Preprint Cambridge Centre for Computational Chemical Engineering ISSN 1473 – 4273

Deep Kernel Learning Approach to Engine Emissions Modelling

Changmin Yu [1], Marko Seslija [1], George Brownbridge [5], Sebastian Mosbach [1,2,5], Markus Kraft [1,2,3,5], Mohammad Parsi [4], Mark Davis [4], Vivian Page [4], Amit Bhave [5]

released: 11 November 2019

[1] Department of Chemical Engineering
and Biotechnology
University of Cambridge
Philippa Fawcett Drive
Cambridge, CB3 0AS
UK

[2] CARES
Cambridge Centre for Advanced Research and
Education in Singapore
#05-05 CREATE Tower
1 CREATE Way
Singapore 138602

[3] School of Chemical
and Biomedical Engineering
Nanyang Technological University
62 Nanyang Drive
Singapore 637459
Email: mk306@cam.ac.uk

[4] Perkins Engines Co. Ltd.
Frank Perkins Way
Peterborough PE1 5FQ
UK

[5] CMCL Innovations
Sheraton House
Castle Park
Cambridge CB3 0AX
UK

Preprint No. 243

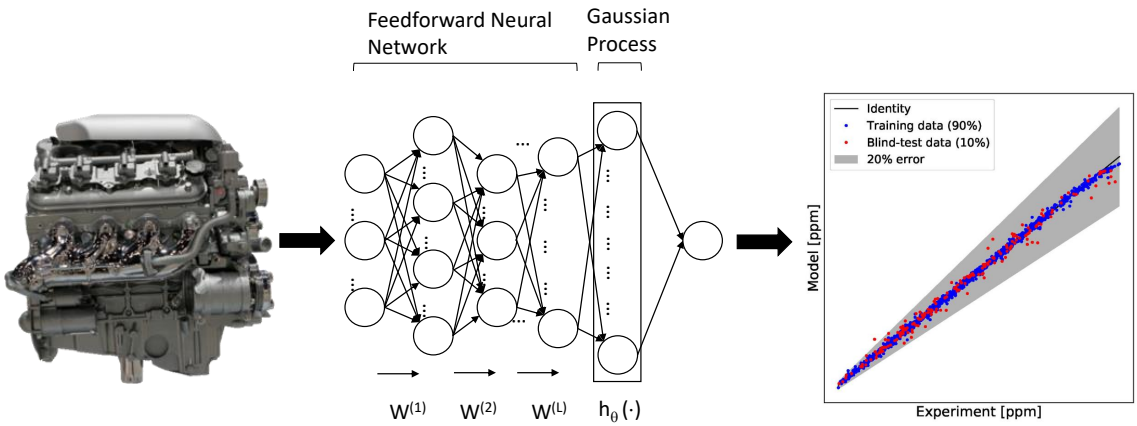
Keywords: Surrogate models, Gaussian processes, deep kernel learning, internal combustion engines, emissions.

Edited by
Computational Modelling Group
Department of Chemical Engineering and Biotechnology
University of Cambridge
West Cambridge Site
Philippa Fawcett Drive
Cambridge CB3 0AS
United Kingdom

Fax: + 44 (0)1223 334796
E-Mail: c4e@cam.ac.uk
World Wide Web: https://como.ceb.cam.ac.uk/

Abstract

We apply Deep Kernel Learning (DKL), which can be viewed as a combination of a Gaussian process (GP) and a deep neural network (DNN), to compression ignition engine emissions and compare its performance to a selection of other surrogate models on the same dataset. Surrogate models are a class of computationally cheaper alternatives to physics-based models. High-Dimensional Model Representation (HDMR) is also briefly discussed and acts as a benchmark model for comparison. We apply the considered methods to a dataset which was obtained from a compression ignition engine and includes as outputs soot and NO_x emissions as functions of 14 engine operating condition variables. We combine a quasi-random global search with a conventional grid-optimisation method in order to identify suitable values for several DKL hyperparameters, which include network architecture, kernel and learning parameters. The performance of DKL, HDMR, plain GPs, and plain DNNs is compared in terms of the root mean squared error (RMSE) of the predictions as well as computational expense of training and evaluation. It is shown that DKL performs best in terms of RMSE in the predictions whilst maintaining the computational cost at a reasonable level, and DKL predictions are in good agreement with the experimental emissions data.



Highlights:

- The application of DKL as a surrogate model to engine emissions is explored.
- A 14-dimensional dataset of emissions from a compression ignition engine is studied.
- Hyperparameters in DKL are systematically optimised.
- DKL outperforms plain GPs, DNNs, and HDMR on the considered dataset.

Contents

1 Introduction

Complex physical systems such as internal combustion engines are generally studied via physical or computational experiments, or their combinations. In many situations, physical experiments may not be feasible or can be highly expensive to perform, and computer experiments are then preferred in such cases. These involve experimenting on a rigorous first-principles or physics-based model instead of the real system [10]. However, expressive computer experiments often involve high computational cost. Therefore, in any application that requires large numbers of data points or model evaluations such as optimisation or parameter estimation, it is inevitable to replace high-fidelity models and real physical experiments by computationally cheap surrogate models.

A surrogate model is an empirical analytical or numerical expression for the quantification of relationships between relevant input features and output labels/values of a physical system [12]. Surrogate models can be viewed as a substitute model which mimics the behaviour of a detailed model or data as closely as possible whilst keeping the computational cost at a minimum. The construction of a surrogate model requires one to choose a mathematical or numerical form for modelling. Surrogate modelling has been studied extensively in applications in numerous areas across science and technology, too numerous to review here, and for a variety of purposes, the most popular ones being parameter estimation [8, 24] and sensitivity analysis [1]. Many statistical and machine learning methods have been proposed as surrogate models in the literature, such as High-Dimensional Model Representation (HDMR) [41, 43], support vector regression (SVR) [7], and radial basis function (RBF) fitting [38], to name a few. The variety of available surrogate modelling techniques is reflective of the fact that there is no such thing as a 'universal surrogate'. Any given technique may work well in some applications, but relatively poorly in others, depending on the unique characteristics in each case, such as dimensionality, oscillatory or discontinuous behaviour, and many others. Learning based Evolutionary Assistive Paradigm for Surrogate Selection (LEAPS2) is a framework that was proposed to recommend the best surrogate(s) with minimal computational effort given the input/output data of a complex physico-numerical system [13]. A frequently encountered issue, particularly in engine applications, is that the input spaces of the datasets are high-dimensional. Another common problem, and again this applies particularly to engine applications, is that the size of the datasets is usually quite limited due to the high cost induced by the data measurement, which further compounds the problem of high dimensionality, although techniques exist to alleviate this [10, 11].

The literature is replete with work related to the application of various machine-learning methods to internal combustion engine modelling, optimisation, and calibration. Ghanbari et al. [14] used support vector machines (SVMs) to predict the performance and exhaust emissions of a diesel engine and showed that SVM modelling is capable of predicting the engine performance and emissions. In the study by Najafi et al. [35], an SVM and an adaptive neuro-fuzzy inference system (ANFIS) are applied to predicting performance parameters and exhaust emissions such as CO_2 and NO_x of a spark ignition (SI) engine and are compared in terms of their performance, and they showed ANFIS is significantly better than SVM. Silitonga et al. [44] applied a method known as kernel-based extreme learning machine (K-ELM) to evaluate the performance and exhaust emissions of

compression ignition (CI) engines with biodiesel-bioethanol-diesel blends at full-throttle conditions. Lughofer et al. [29] investigated the modelling of NO_x emissions of a diesel engine using a fuzzy model directly from measurement data, which is then shown to be a good alternative to physics-based models. Yilmaz et al. [54] compared the response surface methodology (RSM), a commonly used surrogate model, with least-squared support vector machine (LSSVM) based on their performance in predicting the performance and exhaust emissions of a diesel engine fuelled with hazelnut oil, and showed that LSSVM narrowly outperforms RSM. Ghobadian et al. [15] studied the application of a multilayer perceptron (MLP) to predicting exhaust emissions of a diesel engine using waste cooking biodiesel fuel and showed that MLP performs quite well in emissions prediction. Further studies in modelling and predicting the performance and exhaust emissions of diesel engines under different conditions can be found in the literature, such as the work of Niu et al. [37] on the comparison of artificial neural network (ANN) and SVM on emissions prediction of a marine diesel engine, and the study by Wong et al. [52] on using relevance vector machine in modelling and prediction of diesel engine performance. Various studies on the application of ANNs to emissions modelling in diesel engines under different conditions can be found for example in [34, 42, 55].

In the field of machine learning applications in diesel engine modelling, one of the most widely used methods is known as extreme learning machine (ELM), which has inspired many extensions and applications in the diesel engine community since its introduction. Extreme learning machines are feedforward neural networks, with a single hidden layer in most cases [19]. ELMs are an alternative to conventional neural networks in the sense that each hidden unit in an ELM is a computational element, which can be same as classical nodes in an MLP, as well as basis functions or a subnetwork with hidden units [18]. Vaughan and Bohac [47] proposed an online adaptive Extreme Learning Machine named Weighted Ring-ELM, which provides real-time adaptive, fully causal predictions of near-chaotic Homogeneous Charge Compression Ignition (HCCI) engine combustion timing. Janakiraman et al. [21] proposed a stochastic gradient based ELM (SG-ELM), a stable online learning algorithm, designed for systems whose estimated parameters are required to remain bounded during learning. Wong et al. [53] studied the ELM-based modelling and optimisation approach for point-by-point engine calibration. Silitonga et al. [44] studied the application of kernel-based ELM [20] for prediction of the engine performance of biodiesel-bioethanol-diesel blends.

In addition to the above-mentioned studies, there exist a large number of works on the application of machine learning to diesel engine calibration and control. For instance, Tietze [46] studied the application of Gaussian process regression for calibrating engine parameters. Jeong et al. [22] applied a hybrid evolutionary algorithm consisting of a genetic algorithm and particle swarm optimisation to optimise diesel engine design with respect to decreasing exhaust emissions. Berger and Rauscher [4] and Berger et al. [5] discussed various learning methods such as linear regression, feedforward neural networks, and GP regression for modelling and optimisation for stationary engine calibration. Malikopoulos et al. [30] proposed a reinforcement-learning-based decentralised control method which allows an internal combustion engine to learn its optimal calibration in real time while running a vehicle.

In this study, we focus on data-driven engine emissions modelling using deep kernel learn-

ing [51] – a state-of-the-art machine learning technique which can be viewed as a standard deep neural network with a Gaussian process as its last layer instead of a fully connected layer. In this way, we can not only use a deep feedforward network to extract a high-level representation of the data, but also take advantage of the non-parametric flexibility induced by the Gaussian process regression. We implement deep kernel learning for a Diesel engine emission dataset, taking 14 input variables including speed, load, injection timing and others to make predictions of NO_x and soot emissions. We then use a systematic two-stage procedure to determine several of the hyperparameters in DKL, and compare the resulting surrogate to a plain deep feedforward network, a plain Gaussian process, as well as high-dimensional model representation (HDMR).

The paper is structured as follows. In section 2, we discuss in some detail HDMR, deep feedforward networks, Gaussian processes and deep kernel learning, respectively. In section 3, we apply the three methods to the target engine emission data, and compare their performance. Conclusions are drawn in section 4.

2 Methods

2.1 High Dimensional Model Representation

Here, we briefly recall a well-established surrogate modelling method, High Dimensional Model Representation (HDMR), which we use as a reference method for comparison.

HDMR is a finite expansion for a given multivariable function [41]. Under its representation, the output function y can be approximated using the following expression:

$$y \approx f(x) = f_0 + \sum_{i=1}^{N_x} f_i(x_i) + \sum_{i=1}^{N_x}\sum_{j=1}^{N_x} f_{ij}(x_i, x_j), \tag{1}$$

where N_x is the dimension of the input space and f_0 represents the mean value of $f(x)$. The above approximation is sufficient in many situations in practice since terms containing functions of more than two input parameters can often be ignored due to their negligible contributions compared to the lower-order terms [28]. The terms in Eqn. (1) can be evaluated by approximating the functions $f_i(x_i)$ and $f_{ij}(x_i, x_j)$ with some orthonormal basis functions, $\phi_k(x_i)$, that can be easily computed. Popular choices for the basis functions include ordinary polynomials [28] and Lagrange polynomials [2]. Apart from applications in chemical kinetics, HDMR has been applied in process engineering [43] and also in engine emissions modelling [26].

2.2 Deep Neural Networks

Deep learning, or deep artificial neural networks, are composed of multiple processing layers to learn a representation of data with multiple levels of abstraction [27]. Deep learning has gained exploding popularity in the machine learning community over the past two decades, and it has been applied in numerous fields including computer vision,

natural language processing, recommendation systems, *etc.*, due to its ability to find both low- and high-level features of the data as well as its scalability to high-dimensional input spaces. However, the construction of deep learning requires specification of many hyperparameters including number of epochs, regularisation strength, dropout rate, *etc.*, and there is no universal paradigm for determining the optimal settings for these hyperparameters. Hence the performance of a deep neural network can heavily depend on engineering techniques such as architecture specification and parameter tuning.

In our study, we are only concerned with fully-connected neural networks. The goal of a neural network is to learn the underlying representation of the given datasets. A deep feedforward network follows the standard forward propagation and backpropagation to tune the learnable parameters of the neural network, then use the trained network for further predictions on similar problems. In Figs. 1 and 2, we show simple graphical representations for forward and backpropagation in a fully-connected network, respectively.

The architecture of a neural network can be described as a directed graph whose vertices are called neurons, or nodes, and directed edges, each of which has an associated weight. The set of nodes with no incoming edges are called input nodes, whereas the nodes with no outgoing edges are called output nodes. We define the first layer to be the input layer and the last layer to be the output layer, and all other layers in between are called the hidden layers. The input data are propagated in a feedforward fashion as follows:

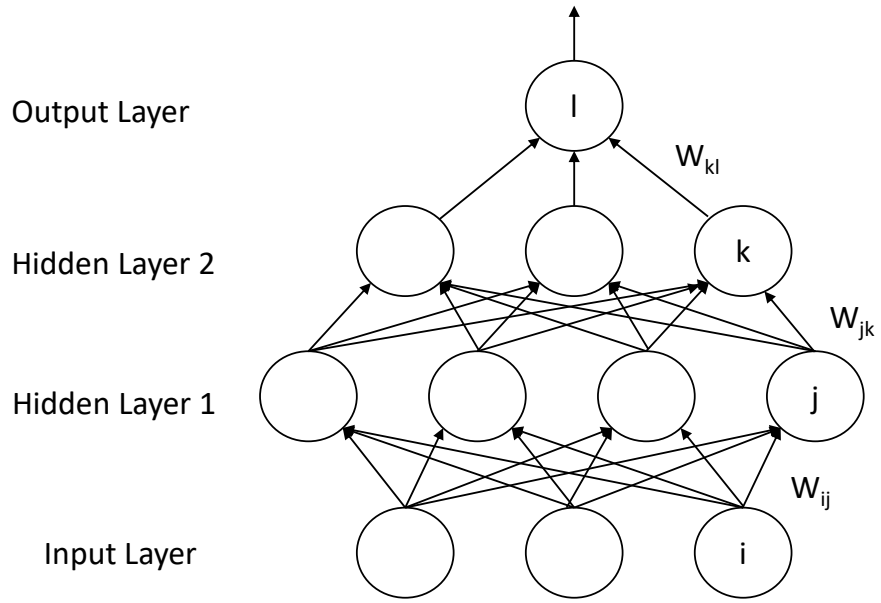


Figure 1: *Forward propagation in a 3-layer feedforward neural network. For each unit in the layers other than the input layer, the output of the unit equals the inner product between all the outputs from the previous layer and the weights followed by a nonlinearity (e.g. the ReLU function).*

Figure 1 shows how input signals are propagated forward in a three-layer fully-connected neural network, where for each unit in any layer other than the input layer, the input to the unit equals the inner product between the output signals from all the units from the

previous layer and the associated weights:

$$z_j = \sum_{i \in \text{previous layer}} w_{ij} y_i$$

The unit then outputs a scalar by applying a nonlinearity, g, to this inner product: $y_j = g(z_j)$. The signals are propagated forward in this way layer by layer until the output layer is reached. Commonly used non-linear activation functions include the rectified linear unit (ReLU) $f(z) = \max(0, z)$, the sigmoid function $\sigma(z) = [1 + \exp(-z)]^{-1}$ and the hyperbolic tangent function $\tanh(z)$. In this case, we have omitted bias terms for simplicity.

At an output node, after taking the linear combination of its predecessor values, instead of applying an activation function, an output rule can be used to aggregate the information across all the output nodes. In a regression problem, as opposed to a classification problem, we simply keep the linear combination as the output prediction.

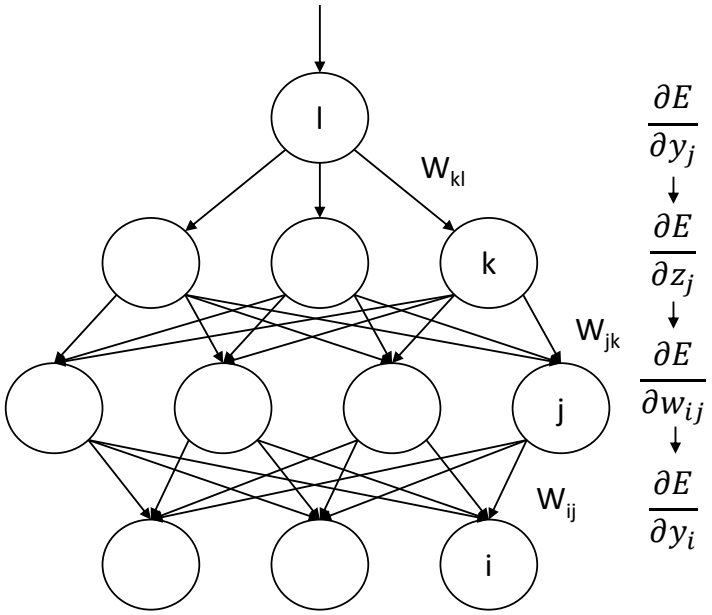


Figure 2: *Backpropagation in a 3-layer feedforward neural network. Computing the derivatives of the cost function with respect to the weight parameters using chain rules, then the parameters are updated using gradient descent with the computed derivatives.*

The weights and bias parameters in neural networks are usually learnt via backpropagation using the cached quantities from the forward propagation stage. Backpropagation, as represented in Fig. 2, is executed using gradient-descent-based optimisation methods. Given a cost function, $E(\hat{y}, y)$, which quantifies the difference between the current predictions of the output ($\hat{y}$) and the actual outputs (y), the derivatives of E with respect to each weight in each layer of the network can be derived using the chain rule,

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_j}, \qquad \frac{\partial E}{\partial w_{ij}} = y_i \frac{\partial E}{\partial z_j}, \tag{2}$$

where the initial $\partial E/\partial y_j$ is derived given the activation function and the cost function. Then we compute $\partial E/\partial z_j$, $\partial E/\partial w_{ij}$ and $\partial E/\partial y_i$ iteratively for each node i in layer l and node j in layer $l-1$ until we reach the input layer. Then we use methods such as stochastic gradient descent to update the weights in the network:

$$w_{ij} \leftarrow w_{ij} - \alpha \frac{\partial E}{\partial w_{ij}}, \tag{3}$$

where α is some scalar known as the learning rate.

Iterative application of forward propagation and backpropagation will gradually decrease the cost function value in general, and if the cost function is convex in the input space, it will eventually converge to the global minimum or somewhere close to the global minimum.

The power of neural networks lies in the composition of the nonlinear activation functions. From the results on universal approximation bounds for superpositions of the sigmoid function by Barron [3], it can be implied that a neural network with sigmoid activation can arbitrarily closely approximate a nonparametric regression mode.

Here we only discuss fully-connected neural networks, where the only learnable parameters are the weight parameters w_{ij} between each node i in layer $l-1$ and each node j in the previous layer l. For a general review of various architectures of deep learning such as convolutional neural networks and recurrent neural networks, the reader is referred to [27] for example.

2.3 Gaussian Processes

A Gaussian process (GP) is a stochastic process, which is a collection of random variables, with the property that any finite subcollection of the random variables have a joint multivariate Gaussian distribution [48]. We denote the fact that a stochastic process $f(\cdot)$ is a Gaussian process with mean function $m(\cdot)$ and covariance function $k(\cdot,\cdot)$ as

$$f(\cdot) \sim \mathcal{GP}\big(m(\cdot), k(\cdot,\cdot)\big). \tag{4}$$

The definition implies that for any $x^{(1)}, x^{(2)}, \ldots, x^{(n)} \in \mathcal{X}$, where $\mathcal{X}$ denotes the set of possible inputs, we have

$$\big[f(x^{(1)}), \ldots, f(x^{(n)})\big]^\top \sim \mathcal{N}\Big(\big[m(x^{(1)}), \ldots, m(x^{(n)})\big]^\top, K\Big), \tag{5}$$

where the covariance matrix K has entries $K_{ij} = k(x^{(i)}, x^{(j)})$.

Gaussian processes can be interpreted as a natural extension of multivariate Gaussian distributions to have infinite index sets, and this extension allows us to think of a Gaussian process as a distribution over random functions. A Gaussian process is fully determined by its mean and covariance functions. The mean function can be any real-valued function, whereas the covariance function has to satisfy that the resulting covariance matrix K for any set of inputs $x^{(1)}, \ldots, x^{(n)} \in \mathcal{X}$ has to be a valid covariance matrix for a multivariate

Gaussian distribution, which implies that K has to be positive semidefinite and this criterion corresponds to Mercer's condition for kernels [31]. Hence the covariance function is sometimes also known as the kernel function. One of the most popular choices of kernel function is the RBF (or squared error) kernel

$$k_{\text{RBF}}(x, x') = \exp\Big(-\frac{\|x - x'\|^2}{2l^2}\Big), \tag{6}$$

where l is a kernel parameter which quantifies the level of local smoothness of the distribution drawn from the Gaussian process.

Gaussian processes have gained increasing popularity in the machine learning community since Neal [36] showed that Bayesian neural networks with infinitely many nodes converge to Gaussian processes with a certain kernel function. Therefore, Gaussian processes can be viewed as a probabilistic and interpretable alternative to neural networks.

In our study, the focus of the application of Gaussian processes lies within regression tasks. Suppose we are given a dataset $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1,\dots,n}$, that one may refer to as the training set, of independent samples from some unknown distribution, where $x^{(i)} \in \mathbb{R}^d$ and $y^{(i)} \in \mathbb{R}$ for $i = 1, \dots, n$. A Gaussian process regression model is then

$$y^{(i)} = f(x^{(i)}) + \epsilon^{(i)}, \quad \text{for } i = 1, \dots, n, \tag{7}$$

with a Gaussian process prior over the functions f, that is $f(\cdot) \sim \mathcal{GP}\big(m(\cdot), k(\cdot, \cdot)\big)$, for some mean function $m(\cdot)$ and valid kernel function $k(\cdot, \cdot)$, and the $\epsilon^{(i)}$ are independent additive Gaussian noises that follow $\mathcal{N}(0, \sigma^2)$ distributions.

Suppose we are given another dataset $\mathcal{D}_* = \{x_*^{(i)}, y_*^{(i)}\}_{i=1,\dots,n_*}$, that one may refer to as the blind-test set, drawn from the same distribution as $\mathcal{D}$. By the definition of a Gaussian process, we have

$$\begin{bmatrix} F \\ F_* \end{bmatrix} \Bigg| X, X_* \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right), \tag{8}$$

where $X = \{x^{(i)}\}_{i=1,\dots,n}$, F represents $[f(x^{(1)}), \dots, f(x^{(n)})]^\top$, $K(X, X)$ represents an $n \times n$ kernel matrix whose (i, j) entry is $K(x^{(i)}, x^{(j)})$, and analogous definitions for quantities with asterisk subscripts, *e.g.* $X_* = \{x_*^{(i)}\}_{i=1,\dots,n_*}$. Then, given the additive Gaussian noises, the joint distribution of $Y = [y^{(1)}, \dots, y^{(n)}]^\top$ and Y_* becomes

$$\begin{bmatrix} Y \\ Y_* \end{bmatrix} \Bigg| X, X_* \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) + \sigma^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) + \sigma^2 I \end{bmatrix}\right). \tag{9}$$

Using the rules for conditioning Gaussian distributions, the posterior distribution of the blind-test data given the training data is then given by

$$\begin{aligned} Y_* | X_*, X, Y &\sim \mathcal{N}(\mu_*, \Sigma_*)\,, \\ \mu_* &= m(X_*) + K(X_*, X)[K(X, X) + \sigma^2 I]^{-1} Y\,, \\ \Sigma_* &= K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma^2 I]^{-1} K(X, X_*)\,. \end{aligned} \tag{10}$$

Eqn. (10) is the posterior Gaussian process regression model for predictions. The kernel parameters θ of the kernel function in the Gaussian process regression model can be learnt by maximising the (log) posterior marginal likelihood

$$\log p(Y|\theta, X) \propto -Y^\top \left(K_\theta(X,X) + \sigma^2 I\right)^{-1} Y - \log \left|K_\theta(X,X) + \sigma^2 I\right| \qquad (11)$$

with respect to the kernel parameters, where we have emphasised the dependence of the kernel matrix K on its parameters θ through a subscript.

2.4 Deep Kernel Learning

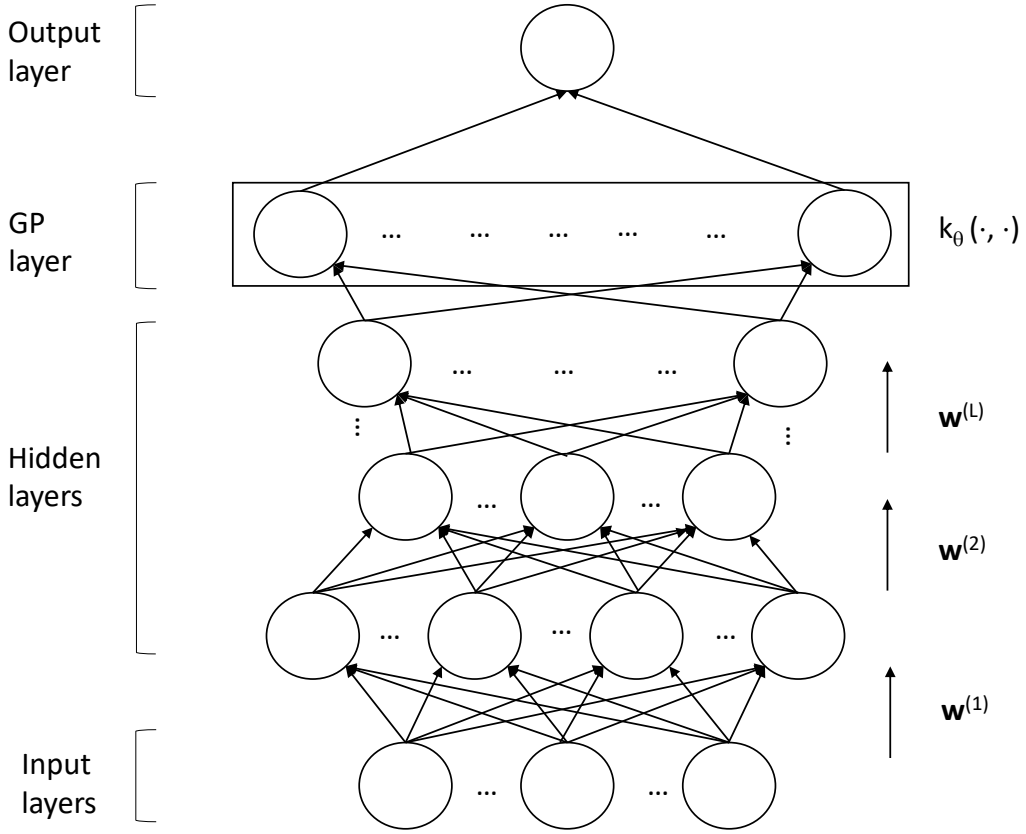


Figure 3: *Deep Kernel Learning: Input data is propagated in a forward fashion through the hidden layers of the neural network parameterised by the weight parameters w. Then, the low-dimensional high-level feature vector as the output of the neural network is fed into a Gaussian process with a base kernel function $k_\theta(\cdot,\cdot)$ for regression. The posterior mean of the Gaussian regression model is taken as the prediction given the input data. (Adapted from [51].)*

We now briefly discuss the main method we apply as a surrogate model, deep kernel learning [51]. Deep kernel learning can be intuitively interpreted as a combination of a deep neural network and a Gaussian process. A graphical representation of a DKL model is shown in Fig. 3, where we can see that the structure consists of a deep neural network

followed by a Gaussian process. As mentioned in the previous section, a Gaussian process is the limit of a Bayesian neural network with an infinite number of nodes, hence the Gaussian process at the end of the DKL architecture can be interpreted as another hidden layer in the deep neural network, but with an infinite number of nodes, and this greatly increases the expressiveness compared to a stand-alone deep neural network. When the data enters the DKL model, it is firstly propagated in a forward fashion through the neural network. The high-dimensional input data is thus transformed by the neural network into a lower-dimensional feature vector, which is then used as the input arguments for Gaussian process regression. The expectation of the resulting posterior distribution is then taken as the value predicted by DKL as a function of the input data. As Gaussian processes naturally do not perform well in high-dimensional input spaces, the deep neural network acts as a feature extractor and dimensionality reduction method for more robust Gaussian process regression. Being a combination of deep learning and kernel learning, DKL encapsulates the expressive power for extracting high-level features and capturing non-stationary structures within the data given its deep architectures and the non-parametric flexibility in kernel learning induced by its probabilistic Gaussian process framework.

We can also view DKL as a Gaussian process with a stand-alone deep kernel. Starting from a base kernel $k(x^{(i)}, x^{(j)}|\theta)$ with kernel parameters θ, the deep kernel can be constructed as

$$k(x^{(i)}, x^{(j)}|\theta) \rightarrow k\big(g(x^{(i)}, w), g(x^{(j)}, w)\big|\theta, w\big)\,, \tag{12}$$

where $g(x; w)$ is a non-linear mapping induced by the neural network with weight parameters w. A popular choice for the base kernel $k(x^{(i)}, x^{(j)}|\theta)$ is again the RBF kernel (Eqn. 6). Inspired by Wilson et al. [51], we also look at the spectral mixture (SM) base kernel

$$k_{\text{SM}}(x, x'|\theta) = \sum_{q=1}^{Q} a_q \frac{|\Sigma_q|^{\frac{1}{2}}}{(2\pi)^{\frac{D}{2}}} \exp\Big(-\frac{1}{2}\big\|\Sigma_q^{\frac{1}{2}}(x - x')\big\|^2\Big) \cos\langle x - x', 2\pi\mu_q\rangle \tag{13}$$

by Wilson and Adams [49], where the learnable kernel parameters $\theta = \{a_q, \Sigma_q, \mu_q\}$ consist of a weight, an inverse length scale, and a frequency vector for each of the Q spectral components, and where $\langle\cdot,\cdot\rangle$ denotes the standard inner product. The spectral mixture kernel is meant to be able to represent quasi-periodic stationary structures within the data.

We denote by $\gamma = \{w, \theta\}$ the parameters of the DKL model, consisting of the neural network weight parameters w and the Gaussian process kernel parameters θ. These parameters are learnt jointly via maximising the log-posterior marginal likelihood of the Gaussian process (Eqn. 11) with respect to γ.

3 Applying DKL to engine emissions

3.1 Dataset

The dataset used in this work was obtained from a Diesel-fuelled compression ignition engine whose main geometric features are provided in Table 1. The data consists of soot and

Table 1: *Specification of the turbocharged 4-stroke Diesel-fuelled compression ignition engine used in this work.*

Quantity	Value
Bore	98 mm
Stroke	120 mm
Connecting rod length	180 mm
Compression ratio	17:1

NO_x emissions taken engine-out during steady-state operation at 1861 distinct operating points. Each of these points is characterised by 14 operating condition variables, which include engine speed and torque, intake manifold temperature, injection pressure, mass fraction of recirculated exhaust gas, start, end, and fuel mass of injection, and combustion chamber wall temperatures. The set of points is spread roughly evenly over the entire engine operating window in terms of speed and load. NO_x emissions are measured in units of parts per million by volume (ppmv) with a nominal error bar of 3% and soot represents the carbon fraction of emitted particulate matter with a stated measurement uncertainty of 5%.

Since the numerical values of the soot response vary over several orders of magnitude, it is necessary to consider their logarithms instead of their raw values. The NO_x response values can be used as is. We split the dataset randomly into disjoint training and blind-test sets which comprise 90% and 10% of the total, respectively.

3.2 Implementation

In all numerical experiments, our DKL implementation consists of a five-layer fully-connected network and a Gaussian process with RBF kernel. The neural network employs the rectified linear unit (ReLU) function [27] as the activation function for each hidden layer, and all the weights are initialised with the He normal initialisation [16]. We use the standard root mean squared error (RMSE) loss function, and the Adam optimiser for optimisation [25].

We implement DKL as a surrogate model into Model Development Suite (MoDS) [6] – an integrated software written in C++ with multiple tools for conducting various generic tasks to develop black-box models. Such tasks include surrogate model creation [43], parameter estimation [24], error propagation [33], and experimental design [32]. Our MoDS-implementation of DKL uses PyTorch [39] and GPyTorch [9].

All simulations were performed on a desktop PC with twelve 3.2 GHz CPU-cores and 16 GB RAM. Even though MoDS allows parallel execution, all simulations were conducted in serial in order to simplify quantification of computational effort. For the same reason, no GPU-acceleration of Torch-based code was considered. We also did not explore the use of kernel interpolation techniques [40, 50] to speed up GP learning.

3.3 Network architecture, kernel, and learning parameters

The DKL framework involves learnable parameters such as network weights and kernel parameters, as well as hyperparameters such as the learning rate, number of iterations, and number of nodes in each layer of the neural network. Before training can be carried out, suitable values of the hyperparameters need to be chosen. We approach this in two ways: Firstly, we make this choice manually, based on previous experience and cross-validation over a small hyperparameter search-space, and secondly, we employ a systematic, optimisation-based procedure. In both cases, we determine the following seven hyperparameters: the number of nodes in each of the four hidden layers, the prior white-noise level of the Gaussian process, the number of epochs, *i.e.* training iterations, and the learning rate.

Table 2: *DKL hyperparameters considered for optimisation.*

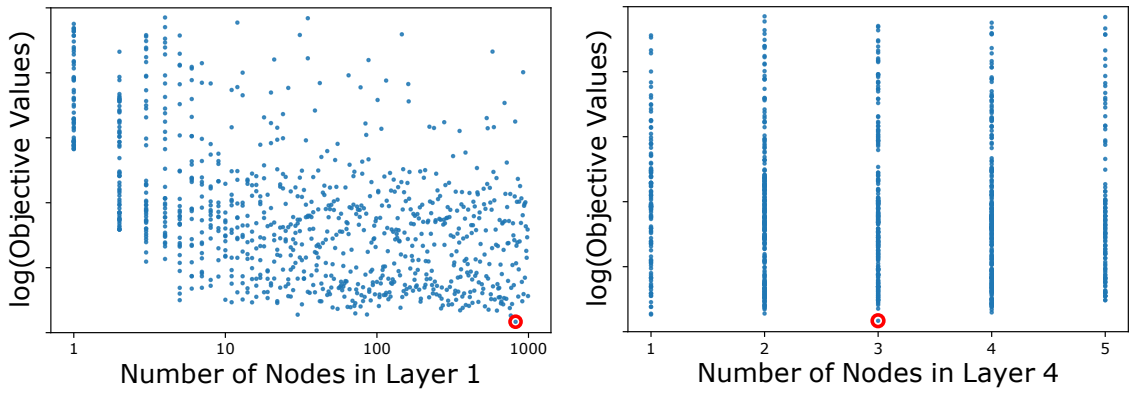
Parameters	Lower Bound	Upper Bound	Scaling	Rounded
N nodes Layer 1	1	1000	Log	Yes
N nodes Layer 2	1	1000	Log	Yes
N nodes Layer 3	1	1000	Log	Yes
N nodes Layer 4	1	15	Linear	Yes
White noise scale	0.0	0.5	Linear	No
Number of epochs	10	1000	Log	Yes
Learning rate	0.001	0.1	Log	No

The optimisation approach consists of two stages: a global quasi-random search followed by a local optimisation with a gradient-free grid-based method, both of which are conducted using MoDS. For the first stage, a Sobol sequence [23], a low-discrepancy sampling method, is employed. We generate 1000 Sobol points within the space spanned by all of the hyperparameters given in Table 2, which also provides the range and scaling type for each parameter. We then fit the DKL under each of these 1000 hyperparameter settings to the training data. The quality of each fit is assessed by calculating the objective function

$$\Phi(\theta) = \Big[\frac{1}{N_\mathrm{b}-1}\sum_{i\in\mathcal{I}_\mathrm{b}}\big(f(x^{(i)})-y^{(i)}\big)^2\Big]^2 + \Big[\frac{1}{N_\mathrm{t}-1}\sum_{i\in\mathcal{I}_\mathrm{t}}\big(f(x^{(i)})-y^{(i)}\big)^2\Big]^2, \qquad (14)$$

where f denotes the surrogate model, *i.e.* the trained DKL, and $x^{(i)}$ and $y^{(i)}$ the experimental operating conditions and responses (soot or NO_x), respectively, of the i^{th} data point. $\mathcal{I}_\mathrm{b}$ denotes the set of indices belonging to the blind-test data points, and N_b denotes their number, whereas $\mathcal{I}_\mathrm{t}$ and N_t refer to the analogous quantities for the training data points. The normalisation of the two parts of the objective function, *i.e.* the training and blind-test parts, by the number of points they contain, implies that the two parts are equally weighted with respect to each another, irrespective of how many points they contain. We tested other forms of the objective function but found empirically that this form yields the best results. Furthermore, we note that including the blind-test points into the objective function is not a restriction. In any application, whatever set of points is given,

it can arbitrarily be split into training and blind-test subsets. Again, we made this choice because we found empirically that it produces the best results.



(a) *Objective against number of nodes in layer 1. DKL performs relatively poorly for small numbers of nodes in hidden layer 1.*

(b) *Objective against number of nodes in layer 4. Best performance is found for 3 extracted features, i.e. 3 nodes in layer 4.*

Figure 4: *Combined training and blind-test objective function value (Eqn. 14) for 1000 Sobol points in the space of hyperparameters of Table 2. The point with the lowest objective value overall is circled.*

Scatter plots of the Sobol points, showing their objective values against two of the architecture parameters, are given in Fig. 4. In Fig. 4a, we observe that best performance is achieved if the number of nodes in the first layer well exceeds the number of inputs. From Fig. 4b, we conclude that the number of features extracted by the neural network part of DKL, *i.e.* the number of nodes in the fourth hidden layer or in other words the number of quantities fed as input to the Gaussian process, for which the objective function attains its minimum is three. The objective value deteriorates appreciably for four and five features.

The best Sobol point as measured by Eqn. (14) (highlighted in Fig. 4) is then optimised further with respect to the white noise of GP, number of epochs and learning rate using Hooke and Jeeve's algorithm [17] – a gradient-free grid-based optimisation algorithm which is also part of MoDS. Since both the Sobol and Hooke-Jeeves method are designed for continuous variables, we treat the discrete parameters, *i.e.* the number of nodes in each layer and the number of epochs, internally as continuous and simply round their values to the nearest integers when passing them on to DKL. This is also indicated in Table 2. Other optimisation methods more suitable for discrete problems, such as genetic algorithms, are expected to perform at least as well, however it is beyond the scope of the present work to explore this.

We note that the Hooke-Jeeves optimisation step achieves only a relatively small improvement upon the best Sobol point, with the algorithm terminating after 68 and 78 iterations for NO_x and soot, respectively. However, one should bear in mind that DKL training itself being based on stochastic optimisation, and thus noise inherently being present in the quality of the fit, presents a challenge to any local optimisation method. We furthermore find that there is little to no benefit in including the architecture parameters of the network, *i.e.* the number of nodes in the hidden layers, into the Hooke-Jeeves optimisation, so we

excluded them from this stage.

The average CPU-time for (serial) evaluation of each Sobol or Hooke-Jeeves point was about 1 minute for NO_x and 2 minutes for soot. The reason for the larger evaluation time for soot is that the optimisation favoured a number of epochs on average about twice as high for soot as for NO_x.

Table 3 shows the best values for the hyperparameters we find using the procedure described above. We note that due to the random nature of both the global search and the DKL training process itself, our procedure does not guarantee a global minimum. The

Table 3: *Best values found for the hyperparameters in DKL through optimisation.*

Output	N nodes Layer 1	N nodes Layer 2	N nodes Layer 3	N nodes Layer 4	White noise scale	Number of epochs	Learning rate
NO_x	822	46	356	3	0.39	210	0.019
Soot	690	5	6	3	0.48	466	0.026

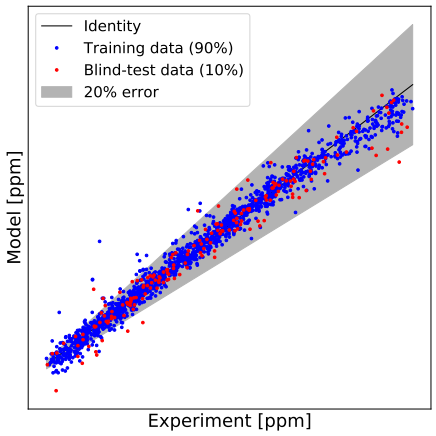
best values found manually for the hyperparameters in DKL are 1000, 500, 50, and 3 for the numbers of nodes in the hidden layers, a white-noise level of 0.1, 200 training epochs, and a learning rate of 0.01.

3.4 Comparison

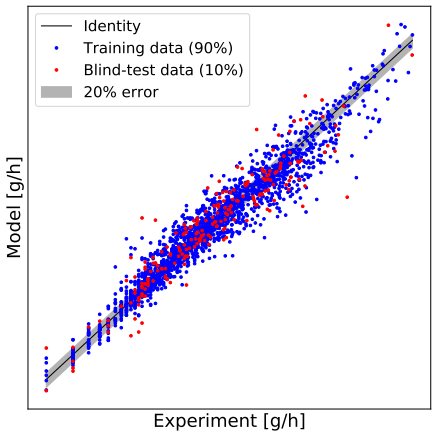
Table 4: *Percentage of predictions within 20% of the experimental value and root mean square errors of NO_x and soot regressions for HDMR and DKL.*

Output	Surrogate	Percentage w/in 20% of exper.			RSME	
		Training	Blind test	Total	Training	Blind test
NO_x	HDMR	94.9%	90.3%	94.5%	44.78	69.17
	DKL (man.)	99.7%	99.4%	99.6%	22.07	38.93
	DKL (opt.)	99.9%	99.5%	99.9%	11.91	34.94
Soot	HDMR	47.6%	36.9%	46.5%	0.17	0.30
	DKL (man.)	74.0%	44.0%	71.0%	0.09	0.22
	DKL (opt.)	94.4%	56.1%	90.6%	0.05	0.17

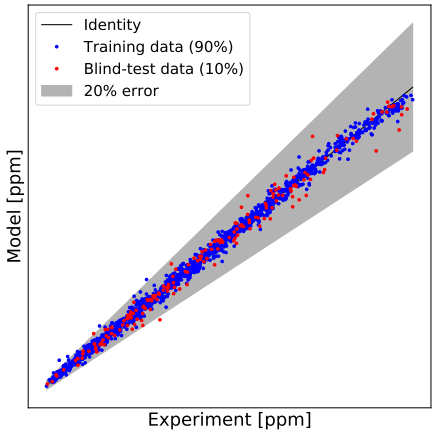
Figure 5 shows a comparison of model versus experiment using HDMR and DKL for both NO_x and soot responses. For DKL, two sets of results are shown: One (Figs. (c) and (d)) obtained using the best values for the hyperparameters found manually, and another one (Figs. (e) and (f)) using the optimised values. The NO_x values are scaled linearly, whereas the soot values are scaled logarithmically. The shaded areas in the plots represent an error margin of 20% with respect to the experimental values. We note that DKL generally produces more accurate regression fits for the data than HDMR for both NO_x and soot,



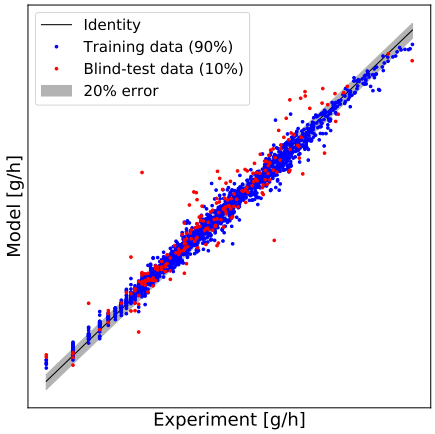
(a) *HDMR regression of NO_x response.*



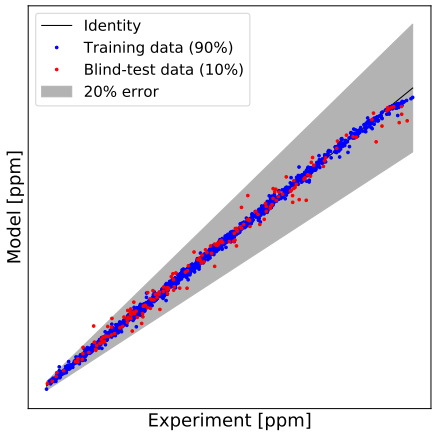
(b) *HDMR regression of soot response.*



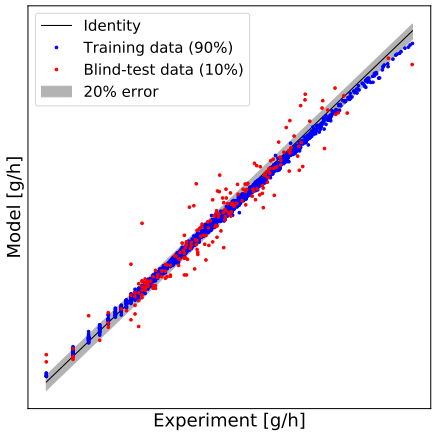
(c) *DKL (manual) for NO_x response.*



(d) *DKL (manual) for soot response.*



(e) *DKL (optimised) for NO_x response.*



(f) *DKL (optimised) for soot response.*

Figure 5: *Modelled NO_x and soot responses against experimental ones using HDMR and two sets of DKL architectures and hyperparameter values. Soot values are logarithmic. For confidentiality reasons, no values are shown on the axes.*

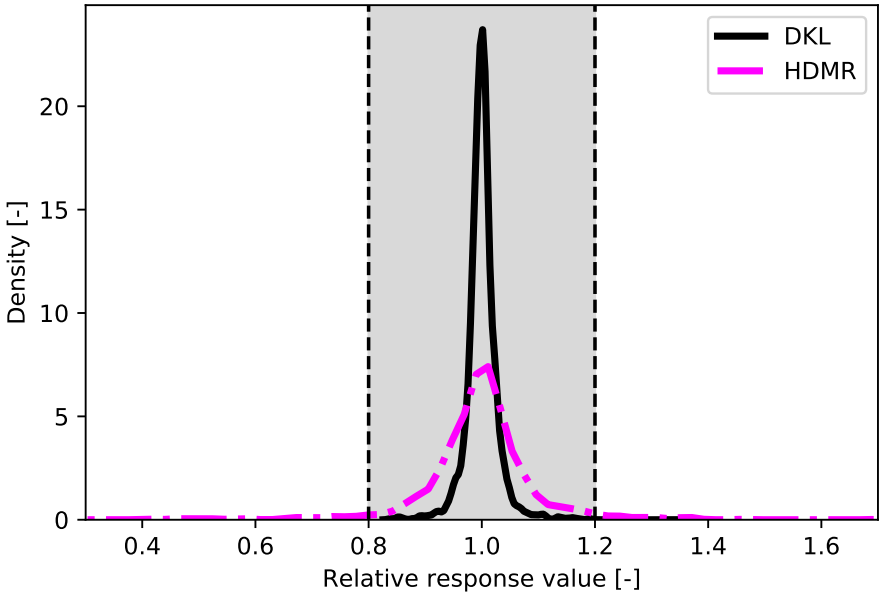
and also that DKL with the optimised hyperparameters is more accurate than with the manual ones. In the NO_x regression with DKL, almost all of the predicted training and blind-test values lie within 20% of the experimental value and a large majority of the predictions align closely with the experimental values. For HDMR, although only few points lie outside the 20% margin, the predictions tend to be distributed further from the experimental values. For the soot regression, similar behaviour is observed, but in this case, significantly more predictions lie outside the 20% error bar with respect to the experimental values. Soot being more challenging than NO_x in terms of regression is entirely expected, due to soot emissions depending much more non-linearly on the engine operating condition and the measurements being intrinsically much more noisy. In addition, we note in Fig. 5 that the spread of training points is quite similar to that of the blind-test points for HDMR, whereas the latter is much wider for DKL, with the effect being stronger for soot than for NO_x. These observations are quantified in Table 4 in terms of the percentage of points which lie within 20% of the experimental values, as well as RSME. The blind-test RMSE is indeed seen to be larger than the training error for both NO_x and soot regression, and the difference is larger in relative terms for soot. This is indicative of over-fitting – a well-known issue with neural networks [45]. Any surrogate with a large number of internal degrees of freedom is prone to over-fitting, especially if this number exceeds the number of data points. Given the number of layers and nodes typically used in a neural network, and hence the associated number of weights, it is clear that in our case the number of degrees of freedom in DKL is much larger than the number of available data points. Another factor contributing to over-fitting is the sparsity of the dataset in the input space, where there are less than 2000 available training points in 14 dimensions. As an aside, an additional consequence of this sparsity is that the variances predicted by Eqn. (10) are too small to be useful, which is why they are not shown in any of the plots.

Figure 6 shows density plots of relative values $\hat{y}/y$ for both outputs for DKL and HDMR, where $\hat{y}$ and y represent the predicted and experimental values, respectively. It can be observed that for both outputs, the relative values for both HDMR and DKL are mainly distributed near unity, and the majority of the predicted values are within 20% of the experimental values for both methods for both outputs. However, it is clear from the plots that the overall distribution of errors of DKL is significantly more centralised at unity than HDMR for the regression of both outputs as the density is more centralised at unity for DKL than HDMR for both outputs. It is also clear that the number of predictions within 20% of the experimental value made by DKL is more than those made by HDMR, and this difference is greater for the soot prediction.

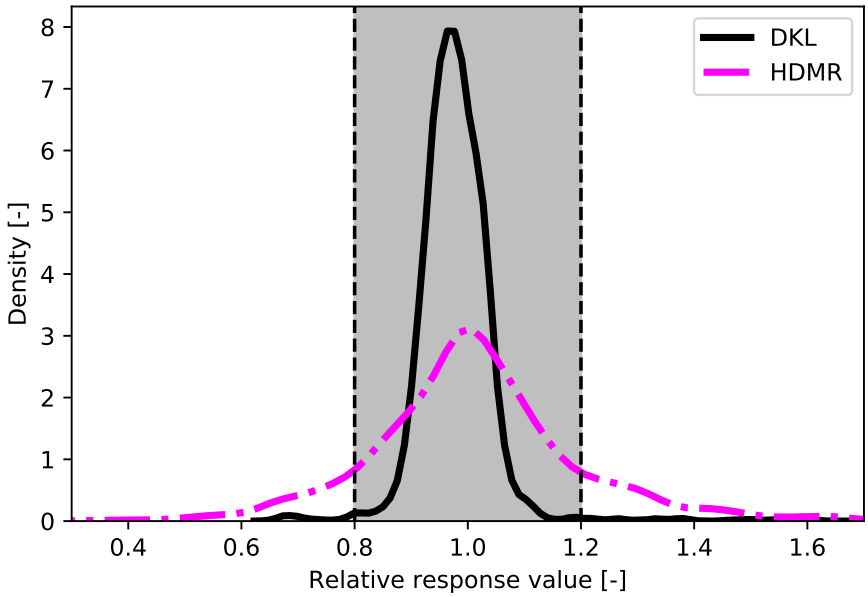
In Table 5, we show a comparison between DKL and HDMR on NO_x regression in terms of their training time and evaluation time. We see that HDMR is significantly computa-

Table 5: *CPU-time comparison between DKL and HDMR.*

Surrogate	Training time [s]	Evaluation time [s]
DKL	56	3
HDMR	2	1



(a) *Distribution of relative values of NO_x response.*



(b) *Distribution of relative values of soot response.*

Figure 6: *Densities of prediction values relative to experiment using HDMR and DKL regression for NO_x and soot emissions, respectively. DKL generates better predictions in terms of the number of points within 20% of the experimental values, and the difference is greater for soot.*

tionally cheaper than the DKL method. The training time for DKL is 56 seconds whereas the HDMR fitting takes less than 2 seconds on the same device. Hence, the trade-off between accuracy and computational cost needs to be taken into consideration when selecting surrogates for particular applications. It is worth noting that evaluation of a trained DKL model is computationally much cheaper than its training process, which indicates that a pre-trained DKL model may be feasible in some near real-time applications. We furthermore note that a large part of this evaluation time is one-off overhead, such that batch-evaluation of collections of points can be achieved with relatively minor additional computational expense.

Table 6: *RMSE performance of the considered surrogates on the Diesel dataset.*

Output	GP (RBF)	GP (SM)	DNN	DKL (RBF)	HDMR
NO_x	110.06	777.93	34.282	**11.91**	44.78
Soot	4.77	2.28	0.52	**0.05**	0.17

Table 6 compares plain Gaussian process regression using RBF and SM kernels, a stand-alone deep neural network, and DKL with RBF kernel, as well as HDMR with respect to root-mean-square error (RMSE) for both NO_x and soot emissions. We observe that the performance of deep kernel learning is significantly superior to the plain Gaussian processes with either the RBF or the spectral mixture kernel on both outputs. This is consistent with expectation, since, as discussed in section 2.4, the 14-dimensional input space of our engine dataset would be expected to cause problems for plain GPs. DKL also outperforms a stand-alone DNN, indicating a genuine benefit in the combination of a GP and a DNN.

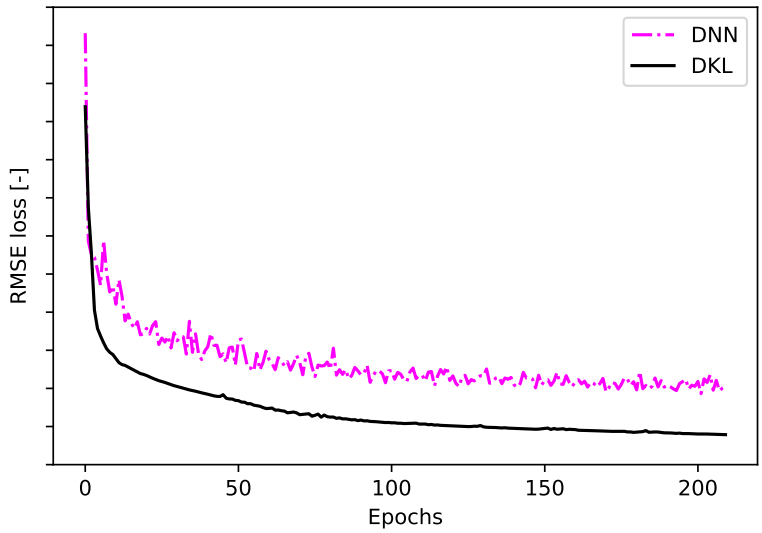


Figure 7: *Loss history of training DKL as well as a plain DNN for NO_x regression.*

In Fig. 7, we show the RMSE loss history of the NO_x regression for DKL and plain neural

network during the training process. We observe that the prediction losses with respect to the experimental values of the DKL training throughout the training process is consistently lower than those of the plain DNN training. Even at the beginning of the training, it can be observed that the loss for the DKL prediction is lower than for the plain DNN prediction. This agrees with our expectation due to the existence of a Gaussian process at the end of DKL, which automatically computes the maximum a posteriori (MAP) estimate given the training dataset as the prediction. We can also see that the loss history curve for the DKL is much smoother than that for the DNN. Hence the RMSE loss of the DKL training process is a robust estimator for the performance of the trained model whereas for the DNN training, the RMSE estimates the performance of the model with larger uncertainty.

From Figs. 5, 6, 7 and Table 6 we conclude that the deep kernel learning shows improvement over HDMR, plain Gaussian processes and plain neural networks in the regression tasks on the considered diesel engine emission dataset.

4 Conclusions

In this paper, we studied Deep Kernel Learning as a surrogate model for diesel engine emission data. Instead of using a physical-based model for modelling the complex system, we have taken a purely data-driven approach. DKL was applied to a commercial Diesel engine dataset for NO_x and soot emissions comprising 1861 data points, with 14 operating condition variables as inputs. We employed a systematic two-stage procedure, consisting of a quasi-random global search and a local gradient-free optimisation stage, to determine seven DKL hyperparameters, which include network architecture as well as kernel and learning parameters. It was found that the global search, conducted through sampling 1000 Sobol points, was most effective in identifying a suitable set of hyperparameters. Local optimisation was found largely ineffective for the network architecture hyperparameters, but led to minor improvement for the kernel and learning parameters. We compared DKL to standard deep feedforward neural networks, Gaussian processes, as well as HDMR, and the results indicate that, overall, DKL outperforms these methods in terms of regression accuracy as measured by RMSE on the considered 14-dimensional engine dataset for NO_x and soot modelling. Further research themes could involve designing DKL with non-stationary kernel functions to deal with the heteroskedasticity of the input arguments.

Acknowledgements

This work was partly funded by the National Research Foundation (NRF), Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme, and by the European Union Horizon 2020 Research and Innovation Programme under grant agreement 646121. Changmin Yu is funded by a Shell PhD studentship, and is thankful to CMCL for the internship programme offered.

Markus Kraft gratefully acknowledges the support of the Alexander von Humboldt foundation.

References

[1] P. Azadi, G. P. Brownbridge, S. Mosbach, O. R. Inderwildi, and M. Kraft. Production of biorenewable hydrogen and syngas via algae gasification: a sensitivity analysis. *Energy Procedia*, 61:2767–2770, 2014. doi:10.1016/j.egypro.2014.12.302.

[2] M. Baran and L. K. Bieniasz. Experiments with an adaptive multicut-HDMR map generation for slowly varying continuous multivariate functions. *Applied Mathematics and Computation*, 258:206–219, 2015. doi:10.1016/j.amc.2015.02.007.

[3] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993. doi:10.1109/18.256500.

[4] B. Berger and F. Rauscher. Robust Gaussian process modelling for engine calibration. *IFAC Proceedings Volumes*, 45(2):159–164, 2012. doi:10.3182/20120215-3-AT-3016.00028.

[5] B. Berger, F. Rauscher, and B. Lohmann. Analysing Gaussian processes for stationary black-box combustion engine modelling. *IFAC Proceedings Volumes*, 44(1): 10633–10640, 2011. doi:10.3182/20110828-6-IT-1002.01160.

[6] CMCL Innovations. MoDS (Model Development Suite), version 0.9.0, 2018. https://cmclinnovations.com/products/mods/.

[7] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 155–161. MIT Press, 1997. URL `http://papers.nips.cc/paper/1238-support-vector-regression-machines.pdf`.

[8] M. Frenklach, H. Wang, and M. J. Rabinowitz. Optimization and analysis of large chemical kinetic mechanisms using the solution mapping method — combustion of methane. *Progress in Energy and Combustion Science*, 18:47–73, 1992. doi:10.1016/0360-1285(92)90032-V.

[9] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*, pages 7587–7597, 2018.

[10] S. S. Garud, I. A. Karimi, and M. Kraft. Design of computer experiments: A review. *Computers & Chemical Engineering*, 106:71–95, 2017. doi:10.1016/j.compchemeng.2017.05.010.

[11] S. S. Garud, I. A. Karimi, and M. Kraft. Smart sampling algorithm for surrogate model development. *Computers & Chemical Engineering*, 96:103–114, 2017. doi:10.1016/j.compchemeng.2016.10.006.

[12] S. S. Garud, I. A. Karimi, G. P. Brownbridge, and M. Kraft. Evaluating smart sampling for constructing multidimensional surrogate models. *Computers & Chemical Engineering*, 108:276–288, 2018. doi:10.1016/j.compchemeng.2017.09.016.

[13] S. S. Garud, I. A. Karimi, and M. Kraft. LEAPS2: Learning based evolutionary assistive paradigm for surrogate selection. *Computers & Chemical Engineering*, 119:352–370, 2018. doi:10.1016/j.compchemeng.2018.09.008.

[14] M. Ghanbari, G. Najafi, B. Ghobadian, R. Mamat, M. Noor, and A. Moosavian. Support vector machine to predict diesel engine performance and emission parameters fueled with nano-particles additive to diesel fuel. In *IOP Conference Series: Materials Science and Engineering*, volume 100, page 012069. IOP Publishing, 2015. doi:10.1088/1757-899X/100/1/012069.

[15] B. Ghobadian, H. Rahimi, A. Nikbakht, G. Najafi, and T. Yusaf. Diesel engine performance and exhaust emission analysis using waste cooking biodiesel fuel with an artificial neural network. *Renewable Energy*, 34(4):976–982, 2009. doi:10.1016/j.renene.2008.08.008.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015. URL https://arxiv.org/pdf/1502.01852.pdf.

[17] R. Hooke and T. A. Jeeves. "Direct search" solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229, 1961. doi:10.1145/321062.321069.

[18] G.-B. Huang and L. Chen. Convex incremental extreme learning machine. *Neurocomputing*, 70(16-18):3056–3062, 2007. doi:10.1016/j.neucom.2007.02.009.

[19] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006. doi:10.1016/j.neucom.2005.12.126.

[20] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529, 2012. doi:10.1109/TSMCB.2011.2168604.

[21] V. M. Janakiraman, X. Nguyen, and D. Assanis. Stochastic gradient based extreme learning machines for stable online learning of advanced combustion engines. *Neurocomputing*, 177:304–316, 2016. doi:10.1016/j.neucom.2015.11.024.

[22] S. Jeong, S. Obayashi, and Y. Minemura. Application of hybrid evolutionary algorithms to low exhaust emission diesel engine design. *Engineering Optimization*, 40 (1):1–16, 2008. doi:10.1080/03052150701561155.

[23] S. Joe and F. Y. Kuo. Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654, 2008. doi:10.1137/070709359.

[24] C. A. Kastner, A. Braumann, P. L. Man, S. Mosbach, G. P. Brownbridge, J. Akroyd, M. Kraft, and C. Himawan. Bayesian parameter estimation for a jet-milling model using Metropolis-Hastings and Wang-Landau sampling. *Chemical Engineering Science*, 89:244–257, 2013. doi:10.1016/j.ces.2012.11.027.

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. URL `https://arxiv.org/pdf/1412.6980.pdf`.

[26] J. Lai, O. Parry, S. Mosbach, A. Bhave, and V. Page. Evaluating emissions in a modern compression ignition engine using multi-dimensional PDF-based stochastic simulations and statistical surrogate generation. SAE Technical Paper 2018-01-1739, 2018. doi:10.4271/2018-01-1739.

[27] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015. doi:10.1038/nature14539.

[28] G. Li, S.-W. Wang, and H. Rabitz. Practical approaches to construct RS-HDMR component functions. *The Journal of Physical Chemistry A*, 106(37):8721–8733, 2002. doi:10.1021/jp014567t.

[29] E. Lughofer, V. Macián, C. Guardiola, and E. P. Klement. Identifying static and dynamic prediction models for NOx emissions with evolving fuzzy systems. *Applied Soft Computing*, 11(2):2487–2500, 2011. doi:10.1016/j.asoc.2010.10.004.

[30] A. A. Malikopoulos, P. Y. Papalambros, and D. N. Assanis. A learning algorithm for optimal internal combustion engine calibration in real time. In *ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 91–100. American Society of Mechanical Engineers, 2007. doi:10.1115/DETC2007-34718.

[31] H. Q. Minh, P. Niyogi, and Y. Yao. Mercer's theorem, feature maps, and smoothing. In *International Conference on Computational Learning Theory*, pages 154–168. Springer, 2006.

[32] S. Mosbach, A. Braumann, P. L. Man, C. A. Kastner, G. P. Brownbridge, and M. Kraft. Iterative improvement of Bayesian parameter estimates for an engine model by means of experimental design. *Combustion and Flame*, 159(3):1303–1313, 2012. doi:10.1016/j.combustflame.2011.10.019.

[33] S. Mosbach, J. H. Hong, G. P. Brownbridge, M. Kraft, S. Gudiyella, and K. Brezinsky. Bayesian error propagation for a kinetic model of n-propylbenzene oxidation in a shock tube. *International Journal of Chemical Kinetics*, 46(7):389–404, 2014. doi:10.1002/kin.20855.

[34] G. Najafi, B. Ghobadian, T. Tavakoli, D. Buttsworth, T. Yusaf, and M. Faizollahnejad. Performance and exhaust emissions of a gasoline engine with ethanol blended gasoline fuels using artificial neural network. *Applied Energy*, 86(5):630–639, 2009. doi:10.1016/j.apenergy.2008.09.017.

[35] G. Najafi, B. Ghobadian, A. Moosavian, T. Yusaf, R. Mamat, M. Kettner, and W. H. Azmi. SVM and ANFIS for prediction of performance and exhaust emissions of a SI engine with gasoline-ethanol blended fuels. *Applied Thermal Engineering*, 95: 186–203, 2016. doi:10.1016/j.applthermaleng.2015.11.009.

[36] R. M. Neal. *Bayesian learning for neural networks*, volume 118 of *Lecture Notes in Statistics*. Springer Science+Business Media, New York, 1996. doi:10.1007/978-1-4612-0745-0.

[37] X. Niu, C. Yang, H. Wang, and Y. Wang. Investigation of ANN and SVM based on limited samples for performance and emissions prediction of a CRDI-assisted marine diesel engine. *Applied Thermal Engineering*, 111:1353–1364, 2017. doi:10.1016/j.applthermaleng.2016.10.042.

[38] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991. doi:10.1162/neco.1991.3.2.246.

[39] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 2017. URL https://openreview.net/pdf?id=BJJsrmfCZ.

[40] J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005. URL http://dl.acm.org/citation.cfm?id=1046920.1194909.

[41] H. Rabitz and Ö. F. Aliş. General foundations of high-dimensional model representations. *Journal of Mathematical Chemistry*, 25(2-3):197–233, 1999. doi:10.1023/A:1019188517934.

[42] C. Sayin, H. M. Ertunc, M. Hosoz, I. Kilicaslan, and M. Canakci. Performance and exhaust emissions of a gasoline engine using artificial neural network. *Applied Thermal Engineering*, 27(1):46–54, 2007. doi:10.1016/j.applthermaleng.2006.05.016.

[43] J. J. Sikorski, G. Brownbridge, S. S. Garud, S. Mosbach, I. A. Karimi, and M. Kraft. Parameterisation of a biodiesel plant process flow sheet model. *Computers & Chemical Engineering*, 95:108–122, 2016. doi:10.1016/j.compchemeng.2016.06.019.

[44] A. S. Silitonga, H. H. Masjuki, H. C. Ong, A. H. Sebayang, S. Dharma, F. Kusumo, J. Siswantoro, J. Milano, K. Daud, T. M. I. Mahlia, W.-H. Cheng, and B. Sugiyanto. Evaluation of the engine performance and exhaust emissions of biodiesel-bioethanol-diesel blends using kernel-based extreme learning machine. *Energy*, 159:1075–1087, 2018. doi:10.1016/j.energy.2018.06.202.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. URL http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf.

[46] N. Tietze. *Model-based Calibration of Engine Control Units Using Gaussian Process Regression*. PhD thesis, Technische Universität Darmstadt, 2015. URL https://core.ac.uk/download/pdf/76649740.pdf.

[47] A. Vaughan and S. V. Bohac. Real-time, adaptive machine learning for non-stationary, near chaotic gasoline engine combustion time series. *Neural Networks*, 70:18–26, 2015. doi:10.1016/j.neunet.2015.04.007.

[48] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006. URL http://www.gaussianprocess.org/gpml/chapters/RW.pdf.

[49] A. Wilson and R. Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International Conference on Machine Learning*, pages 1067–1075, 2013. URL http://proceedings.mlr.press/v28/wilson13.pdf.

[50] A. Wilson and H. Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *International Conference on Machine Learning*, pages 1775–1784, 2015. URL https://arxiv.org/pdf/1503.01057.pdf.

[51] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016. URL https://arxiv.org/pdf/1511.02222.pdf.

[52] K. I. Wong, P. K. Wong, and C. S. Cheung. Modelling and prediction of diesel engine performance using relevance vector machine. *International Journal of Green Energy*, 12(3):265–271, 2015. doi:10.1080/15435075.2014.891513.

[53] P. K. Wong, X. H. Gao, K. I. Wong, and C. M. Vong. Online extreme learning machine based modeling and optimization for point-by-point engine calibration. *Neurocomputing*, 277:187–197, 2018. doi:10.1016/j.neucom.2017.02.104.

[54] N. Yilmaz, E. Ileri, A. Atmanlı, A. D. Karaoglan, U. Okkan, and M. S. Kocak. Predicting the engine performance and exhaust emissions of a diesel engine fueled with hazelnut oil methyl ester: the performance comparison of response surface methodology and LSSVM. *Journal of Energy Resources Technology*, 138(5):052206, 2016. doi:10.1115/1.4032941.

[55] T. F. Yusaf, D. R. Buttsworth, K. H. Saleh, and B. F. Yousif. CNG-diesel engine performance and exhaust emission analysis with the aid of artificial neural network. *Applied Energy*, 87(5):1661–1669, 2010. doi:10.1016/j.apenergy.2009.10.009.