance on the full feature space. Here, in extension of the previous case, surrogates work with samples comprised of discrete as well as continuous features.

The objective of the last two experiments is to exploit the information gathered by hyperparameter tuning. In the third experiment, the 20 best-performing hyperparameter configurations of each family (with respect to R^2) are used to perform training on progressively larger sets to investigate their scaling properties. Following that, the fourth experiment aims to produce

[9] Hyperparameter tuning of each surrogate family was terminated after 2 days. Instances that reached this limit may be identified in Tables 7 and 8 in the Appendix.

surrogates suitable for practical use by retraining selected well-scaling instances on large training sets to satisfy the goals of this work.

3.3 Adaptive Sampling

All of the surrogate modelling techniques studied in this project face a common challenge: their accuracy is limited by the quantity of training samples which are available from the expensive MC TBR model. Adaptive sampling procedures can improve upon this limitation by taking advantage of statistical information which is accumulated during the training of any surrogate model. Rather than training the surrogate on a single sample set generated according to a fixed strategy, sample locations are chosen periodically during training so as to best suit the model under consideration.

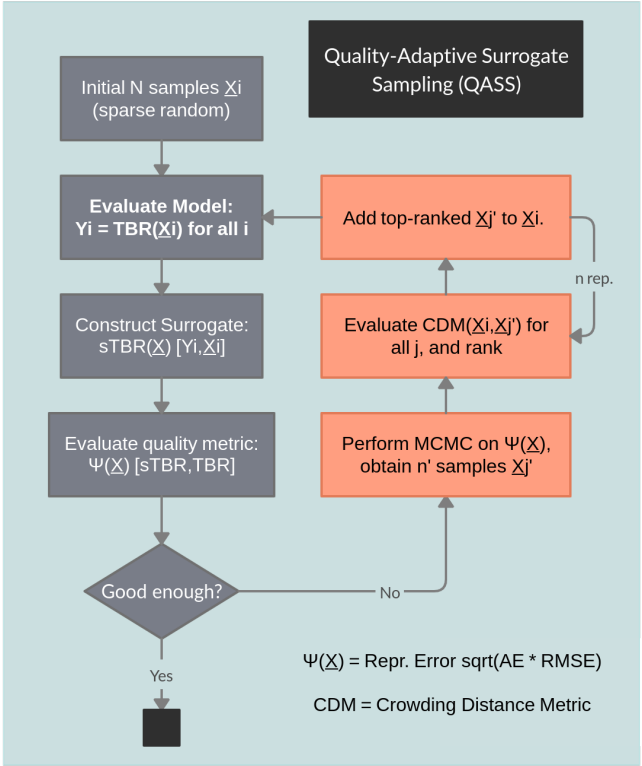


Figure 8: Schematic of QASS algorithm

Adaptive sampling techniques appear frequently in the literature and have been specialised for surrogate modelling. Garud's [24] "Smart Sampling Algorithm" achieved notable success by incorporating surrogate quality and crowding distance scoring to identify optimal new samples, but was only tested on a single-parameter domain. We theorised that a nondeterministic sample generation approach, built around Markov Chain Monte Carlo methods (MCMC), would fare better for high-dimensional models by more thoroughly exploring all local optima in the feature space. MCMC produces a progressive chain of sample points, each drawn according to the same symmetric proposal distribution[10] from the prior point. These sample points will converge to a desired posterior distribution, so long as the acceptance probability for these draws has a particular functional dependence on that posterior value (see [26] for a review).

Many researchers have embedded surrogate methods into MCMC strategies for parameter optimisation [27, 28], in particular the ASMO-PODE algorithm [29] which makes use of MCMC-based adaptive sampling to attain greater surrogate precision around prospective optima. Our novel approach draws inspiration from ASMO-PODE, but instead uses MCMC to generate samples which increase surrogate precision throughout the entire parameter space.

[10] An adaptive MCMC procedure [25], which adjusts an ellipsoidal proposal distribution to fit the posterior, was also implemented but not fully tested.

We designed the Quality-Adaptive Surrogate Sampling algorithm (QASS, Figure 8) to iteratively increment the training/test set with sample points which maximise surrogate error and minimise a crowding distance metric (CDM) [30] in feature space. On each iteration following an initial training of the surrogate on N uniformly random samples, the surrogate was trained and absolute error calculated. MCMC was then performed on the error function generated by performing nearest-neighbor interpolation on these test error points. The resultant samples were culled by 50% according to the CDM, and then the n highest-error candidates were selected for reintegration with the training/test set, beginning another training epoch. Validation was also performed during each iteration on independent, uniformly-random sample sets.

4 Results

Having outlined a variety of models and metrics tracked for the purposes of their objective comparison, we proceed to present and discuss our results in the next sections.

4.1 Results of Decoupled Sampling

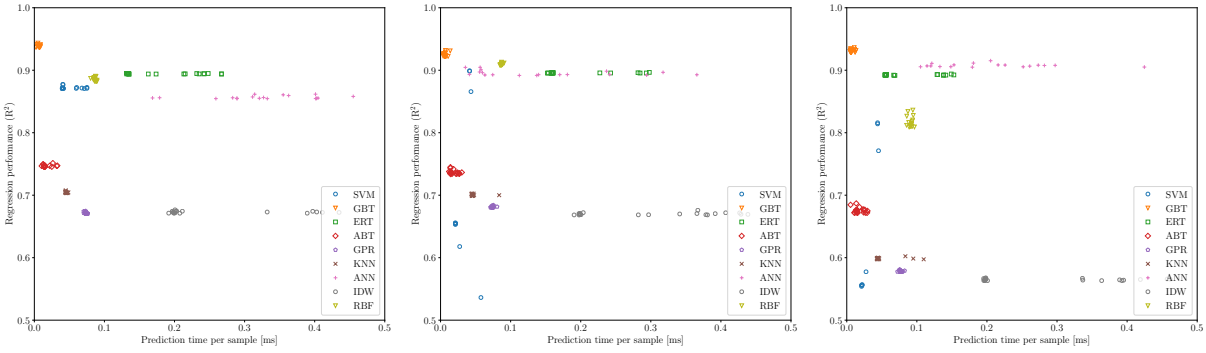
We begin by comparing the performance of a diverse set of surrogate families on previously generated samples of the expensive MC TBR model. Through the four experimental cases described in Section 3.2.1, we aim to study properties of the considered models in terms of regression performance, training and prediction time.

4.1.1 Hyperparameter Tuning

The first two experiments perform Bayesian optimisation to maximise R^2 in a cross-validation setting as a function of model hyperparameters. While in the first experiment we limit training and test sets to the scope of four selected slices of the feature space, in the second experiment we lift this restriction to examine surrogate capability to model a more complex domain.

The results displayed in Figure 9 (and listed in Table 7 in the Appendix) indicate that in the first experiment, GBTs clearly appear to be the most accurate as well as the fastest surrogate family in terms of mean prediction time. Following that, we note that ERTs, SVMs and ANNs also achieved satisfactory results with respect to both examined metrics. While the remainder of tested surrogate families does not exhibit problems in complexity, its regression performance falls below average.

The results of the second experiment, shown in Figure 10 (and listed in Table 8 in the Appendix), seem to confirm our expectations. Compared to the previous case, we observe that many surrogate families consistently achieved worse regression performance and prediction times. The least affected models appear to be GBTs, ANNs and ERTs, which are known to be capable of capturing relationships involving mixed feature types that were deliberately withheld in the first experiment. With only negligible differences, the first two of these families appear to be tied for the best performance as well as the shortest prediction time. We observe that ERTs and RBFs also demonstrated satisfactory results, relatively outperforming the remaining surrogates in terms of regression performance, and in some cases also in prediction time.



(a) Run 2, batches 0-2 (b) Run 2, batches 100-102 (c) Run 2, batches 200-202

Figure 9: 20 best-performing surrogates per each considered family, plotted in terms of complexity (as $\bar{t}_{\text{pred.}}$) and regression performance (as R^2) on selected slices of run 2, evaluated in experiment 1. Here, batches refer to subsets of training and test datasets that may be matched to slices using Table 3.

Following both hyperparameter tuning experiments, we conclude that while domain restrictions employed in the first case have proven effective in improving the regression performance of some methods, this result has fluctuated considerably depending on the selected slices. Furthermore, in all instances the best results were achieved by families of surrogates that were nearly unaffected by this modification.

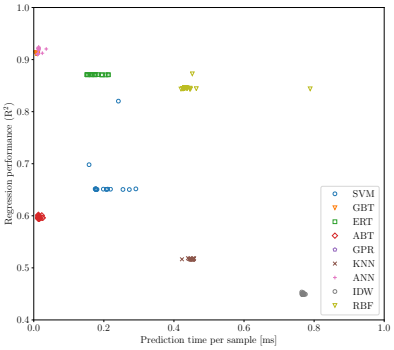


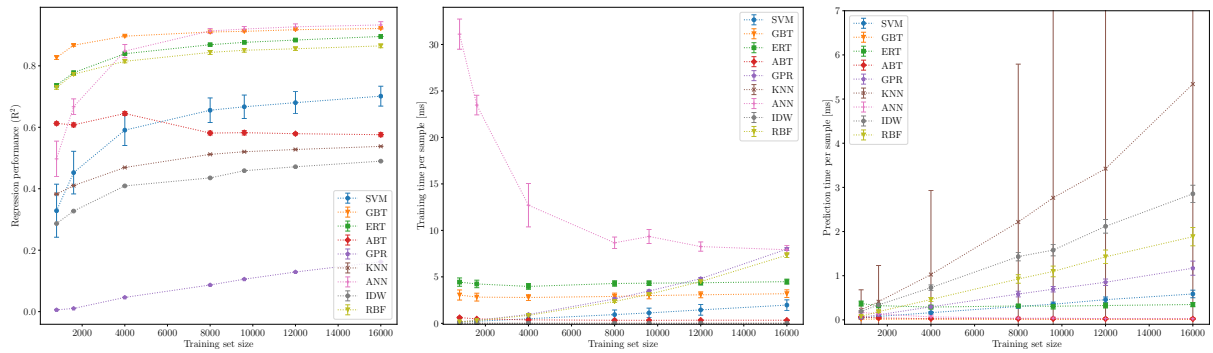
Figure 10: Results of experiment 2, plotted analogously to Figure 9.

4.1.2 Scaling Benchmark

In the third experiment we examine surrogate scaling properties by correlating metrics of interest with training set size. Firstly, the results shown in Figure 11a (and listed in Table 9 in the Appendix) suggest that the most accurate families from the previous experiments consistently maintain their relative advantage over others, even as we introduce more training points. While such families achieve nearly comparable performance on the largest dataset, in the opposite case tree-based approaches clearly outperform ANNs. This can be observed particularly on sets of sizes up to 6000.

Next, we examine scaling behaviour in terms of the mean training time (displayed in Figure 11b and listed in Table 10 in the Appendix). Consistent with our expectation, the shortest times were achieved by instance-based learning methods (e.g. KNN, IDW) that are trained trivially at the expense of increased lookup complexity later during prediction. Furthermore, we observe that the majority of tree-based algorithms also perform and scale well, unlike RBFs and GPR which appear to behave superlinearly. We note that ANNs, which are the only family to utilise parallelisation during training, show an inverse scaling characteristic. Our conjecture is that this effect may be caused by a constant multi-threading overhead that possibly dominates the training process on relatively small training sets.

Finally, we study scaling with respect to the mean prediction time (shown in Figure 11c and listed in Table 11 in the Appendix). Our initial observation is that all tested families with the exception of previously mentioned instance-based models offer desirable characteristics overall.



(a) Regression performance (as R^2) (b) Complexity (as $\overline{t}_{\text{trn.}}$) (c) Complexity (as $\overline{t}_{\text{pred.}}$)

Figure 11: Various metrics collected during experiment 3 (scaling benchmark) displayed as a function of training set size.

Analogous to previous experiments, GBTs, ABTs and ANNs appear to be tied, as they not only exhibit comparable times but also similar scaling slopes. Following that, we notice a clear hierarchy of ERTs, SVMs, GPR and RBFs, trailed by IDW and KNNs.

4.1.3 Model Comparison

In the fourth experiment proposed in Section 3.2.1, we exploit previously collected information to produce surrogates with desirable properties for practical use. We aim to create models that yield: (a) the best regression performance regardless of other features, (b) acceptable performance with the shortest mean prediction time, or (c) acceptable performance with the smallest training set. To this end, we trained 8 surrogates that are presented in Figure 12 and Table 6.

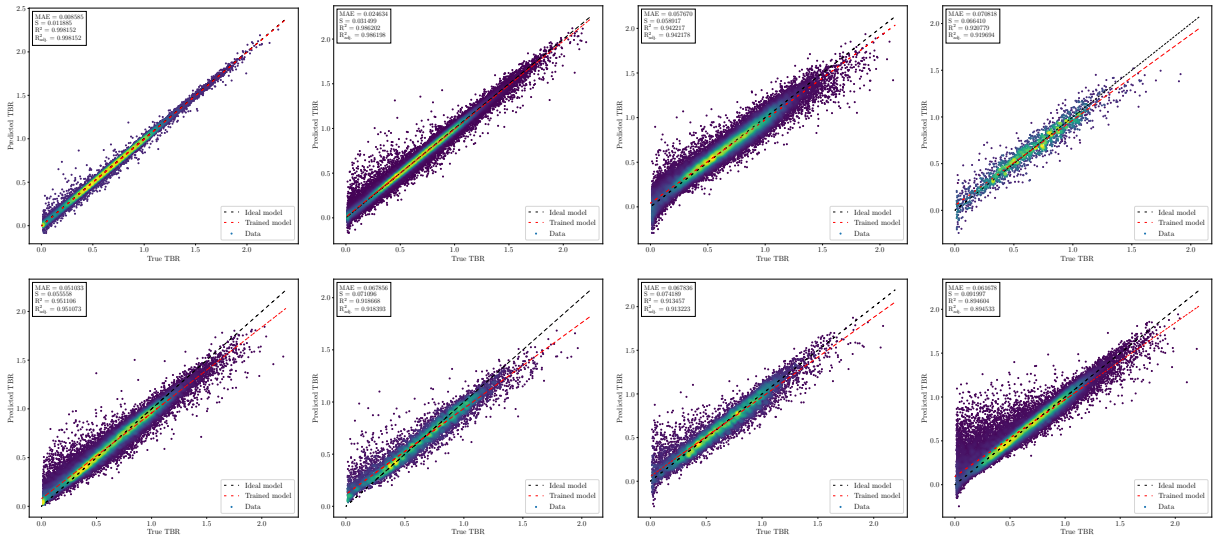


Figure 12: Regression performance of models 1-4 (row 1, from the left) and 5-8 (row 2) trained in experiment 4 (model comparison), viewed as true vs. predicted TBR on a test set of a selected cross-validation fold. Points are coloured by density.

Having selected ANNs, GBTs, ERTs, RBFs and SVMs based on the results of experiments 2-3, we utilised the best-performing hyperparameters. In pursuit of goal (a), the best approximator (no. 1, ANN) achieved $R^2 = 0.998$ and mean prediction time $\bar{t}_{\text{pred.}} = 1.124\,\mu\text{s}$. These correspond to a standard error $S = 0.013$ and a relative speedup $\omega = 6\,916\,416\times$ with respect to the MC TBR evaluation baseline measured during run 1 (see Table 2 for details). Satisfying goal (b), the fastest model (no. 2, ANN) achieved $R^2 = 0.985$, $\bar{t}_{\text{pred.}} = 0.898\,\mu\text{s}$, $S = 0.033$ and $\omega = 8\,659\,251\times$. While these surrogates were trained on the entire available set of 500 000 datapoints, to satisfy goal (c) we also trained a more simplified model (no. 4, GBT) that achieved $R^2 = 0.913$, $\bar{t}_{\text{pred.}} = 6.125\,\mu\text{s}$, $S = 0.072$ and $\omega = 1\,269\,777\times$ with a set of size only 10 000.

		Regression performance				Complexity		
Model	$\lvert\mathcal{T}\rvert$	MAE [TBR]	S [TBR]	R^2 [rel.]	$R^2_{\text{adj.}}$ [rel.]	$\bar{t}_{\text{trn.}}$ [ms]	$\bar{t}_{\text{pred.}}$ [ms]	ω [rel.]
1 (ANN)	500	$\mathbf{0.009 \pm 0.000}$	$\mathbf{0.013 \pm 0.001}$	$\mathbf{0.998 \pm 0.000}$	$\mathbf{0.998 \pm 0.000}$	3.659 ± 0.035	0.001 ± 0.000	$6\,916\,416\times$
2 (ANN)	500	0.025 ± 0.001	0.033 ± 0.001	0.985 ± 0.001	0.985 ± 0.001	2.989 ± 0.026	$\mathbf{0.001 \pm 0.000}$	$8\,659\,251\times$
3 (GBT)	200	0.058 ± 0.001	0.059 ± 0.000	0.941 ± 0.001	0.941 ± 0.001	2.221 ± 0.010	0.007 ± 0.000	$1\,169\,933\times$
4 (GBT)	10	0.071 ± 0.002	0.072 ± 0.003	0.913 ± 0.006	0.912 ± 0.006	$\mathbf{1.621 \pm 0.008}$	0.006 ± 0.000	$1\,269\,777\times$
5 (ERT)	200	0.051 ± 0.000	0.056 ± 0.000	0.950 ± 0.001	0.950 ± 0.001	2.634 ± 0.010	0.214 ± 0.004	$36\,308\times$
6 (ERT)	40	0.068 ± 0.000	0.072 ± 0.000	0.917 ± 0.001	0.917 ± 0.001	2.368 ± 0.005	0.188 ± 0.008	$41\,370\times$
7 (RBF)	50	0.068 ± 0.001	0.077 ± 0.002	0.910 ± 0.003	0.910 ± 0.003	3.453 ± 0.019	1.512 ± 0.016	$5143\times$
8 (SVM)	200	0.062 ± 0.000	0.094 ± 0.002	0.891 ± 0.003	0.891 ± 0.003	33.347 ± 0.382	2.415 ± 0.011	$3220\times$

Table 6: Results of experiment 4. Here, figures are reported over 5 cross-validation folds, $\lvert\mathcal{T}\rvert$ denotes cross-validation set size ($\times 10^3$) and ω is a relative speedup with respect to $\bar{t}_{\text{eval.}} = 7.777\,\text{s}$ measured in the MC TBR model during run 1 (see Table 2). The best-performing metrics are highlighted in bold.

Overall we found that due to their superior performance, boosted tree-based approaches seem to be advantageous for fast surrogate modelling on relatively small training sets (up to the order of 10^4). Conversely, while neural networks perform poorly in such a setting, they dominate on larger training sets (at least of the order of 10^5) both in terms of regression performance and mean prediction time.

4.2 Results of Adaptive Sampling

In order to test our QASS prototype, several functional toy theories for TBR were developed as alternatives to the expensive MC model. By far the most robust of these was the following sinusoidal theory with adjustable wavenumber parameter n:

$$\text{TBR} = \frac{1}{|C|} \sum_{i \in C} [1 + \sin(2\pi n(x_i - 1/2))] \qquad (3)$$

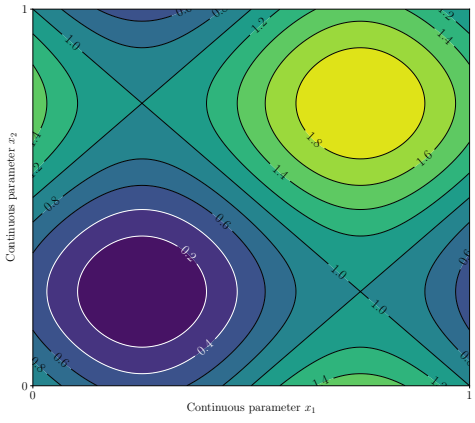


Figure 13: Sinusoidal toy TBR theory over two continuous parameters, $n = 1$.

plotted in Figure 13 for $n = 1$ and two continuous parameters C. ANNs trained on this model demonstrated similar performance to those on the expensive MC model. QASS performance was verified by training a 1h3f(256) ANN on the sinusoidal theory for varied quantities of initial, incremental, and MCMC candidate samples. Although the scope of this project did not include thorough searches of this hyperparameter domain, sufficient runs were made to identify some likely trends.

An increase in initial samples with increment held constant had a strong impact on final surrogate precision, an early confirmation of basic functionality. An increase in MCMC candidate samples was seen to have a positive but very weak effect on final surrogate precision, suggesting that the runtime of MCMC on each iteration can be limited for increased efficiency. The most complex dynamics arose with the adjustment of sample increment, shown in Figure 14. For each tested initial sample quantity N, the optimal number of step samples was seen to be well-approximated by $\sqrt{N}$. The plotted error trends suggest that incremental samples larger than this optimum give slower model improvement on both the training and evaluation sets, and a larger minimum error on the evaluation set. This performance distinction is predicted to be even more significant when trained on the expensive MC model, where the number of sample evaluations will serve as the primary bottleneck for computation time.

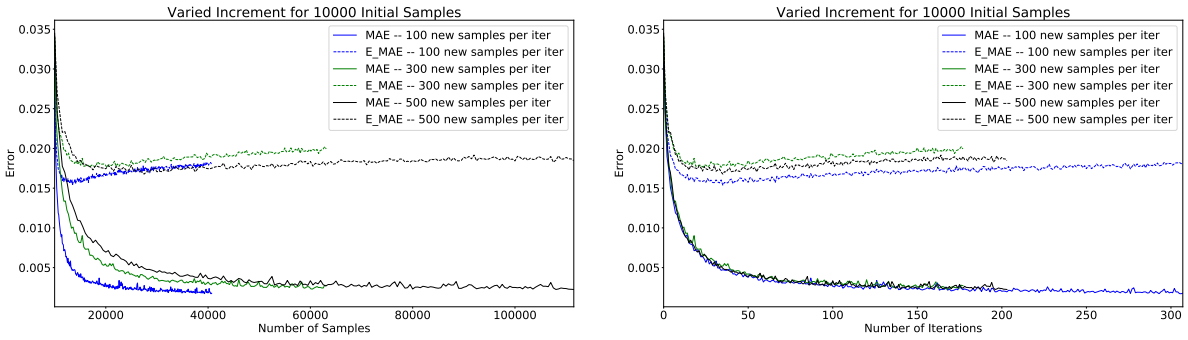


Figure 14: QASS absolute training error over total sample quantity (left) and number of iterations (right). MAE represents surrogate error on the adaptively-sampled training/test set, and E_MAE on the independent evaluation sets.

The plateau effect in surrogate error on the evaluation set, seen in Figure 14, was universal to all configurations and thought to warrant further investigation. At first this was suspected to be a residual effect of retraining the same ANN instance without adjustment to data normalisation. A "Goldilocks scheme" for checking normalisation drift was implemented and tested, but did not affect QASS performance. Schemes in which the ANN is periodically retrained were also discarded, as the retention of network weights from one iteration to the next was demonstrated to greatly benefit QASS efficiency. Further insight came from direct comparison between QASS and a baseline scheme with uniformly random incremental samples, shown in Figure 15.

Such tests revealed that while QASS has unmatched performance on its own adaptively-sampled training set, it is outperformed by the baseline scheme on uniformly-random evaluation sets. We suspected that while QASS excels in learning the most strongly peaked regions of the TBR theory, this comes at the expense of precision in broader, smoother regions where uniformly random sampling suffices. Therefore a mixed scheme was implemented, with half MCMC samples and half uniformly random samples incremented on each iteration, which is also shown in Figure 15. An increase in initial sample size was observed to also resolve precision in these smooth regions of the toy theory, as the initial samples were obtained from a uniform random distribution. As shown in Figure 16, with 100 000 initial samples it was possible to obtain a ∼40% decrease in error as compared to the baseline scheme, from 0.0025 to 0.0015 mean averaged error. Comparing at the point of termination for QASS, this corresponds to a ∼6% decrease in the number of total samples needed to train a surrogate with the same error.

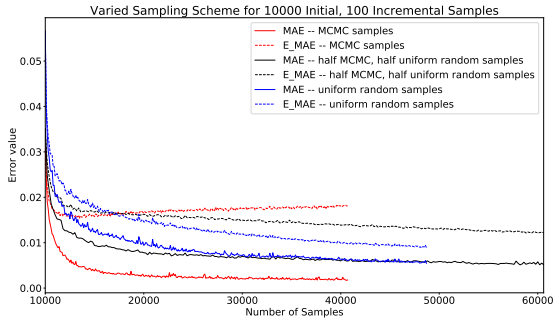


Figure 15: Absolute training error for QASS, baseline scheme, and mixed scheme.

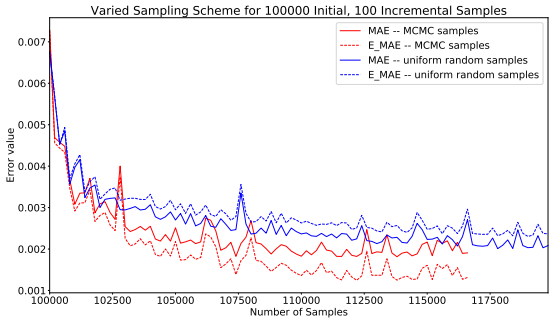


Figure 16: Absolute training error for QASS and baseline scheme, with 100k initial samples.

5 Conclusion

Over the course of this project, we employed a broad spectrum of data analysis and machine learning techniques to develop fast and high-quality surrogates for a MC TBR simulation in use at UKAEA. Having implemented a sampling software to efficiently evaluate this expensive MC model, we deployed it on a high performance cluster to generate over 900 000 datapoints for training and test purposes. We investigated possibilities for simplification of the parameter space, and concluded that no straightforward reduction was possible. After reviewing 9 surrogate model families, examining their behaviour on constrained and unrestricted feature space, and studying their scaling properties, we retrained the best-performing instances to produce properties desirable for practical use. The fastest surrogate, an artificial neural network trained on 500 000 datapoints, featured $R^2 = 0.985$ with mean prediction time of 0.898 µs, representing a relative speedup of $8 \cdot 10^6$ with respect to the MC model. Alternatively, we also demonstrated the possibility of achieving comparable results using only a training set of size 10 000.

After a thorough review of the literature, we developed a novel adaptive sampling algorithm, QASS, capable of interfacing with any of the studied surrogates. Preliminary testing on a toy theory, qualitatively comparable to the MC TBR model, demonstrated the effectiveness of QASS and behavioural trends consistent with the design of the algorithm. With 100 000 initial samples and 100 incremental samples per iteration, QASS achieved a ∼40% decrease in surrogate error as compared with a baseline random sampling scheme. Equivalently, the same surrogate error as the baseline could be achieved with ∼6% fewer total samples of the expensive theory. Further optimisation over the hyperparameter space has strong potential to increase this performance, in particular by decreasing the required quantity of initial samples. This will allow for future deployment of QASS in coalition with any of the most effective identified surrogates to facilitate learning during evaluation of the MC TBR model.

Acknowledgements

The authors would like to thank Simeon Bash, Vignesh Gopakumar, Prof. Nikolaos Konstantinidis, Nikolaos Nikolaou, Prof. Emily Nurse, Jonathan Shimwell and Ingo Waldmann for their supervision and valuable suggestions related to this work.

References

[1] Jacob Søndergaard. "Optimization Using Surrogate Models". PhD thesis. Technical University of Denmark, 2003.

[2] R.H. Myers and D.C. Montgomery. *Response Surface Methodology: Product and Process Optimization Using Designed Experiments*. 2nd. New York: John Wiley & Sons, 2002.

[3] F.A. Hernández and P. Pereslavtsev. "First principles review of options for tritium breeder and neutron multiplier materials for breeding blankets in fusion reactors". In: *Fusion Engineering and Design* 137 (Dec. 2018), pp. 243–256. ISSN: 0920-3796.

[4] M Keilhacker. "JET deuterium: tritium results and their implications". In: *Philosophical Transactions of The Royal Society A: Mathematical, Physical and Engineering Sciences* 357 (Mar. 1999), pp. 415–442.

[5] M. Coleman and S. McIntosh. "BLUEPRINT: A novel approach to fusion reactor design". In: *Fusion Engineering and Design* 139 (Feb. 2019), pp. 26–38. ISSN: 0920-3796.

[6] Paul K. Romano et al. "OpenMC: A state-of-the-art Monte Carlo code for research and development". In: *Annals of Nuclear Energy* 82 (2015). Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013, SNA + MC 2013. Pluri- and Trans-disciplinarity, Towards New Modeling and Numerical Simulation Paradigms, pp. 90–97. ISSN: 0306-4549.

[7] Ian T Jolliffe and Jorge Cadima. "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (Apr. 2016), p. 20150202.

[8] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] Mohamed Amine Bouhlel and Joaquim Martins. "Gradient-enhanced kriging for high-dimensional problems". In: *Engineering with Computers* (Feb. 2018).

[10] Georges Matheron. "Principles of geostatistics". In: *Economic Geology* 58.8 (Dec. 1963), pp. 1246–1266. ISSN: 0361-0128.

[11] *Kriging Variogram Model*. URL: https://vsp.pnnl.gov/help/Vsample/Kriging%7B%5C_%7DVariogram%7B%5C_%7DModel.htm (visited on 20/04/2020).

[12] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 0893-6080.

[13] Rong-En Fan et al. "LIBLINEAR: A library for large linear classification". In: *Journal of machine learning research* 9.Aug (2008), pp. 1871–1874.

[14] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[15] JH Friedman. *Stochastic Gradient Boosting Technical Report*. 1999.

[16] Trevor Hastie, Robert Tibshirani and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[17] Pierre Geurts, Damien Ernst and Louis Wehenkel. "Extremely randomized trees". In: *Machine learning* 63.1 (2006), pp. 3–42.

[18] Harris Drucker. "Improving regressors using boosting techniques". In: *ICML*. Vol. 97. 1997, pp. 107–115.

[19] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

[20] François Chollet et al. *Keras*. https://keras.io. 2015.

[21] Donald Shepard. "A two-dimensional interpolation function for irregularly-spaced data". In: *Proceedings of the 1968 23rd ACM national conference*. 1968, pp. 517–524.

[22] Mohamed Amine Bouhlel et al. "A Python surrogate modeling framework with derivatives". In: *Advances in Engineering Software* (2019), p. 102662. ISSN: 0965-9978.

[23] Jonas Močkus. "On Bayesian methods for seeking the extremum". In: *Optimization techniques IFIP technical conference*. Springer. 1975, pp. 400–404.

[24] Sushant Garud, Iftekhar Karimi and Markus Kraft. "Smart Sampling Algorithm for Surrogate Model Development". In: *Computers & Chemical Engineering* 96 (Oct. 2016).

[25] Jie Zhang, Souma Chowdhury and Achille Messac. "An adaptive hybrid surrogate model". In: *Structural and Multidisciplinary Optimization* 46.2 (2012), pp. 223–238. ISSN: 1615-1488.

[26] Jun Zhou, Xiaosi Su and Geng Cui. "An adaptive Kriging surrogate method for efficient joint estimation of hydraulic and biochemical parameters in reactive transport modeling". In: *Journal of Contaminant Hydrology* 216 (Sept. 2018), pp. 50–57. ISSN: 0169-7722.

[27] Wei Gong and Qingyun Duan. "An adaptive surrogate modeling-based sampling strategy for parameter optimization and distribution estimation (ASMO-PODE)". In: *Environmental Modelling & Software* 95 (Sept. 2017), pp. 61–75. ISSN: 1364-8152.

[28] Jiangjiang Zhang et al. "Surrogate-Based Bayesian Inverse Modeling of the Hydrological System: An Adaptive Approach Considering Surrogate Approximation Error". In: *Water Resources Research* 56.1 (Jan. 2020), e2019WR025721. ISSN: 0043-1397.

[29] Victor Ginting et al. "Application of the two-stage Markov chain Monte Carlo method for characterization of fractured reservoirs using a surrogate flow model". In: *Computational Geosciences* 15.4 (2011), p. 691. ISSN: 1573-1499.

[30] Antti Solonen et al. "Efficient MCMC for Climate Model Parameter Estimation: Parallel Adaptive Chains and Early Rejection". en. In: *Bayesian Anal.* 7.3 (2012), pp. 715–736. ISSN: 1936-0975.

[31] Jerome Sacks, Susannah B Schiller and William J Welch. "Designs for computer experiments". In: *Technometrics* 31.1 (1989), pp. 41–47.

[32] Herman Wold. "Soft modelling by latent variables: the non-linear iterative partial least squares (NIPALS) approach". In: *Journal of Applied Probability* 12.S1 (1975), pp. 117–142.

[33] Mohamed Amine Bouhlel et al. "Improving kriging surrogates of high-dimensional design models by Partial Least Squares dimension reduction". In: *Structural and Multidisciplinary Optimization* 53.5 (2016), pp. 935–952.

[34] Mohamed Amine Bouhlel et al. "An improved approach for estimating the hyperparameters of the kriging model for high-dimensional problems through the partial least squares method". In: *Mathematical Problems in Engineering* 2016 (2016).

Appendix A Online Materials Overview

In this appendix, we provide a brief overview of datasets, software, documentation and other resources available online for public use. All files related to this project are hosted by GitHub in a group at `https://github.com/ucl-tbr-group-project`, organised as follows:

Simple Sphere Study OpenMC TBR simulation software used to generate TBR values.

Sampling Python implementation of the Approximate TBR Evaluator.

Data Over 900 000 datapoints and TBR values generated in sampling runs 0-2.

Regression Python implementation of the surrogate models presented in this work as well as various preprocessing, plotting and evaluation tools.

Hyperparameter Optimisation Results, plots and software commands used to produce them.

Documentation Notes, extended bibliography, this LaTeX document and source files.

Unless specified otherwise, these materials are shared in accordance with the MIT Licence.

Appendix B Detailed Results

		Regression performance				Complexity	
Family	#	MAE [TBR]	S [TBR]	R^2 [rel.]	$R^2_{\text{adj.}}$ [rel.]	$\bar{t}_{\text{trn.}}$ [ms]	$\bar{t}_{\text{pred.}}$ [ms]
SVM	1040	0.078 ± 0.012	0.096 ± 0.013	0.865 ± 0.031	0.862 ± 0.031	0.247 ± 0.014	0.035 ± 0.006
GBT	1000	0.062 ± 0.010	0.062 ± 0.008	0.934 ± 0.007	0.933 ± 0.007	3.253 ± 0.618	0.004 ± 0.000
ERT	1000	0.076 ± 0.007	0.077 ± 0.004	0.898 ± 0.005	0.896 ± 0.005	3.257 ± 1.698	0.125 ± 0.041
ABT	1000	0.146 ± 0.016	0.097 ± 0.007	0.732 ± 0.026	0.726 ± 0.027	0.182 ± 0.015	0.014 ± 0.001
GPR	1000	0.150 ± 0.020	0.137 ± 0.015	0.642 ± 0.041	0.635 ± 0.042	0.241 ± 0.001	0.075 ± 0.001
KNN	1000	0.143 ± 0.020	0.136 ± 0.019	0.661 ± 0.045	0.654 ± 0.046	0.002 ± 0.000	0.744 ± 0.030
ANN	461	0.072 ± 0.008	0.082 ± 0.008	0.895 ± 0.020	0.893 ± 0.021	26.211 ± 8.408	0.294 ± 0.027
IDW	1000	0.151 ± 0.020	0.140 ± 0.020	0.631 ± 0.047	0.623 ± 0.048	0.001 ± 0.000	0.290 ± 0.028
RBF	1000	0.078 ± 0.012	0.086 ± 0.012	0.881 ± 0.028	0.878 ± 0.029	0.193 ± 0.003	0.089 ± 0.001

Table 7: Results of experiment 1 (single slice hyperparameter tuning) as means and standard deviations over 4 tested slices. Column # gives the number of Bayesian optimisation iterations. While regression performance is reported for the best instance (in R^2) per surrogate family, complexity is measured over all tested instances.

		Regression performance				Complexity	
Family	#	MAE [TBR]	S [TBR]	R^2 [rel.]	$R^2_{\text{adj.}}$ [rel.]	$\bar{t}_{\text{trn.}}$ [ms]	$\bar{t}_{\text{pred.}}$ [ms]
SVM	1000	0.091	0.113	0.820	0.818	0.522 ± 0.235	0.201 ± 0.042
GBT	581	0.070	0.072	0.914	0.913	14.431 ± 42.075	0.006 ± 0.003
ERT	901	0.086	0.087	0.871	0.870	3.729 ± 12.784	0.169 ± 0.050
ABT	1000	0.179	0.121	0.602	0.596	0.208 ± 0.075	0.014 ± 0.005
GPR	1000	0.268	0.186	0.091	0.078	0.985 ± 0.014	0.278 ± 0.008
KNN	1000	0.182	0.153	0.519	0.512	0.003 ± 0.000	4.525 ± 4.181
ANN	760	0.061	0.071	0.924	0.923	4.453 ± 3.510	0.014 ± 0.004
IDW	1000	0.194	0.162	0.454	0.446	0.001 ± 0.000	0.768 ± 0.015
RBF	1000	0.083	0.089	0.873	0.871	1.011 ± 0.265	0.503 ± 0.143

Table 8: Results of experiment 2 (full feature space hyperparameter tuning), shown analogously to Table 7.

	Regression performance as R^2 [rel.] by cross-validation set size						
Family	1000	2000	5000	10 000	12 000	15 000	20 000
SVM	0.328 ± 0.086	0.452 ± 0.069	0.591 ± 0.050	0.656 ± 0.040	0.667 ± 0.038	0.680 ± 0.036	0.701 ± 0.032
GBT	$\mathbf{0.827 \pm 0.006}$	$\mathbf{0.867 \pm 0.003}$	$\mathbf{0.897 \pm 0.002}$	$\mathbf{0.910 \pm 0.002}$	$\mathbf{0.912 \pm 0.002}$	$\mathbf{0.918 \pm 0.002}$	0.922 ± 0.002
ERT	0.737 ± 0.000	0.778 ± 0.000	0.839 ± 0.000	0.869 ± 0.000	0.876 ± 0.000	0.884 ± 0.000	0.896 ± 0.000
ABT	0.613 ± 0.005	0.608 ± 0.006	0.645 ± 0.007	0.581 ± 0.006	0.582 ± 0.008	0.579 ± 0.004	0.576 ± 0.007
GPR	0.006 ± 0.000	0.010 ± 0.000	0.046 ± 0.000	0.087 ± 0.000	0.106 ± 0.000	0.129 ± 0.000	0.162 ± 0.000
KNN	0.383 ± 0.001	0.410 ± 0.000	0.469 ± 0.000	0.512 ± 0.001	0.520 ± 0.001	0.528 ± 0.001	0.538 ± 0.001
ANN	0.497 ± 0.058	0.667 ± 0.026	0.848 ± 0.022	$\mathbf{0.914 \pm 0.008}$	$\mathbf{0.920 \pm 0.008}$	$\mathbf{0.927 \pm 0.009}$	$\mathbf{0.933 \pm 0.011}$
IDW	0.287 ± 0.001	0.327 ± 0.001	0.409 ± 0.000	0.435 ± 0.000	0.459 ± 0.000	0.471 ± 0.000	0.490 ± 0.000
RBF	0.729 ± 0.006	0.773 ± 0.004	0.815 ± 0.005	0.844 ± 0.007	0.851 ± 0.006	0.856 ± 0.006	0.865 ± 0.006

Table 9: Results of experiment 3 (scaling benchmark) in terms of R^2. The best-performing family (or families) for each set size is highlighted in bold.

	Complexity as $\bar{t}_{\text{trn.}}$ [ms] by cross-validation set size						
Family	1000	2000	5000	10 000	12 000	15 000	20 000
SVM	0.121 ± 0.061	0.224 ± 0.149	0.495 ± 0.356	0.941 ± 0.495	1.123 ± 0.513	1.448 ± 0.570	1.963 ± 0.563
GBT	3.049 ± 0.543	2.820 ± 0.422	2.784 ± 0.339	2.894 ± 0.357	2.984 ± 0.345	3.071 ± 0.319	3.189 ± 0.417
ERT	4.437 ± 0.447	4.239 ± 0.404	3.972 ± 0.310	4.288 ± 0.293	4.322 ± 0.245	4.365 ± 0.249	4.485 ± 0.271
ABT	0.621 ± 0.083	0.477 ± 0.040	0.374 ± 0.047	0.331 ± 0.034	0.324 ± 0.038	0.339 ± 0.034	0.336 ± 0.036
GPR	0.176 ± 0.014	0.324 ± 0.026	0.952 ± 0.051	2.630 ± 0.115	3.482 ± 0.105	4.807 ± 0.116	7.986 ± 0.163
KNN	0.006 ± 0.001	0.005 ± 0.001	0.005 ± 0.001	0.006 ± 0.001	0.007 ± 0.001	0.007 ± 0.001	0.008 ± 0.001
ANN	31.111 ± 1.624	23.480 ± 1.054	12.708 ± 2.332	8.665 ± 0.620	9.344 ± 0.757	8.259 ± 0.499	7.916 ± 0.472
IDW	$\mathbf{0.005 \pm 0.001}$	$\mathbf{0.003 \pm 0.001}$	$\mathbf{0.002 \pm 0.000}$	$\mathbf{0.002 \pm 0.000}$	$\mathbf{0.002 \pm 0.000}$	$\mathbf{0.003 \pm 0.000}$	$\mathbf{0.003 \pm 0.000}$
RBF	0.162 ± 0.009	0.290 ± 0.024	0.860 ± 0.071	2.347 ± 0.116	3.132 ± 0.135	4.491 ± 0.157	7.334 ± 0.260

Table 10: Results of experiment 3 (scaling benchmark) in terms of $\bar{t}_{\text{trn.}}$, displayed analogously to Table 9.

	Complexity as $\bar{t}_{\text{pred.}}$ [ms] by cross-validation set size						
Family	1000	2000	5000	10 000	12 000	15 000	20 000
SVM	0.042 ± 0.010	0.078 ± 0.014	0.162 ± 0.029	0.304 ± 0.042	0.354 ± 0.049	0.457 ± 0.061	0.586 ± 0.086
GBT	$\mathbf{0.017 \pm 0.003}$	$\mathbf{0.016 \pm 0.003}$	$\mathbf{0.013 \pm 0.001}$	$\mathbf{0.011 \pm 0.001}$	$\mathbf{0.011 \pm 0.001}$	$\mathbf{0.011 \pm 0.001}$	$\mathbf{0.011 \pm 0.001}$
ERT	0.369 ± 0.057	0.322 ± 0.049	0.290 ± 0.041	0.315 ± 0.038	0.308 ± 0.063	0.325 ± 0.056	0.347 ± 0.050
ABT	0.065 ± 0.014	0.042 ± 0.005	0.028 ± 0.005	0.025 ± 0.003	0.024 ± 0.003	0.023 ± 0.002	0.022 ± 0.002
GPR	0.054 ± 0.008	0.114 ± 0.012	0.295 ± 0.020	0.584 ± 0.064	0.694 ± 0.062	0.852 ± 0.073	1.170 ± 0.160
KNN	0.226 ± 0.454	0.411 ± 0.819	1.027 ± 1.901	2.216 ± 3.575	2.764 ± 4.271	3.424 ± 5.205	5.340 ± 6.862
ANN	0.171 ± 0.018	0.133 ± 0.066	0.066 ± 0.041	0.037 ± 0.026	0.038 ± 0.027	0.028 ± 0.016	0.025 ± 0.017
IDW	0.191 ± 0.023	0.338 ± 0.036	0.730 ± 0.062	1.432 ± 0.096	1.580 ± 0.130	2.118 ± 0.155	2.854 ± 0.194
RBF	0.079 ± 0.012	0.187 ± 0.021	0.461 ± 0.049	0.926 ± 0.097	1.097 ± 0.121	1.427 ± 0.154	1.884 ± 0.207

Table 11: Results of experiment 3 (scaling benchmark) in terms of $\bar{t}_{\text{pred.}}$, displayed analogously to Table 9.