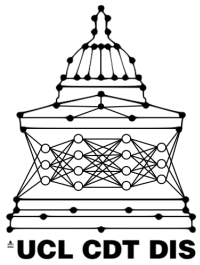


UCL CDT DIS Note

21st April 2020

UK Atomic Energy Authority

Results from TBR Group Project

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

[a]University College London

The abstract of my report.

© 2020 University College London for the benefit of the Centre for Doctoral Training in Data Intensive Sciences. Reproduction of this article or parts of it is allowed as specified in the CC-BY-4.0 license.

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 Evaluation of Supervised Learning Surrogates . . . 11
3.2.1 Experiments . . . 12
3.3 Adaptive Sampling . . . 12

4 Results **13**
4.1 Supervised Model Evaluation . . . 13
4.1.1 Hyperparameter Tuning . . . 14
4.1.2 Scaling Benchmark . . . 14
4.1.3 Competitive Training . . . 14
4.2 Results of Adaptive Sampling . . . 15

5 Conclusion **16**

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 soluble. 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 the underlying expensive model.

In this internship project, in coordination with the UK Atomic Energy Authority (UKAEA) and Culham Centre for Fusion Energy (CCFE), we sought to develop a surrogate model for the tritium breeding ratio (TBR) in a Tokamak nuclear fusion reactor. Our expensive model was a MC-based neutronics simulation [3], itself a spherical approximation of the Joint European Torus (JET) at CCFE, which returns a prediction of the TBR for a given reactor configuration. 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 and CCFE. 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 [4]. Early prototype reactors made use of the deuterium (^{2}H) isotope of hydrogen in order to achieve fusion under more accessible conditions, but achieved limited success. The current frontier generation of fusion reactors, such as JET and the under-construction International Thermonuclear

Experimental Reactor (ITER), make use of tritium (^{3}H) fuel for further efficiency gain. Experimentation at JET dating back to 1997 [5] 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 (deuterium oxide) 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 [4], 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. fig. 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 TBR function which takes these same input parameters and approximates the MC TBR model with the greatest achievable accuracy.

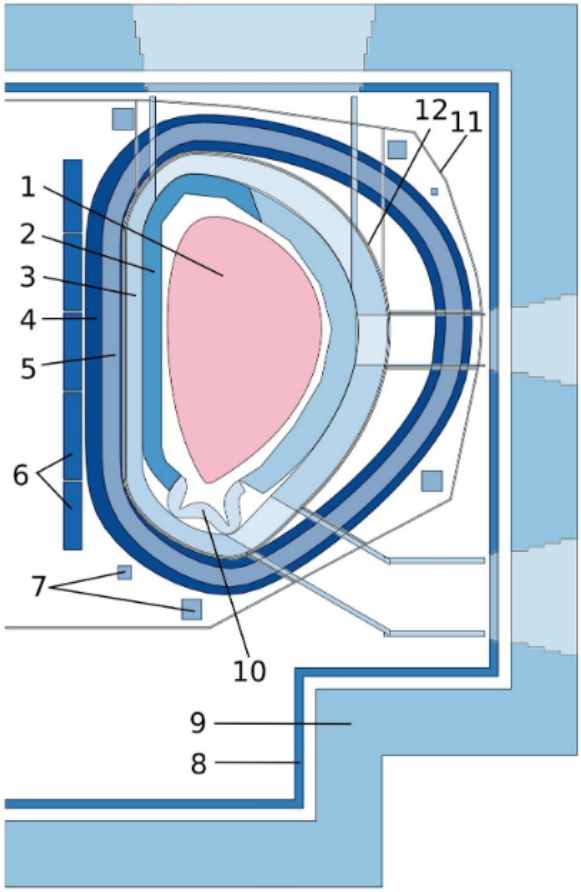


Figure 1: Typical single-null reactor configuration as specified by BLUEPRINT [6]: 1 — plasma, 2 — breeding blankets

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 MC TBR model that we understand to be the expensive function for our surrogate modelling is fundamentally a Monte Carlo simulation based on the OpenMC framework. At input the software expects a set of 19 parameters, discrete and continuous, that are fully listed in table 1. Following a brief period of time (usually on the order of tens of seconds), during which a fixed number of events is simulated, the software outputs the mean and the standard deviation of the TBR aggregated over the simulated run. The former of these two outputs we accept to be the output TBR value that is subject to approximation.

	Parameter name	Acronym	Type	Domain	Description
Blanket	Breeder fraction†	BBF	Continuous	$[0, 1]$	TODO
	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	$[0, 1]$	
	Armour material	FAM	Discrete	$\{tungsten\}$	
	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. Fractions marked with superscripts†‡ are independently required to sum to one.

In our work, 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 inputs, we take steps to unify our feature interface in order to attain this property. In particular, we eliminate discrete features by embedding each such feature into a finite multitude of continuous features 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 renders all features continuous, the modification comes at the expense of increasing the dimensionality of the feature space to 27 features in total. This is further discussed in section 2.3.

2.2 Dataset Generation

While we deliberately make no assumptions about the internal properties of the MC TBR simulation, effectively treating it as a black box model, our means of studying its behaviour are limited to inspection of its outputs at various points in the feature space, and inference thereof. We thus require to gather sufficiently large and representative quantities of samples to ensure statistical significance of our findings.

With grid search in high-dimensional domain clearly intractable, we decided to implement uniform pseudo-random sampling 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 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. We deployed ATE at the UCL Hypatia cluster RCIF partition, which consists of 4 homogeneous nodes, each containing 40 cores. We completed three data generation runs, which are summarised in table 2.

#	Sample count	Job division	Run time (incl. waiting)	Description
0	100 000	100×1000	2 days, 23 hours	Testing run using older MC TBR version.
1	500 000	500×1000	13 days, 20 hours	Fully uniform sampling in the entire domain.
2	400 000	400×1000	10 days	Mixed sampling, discrete features fixed.

Table 2: Parameters of sampling runs.

Skipping run zero, which was performed using older, fundamentally different version of the MC TBR software, and was thus treated as a technical proof-of-concept, we generated the 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 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 between these *slices*, the remaining unassigned features were uniformly sampled only in the first slice, and reused in the other three.

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

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

Since some methods applied in this work are not scale-invariant or perform suboptimally when presented with arbitrarily scaled problems, all results produced in the sampling runs 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.

2.3 Dimensionality Reduction

Model training over high-dimensional parameter spaces may be facilitated 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. Feature extraction methods, on the other hand, aim to identify a transformation from 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.

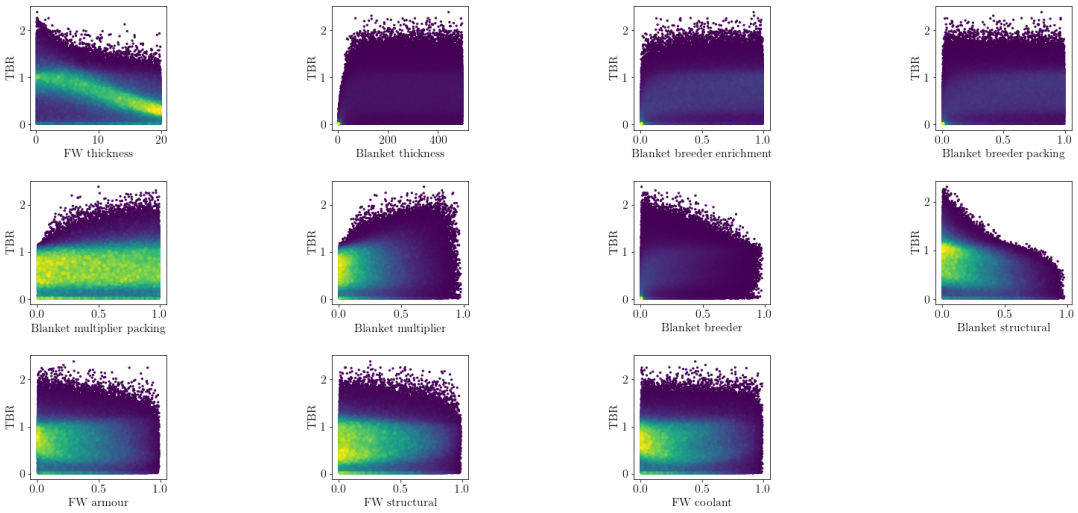


Figure 2: Marginalised dependence of TBR on the choice of continuous features.

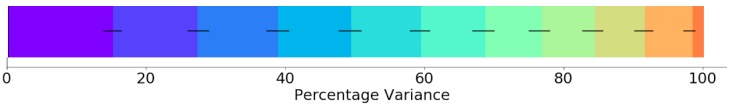
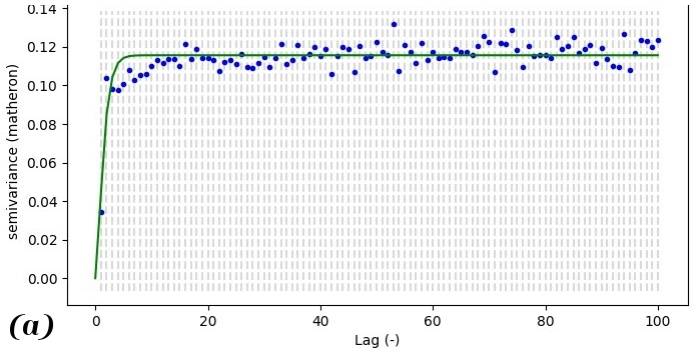


Figure 3: Cumulative variance for optimal features identified by PCA

2.3.1 Principal Component Analysis

To pursue linear feature extraction, principal component analysis (PCA) was performed via scikit-learn on a set of 300,000 uniform samples of the MC TBR model.



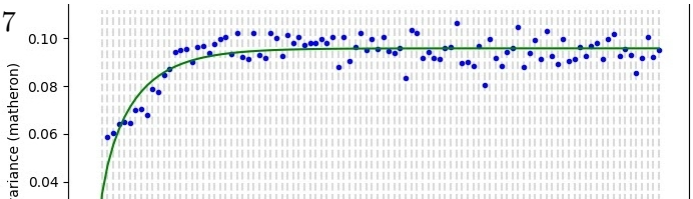


Figure 13 shows the resultant cumulative variance of the feature vectors identified by PCA. The similar share of variance among all eleven features demonstrated irreducibility of the TBR model by linear methods.

2.3.2 Variogram Computations

Kriging, or Gaussian process regression, is a geostatistical surrogate modelling technique which relies on correlation functions over distance (lag) in the paramater space [2]. 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 [3] for three discrete slices with coolant material varied, but all other discrete parameters fixed. Fits [4] 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 are a family of approaches to feature extraction driven by neural networks. Faced with many possible alternatives, we implemented a conventional autoencoder, which relies on a three-layer network that is trained to replicate the identity mapping. 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.

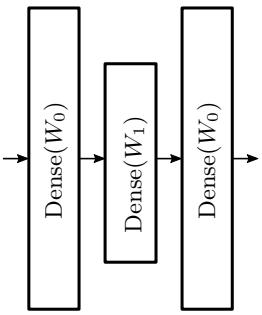


Figure 5: Autoencoder with input width W_0 and bottleneck width W_1.

This approach was applied to two sets of samples: (i) a subset of data obtained from run 1 and (ii) a subset of a single slice obtained from run 2. Our expectation was that while the former case would provide meaningful insight into correlations within the feature space, the latter would validate our autoencoder implementation by analysing a set of points that are trivially reducible due to a number of fixed features.

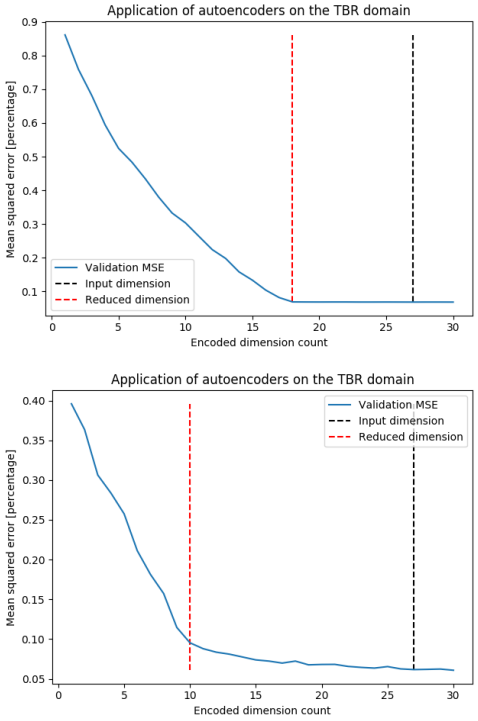


Figure 6: Autoencoder loss in two considered cases of dimensional reduction.

The results of both experiments are shown in fig. 6. Consistent with our motivation, in each plot we can clearly identify a constant plateau in the region of large dimensionality followed by a point, from which a steep increase in loss is observed towards lower dimension counts. We consider this critical point to correspond with the largest viable dimensional reduction without significant information loss. With this methodology we find that the autoencoder was able to reduce the datasets into a subspace of 17 dimensions in the first case and 10 dimensions in the second case.

Confirming our expectation that in the latter, trivially reducible case the autoencoder should achieve stricter dimensional reduction, we are encouraged to believe that our implementation is indeed efficient. However, we must also conclude that in both examined cases this method failed to produce a reduction that would be dominate a naïve approach, in which we eliminate 7 dimensions due to overdetermined one-hot-encoded features, 2 dimensions due to sum-to-one constraints, and in the latter case 7 additional dimensions due to artificially fixed features. This supports the findings from PCA as well as the presented semivariograms.

3 Methodology

Assuming that input has been appropriately treated to eliminate redundant features, we may turn to characterise proposed surrogate models and the criteria used for their evaluation. The task all presented surrogates strive to solve can be formulated using the language of conventional regression problems. In the scope of this work, we explore various possible choices available to us in the scheme of supervised and unsupervised learning.

Labeling the expensive Monte Carlo simulation $f(x)$, 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 similarity metric. Furthermore, in order to be considered *viable*, surrogates are required to achieve expected evaluation time that does not exceed the expected evaluation time of $f(x)$.

In the supervised learning setting, 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 specific model class and appropriate choice of its hyperparameters, surrogate models $\hat{f}(x)$ are trained to minimise empirical risk with respect to $\mathcal{T}$ and a model-specific loss function $\mathcal{L}$, where 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 unsupervised setting can be viewed as an extension of this method. 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 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 by the results of evaluation in order to focus on improvement of surrogate performance in complex regions within the domain.

3.1 Metrics

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

Table 4 provides exhaustive listing and description of metrics recorded in the 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 accuracy of their predictions. In addition, we also track relative measures that are better-suited for model comparison between works as they maintain invariance with respect to the selected domain and image space. For analysis of evaluation time, surrogates are assessed in terms of wall time elapsed during training and predicition. This closely models the practical use case, in which they are trained and used as drop-in replacements for the expensive model. Since training set sizes remain to be determined, all times are reported per a single sample. Even though some surrogates support acceleration by means of parallelisation, measures were taken to ensure sequential processing of samples to achieve comparability between considered models.

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.]
Evaluation time 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	(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 training set of size N_0 and testing 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.

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 standard choice of $k = 5$. Herein, a sample set is subdivided into 5 disjoint folds which are repeatedly interpreted as training and testing sets, maintaining a constant ratio of samples between the two. In each such interpretation experiments are repeated, and the overall value of each metric of interest is reported as the mean across all folds.

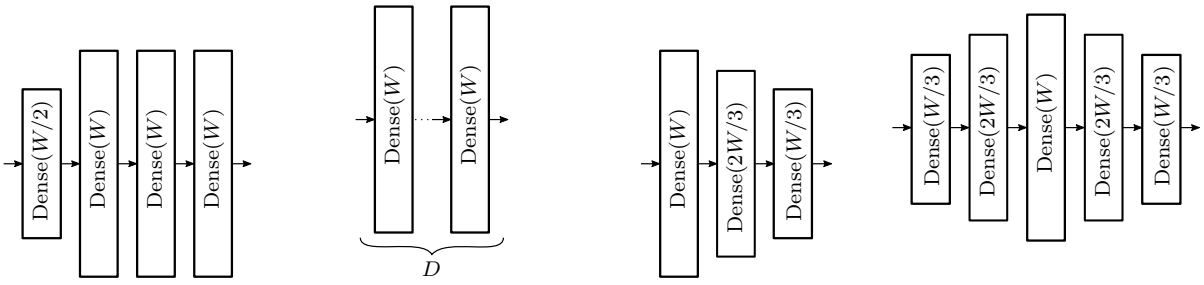
3.2 Evaluation of Supervised Learning Surrogates

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 classes to draw from. 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, randomized algorithms, neural networks and mathematical approaches developed specifically for the purposes of surrogate modelling. For each of these classes, a state-of-the-art implementation was selected and adapted to operate with TBR samples.

Surrogate	Acronym	Implementation	Hyperparameters
Support vector machines	SVM	SciKit Learn	3
Gradient boosted trees	GBT	SciKit Learn	11
Extremely randomized trees	ERT	SciKit Learn	7
AdaBoosted decision trees	AB	SciKit Learn	3
Gaussian process regression	GPR	SciKit Learn	2
k nearest neighbours	KNN	SciKit Learn	3
Neural networks	NN	Keras (TensorFlow)	2
Inverse distance weighing	IDW	SMT	1
Radial basis functions	RBF	SMT	3
Stochastic gradient descent	SGD	SciKit Learn	13
Ridge regression	RR	SciKit Learn	4
Kriging	KRG	SMT	4

Table 5: Considered surrogate model classes.

In some rows of the table, a single class in reality represents a multitude of fundamentally similar approaches. Good examples of this are kriging methods and neural networks. In the case of the former, multiple algorithms such as kriging based on partial least squares and gradient-enhanced kriging are considered. In the latter, a host of parametric graphs are defined to realise simplistic network architecture search (see fig. 7 for details). Discrimination between such options is considered an additional hyperparameter of the corresponding surrogate class.



(a) 1h3f(W) (b) df(D, W) (c) 3pyr(W) (d) 5diam(W)

Figure 7: Selected parametric neural network architectures. All layers except the last use ReLU activation. Prediction information flow is indicated by arrows. Conections between neurons are omitted for clarity.

3.2.1 Experiments

The presented surrogate candidates are evaluated in four experimental cases:

1. Hyperparameter tuning in simplified domain,
2. Hyperparameter tuning in full domain,
3. Scaling benchmark,
4. Competitive surrogate training.

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 is employed as standard hyperparameter tuning algorithm that is considered efficient for this purpose. We set its objective to maximise R^2.

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, accuracy of such models may be replicated at large scale by decomposing the TBR domain into continuous subsets, and learning ensemble surrogate comprised of models that are trained on each subset independently, incurring exponential complexity penalty in the process. To this end, training samples are filtered to only contain a single assignment of discrete features that are subsequently completely withheld from evaluated surrogates. This is repeated four times to investigate variance in behaviour under different feature assignments.

The second experiment conventionally measures surrogate performance on the full TBR regression task. In extension of the previous case, surrogates work with unfiltered samples comprised of discrete as well as continuous features.

The goal of the last two experiments is to exploit the information gathered by hyperparameter tuning for study of scaling properties. In the third experiment, the best-performing hyperparameter choices of each class are used to learn surrogates using progressively larger training sets. Following that, the fourth experiment attempts to produce competitive surrogates by retraining several selected well-scaling models on the largest available data set.

3.3 Adaptive Sampling

All of the surrogate modelling techniques studied in this project face a shared challenge: their accuracy is limited by the quantity of training samples which are available from the expensive MC 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 incrementally so as to best suit the model under consideration.

Adaptive sampling techniques are widespread in the literature and have been specialised for surrogate modelling. Garud's [1] "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 parameter space. MCMC produces a progressive chain of sample points, each drawn according to the same symmetric proposal distribution 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 [2] for a review).

Many researchers have embedded surrogate methods into MCMC strategies for parameter optimisation [3][4], in particular the ASMO-PODE algorithm [5] 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.

We designed the Quality-Adaptive Surrogate Sampling algorithm (QASS) to iteratively increment the training/test set with sample points which maximise surrogate error and minimise a crowding distance metric (CDM) [6] in parameter space. On each iteration following an initial training of the surrogate on N uniformly random samples, the surrogate was trained and AE calculated for . 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.

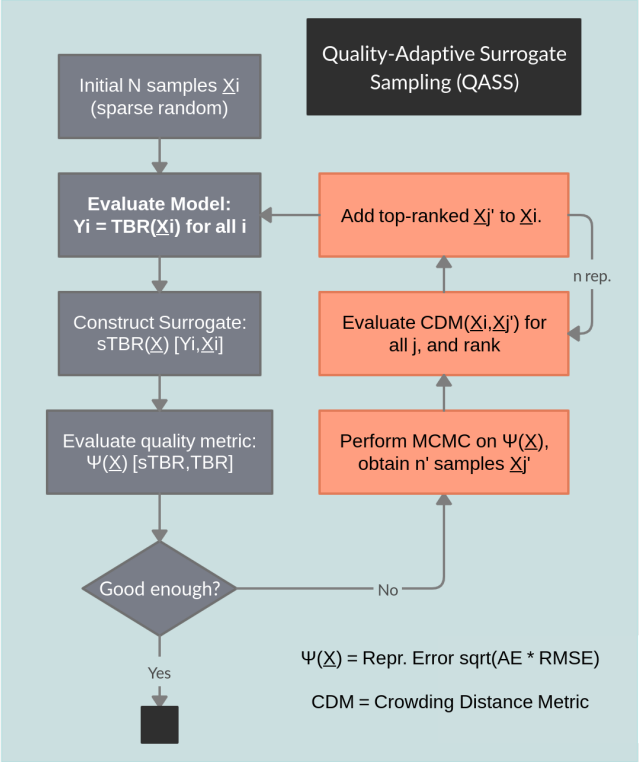


Figure 8: Schematic of QASS algorithm

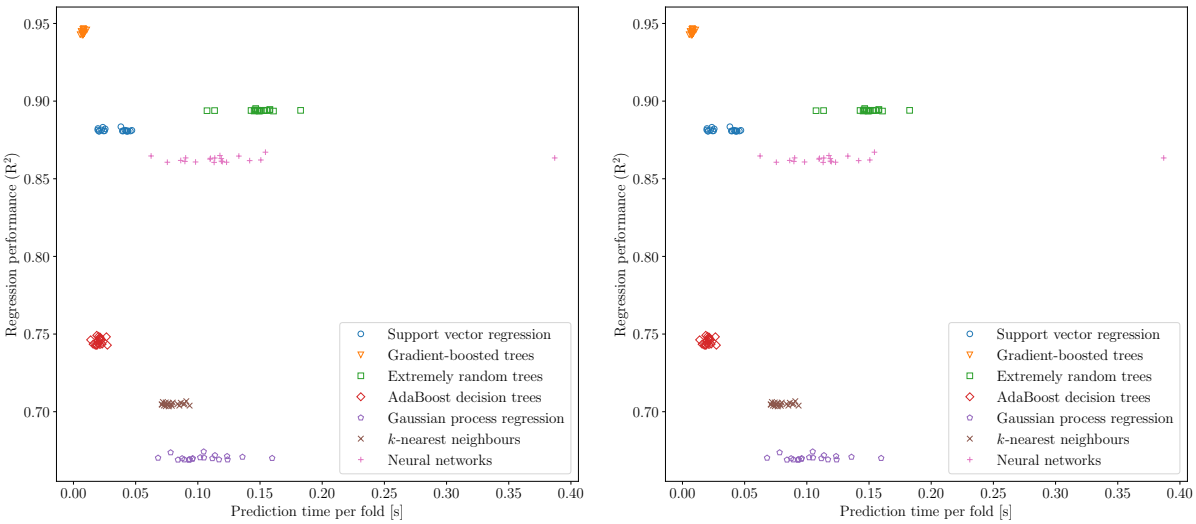
4 Results

4.1 Supervised Model Evaluation

TODO

4.1.1 Hyperparameter Tuning

TODO



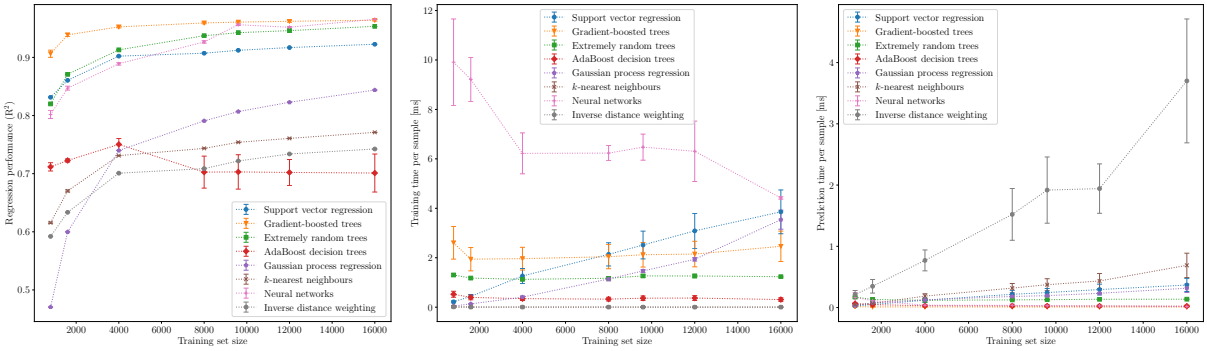
(a) Experiment 1 (single slice)

(b) Experiment 2 (full dataset)

Figure 9: Regression performance (as R^2) plotted against mean prediction time for the best N models of each surrogate class.

4.1.2 Scaling Benchmark

TODO



(a) Regression performance (as R^2)

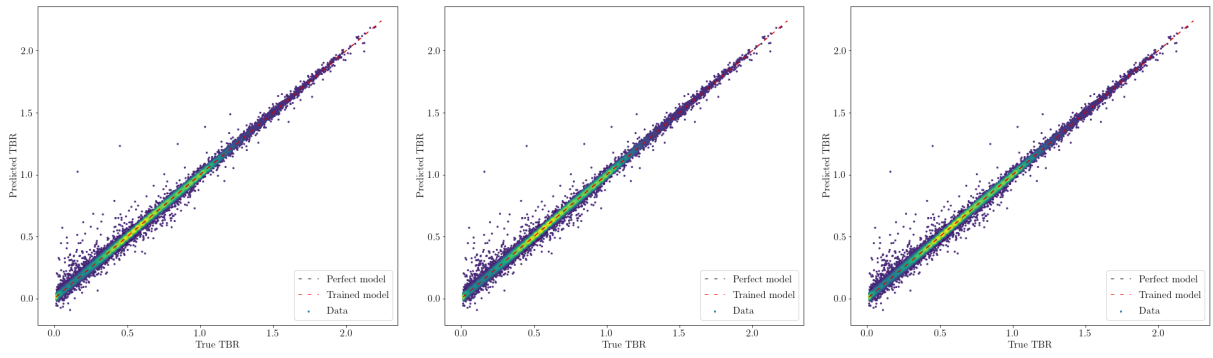
(b) Mean training time

(c) Mean prediction time

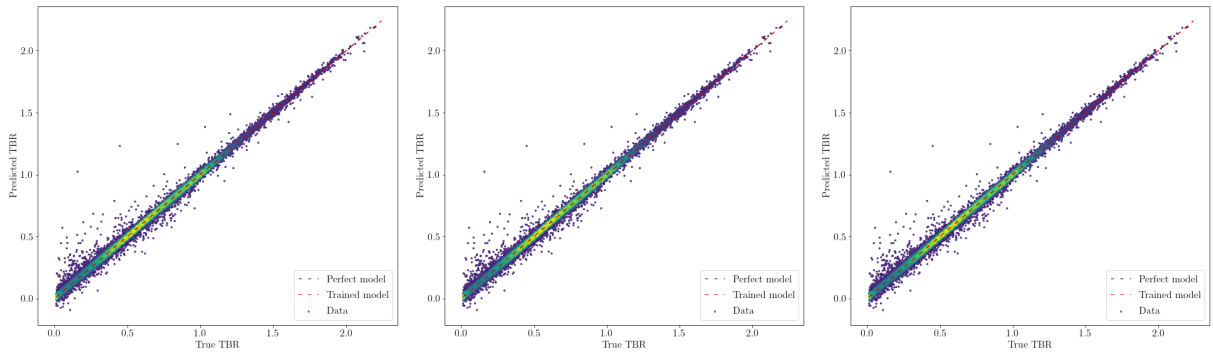
Figure 10: Various metrics collected during experiment 3 (scaling benchmark) displayed as a function of training set size.

4.1.3 Competitive Training

TODO



(a) Model 1 (b) Model 2 (c) Model 3



(d) Model 1 (e) Model 2 (f) Model 3

Figure 11: Regression performance of the best models trained in experiment 4.

4.2 Results of Adaptive Sampling

Define sinusoidal toy model and justify

Explain hyperparameter tests: initsamples, stepsamples, MCMC length

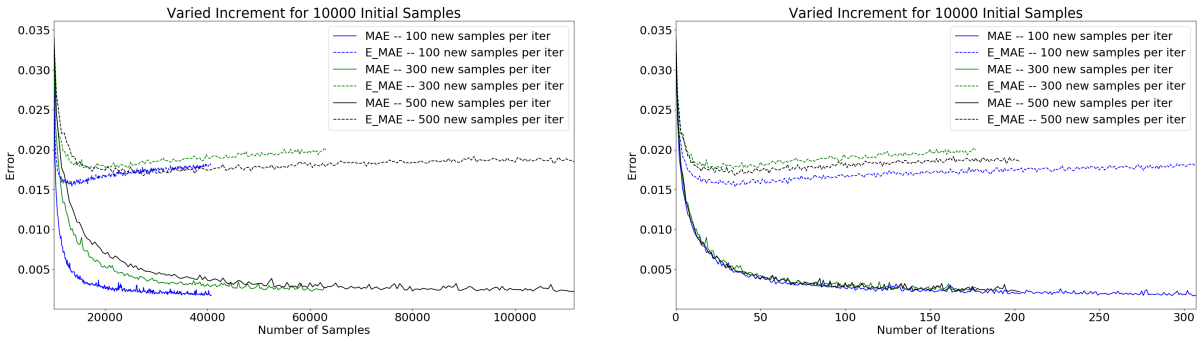


Figure 12: QASS absolute training error over total sample quantity (left) and number of iterations (right)

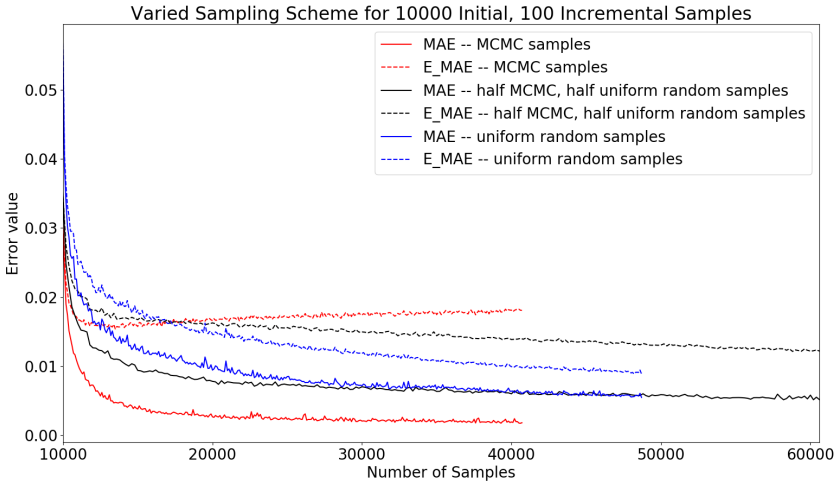


Figure 13: Absolute training error for QASS, uniform random scheme, and mixed scheme

5 Conclusion

Conclusion