Combining Gaussian Processes and Neural Networks in Surrogate Modeling for Covariance Matrix Adaptation Evolution Strategy

Jan Koza[1], Jiří Tumpach[2], Zbyněk Pitra[3], and Martin Holeňa[4]

[1] Czech Technical University, Faculty of Information Technology, Prague, kozajan@fit.cvut.cz
[2] Charles University, Faculty of Mathematics and Physics, Prague, tumpach@cs.cas.cz
[3] Czech Technical University, Faculty of Nuclear Sciences and Physical Engineering, Prague, z.pitra@gmail.com
[4] Academy of Sciences, Institute of Computer Science, Prague, martin@cs.cas.cz

Abstract: This paper focuses on surrogate models for Covariance Matrix Adaptation Evolution Strategy (CMA-ES) in continuous black-box optimization. Surrogate modeling has proven to be able to decrease the number of evaluations of the objective function, which is an important requirement in some real-world applications where the evaluation can be costly or time-demanding. Surrogate models achieve this by providing an approximation instead of the evaluation of the true objective function. One of the state-of-the-art models for this task is the Gaussian process. We present an approach to combining Gaussian processes with artificial neural networks, which was previously successfully applied to other machine learning domains.

The experimental part employs data recorded from previous CMA-ES runs, allowing us to assess different settings of surrogate models without running the whole CMA-ES algorithm. The data were collected using 24 noiseless benchmark functions of the platform for comparing continuous optimizers COCO in 5 different dimensions. Overall, we used data samples from over 2.8 million generations of CMA-ES runs. The results examine and statistically compare six covariance functions of Gaussian processes with the neural network extension. So far, the combined model did not show up to outperform the Gaussian process alone. Therefore, in conclusion, we discuss possible reasons for this and ideas for future research.

Keywords: black-box optimization, surrogate modeling, artificial neural networks, Gaussian processes, covariance functions

1 Introduction

In evolutionary black-box optimization, an increasing attention is paid to tasks with *expensive evaluation* of the black-box objective function. It is immaterial whether that expensiveness is due to time-consuming computation like in long simulations [17], or due to evaluation through costly experiments like in some areas of science [3]. To deal with such expensive evaluations, black-box optimization has in the late 1990s and early 2000s adopted an approach called *surrogate modeling* or *metamodeling* [7, 13, 15, 34, 42, 45, 48].

Basically, a surrogate model in continuous black-box optimization is any regression model that with a sufficient fidelity approximates the true black-box objective function and replaces it in some of its evaluations. Among the regression models most frequently used to this end, we are most interested in *Gaussian processes (GPs)* [47], due to the fact that they estimate not only the expected value of the true objective function, but the whole distribution of its values. Apart from regression, they are also encountered in classification, and play a key role in Bayesian optimization [22, 28], where the distribution of values provides selection criteria that are alternatives to the objective function value, such as the probability of improvement or the expected improvement.

The importance of GPs in machine learning incited attempts to integrate them with the leading learning paradigm of the last decades – neural learning, including deep learning. The attractiveness of this research direction is further supported by recent theoretical results concerning relationships of asymptotic properties of important kinds of artificial neural networks (ANNs) to properties of GPs [35,39,41]. The integration of GP with neural learning has been proposed on two different levels:

(i) *Proper integration* of an ANN with a GP, in which the GP forms the final output layer of the ANN [9, 52].

(ii) Only a *transfer of the layered structure*, which is a crucial feature, to the GP context, leading to the concept of deep GPs (DGPs) [8, 12, 22, 23].

The recalled research into the integration of GPs with neural learning has used regression data [8, 9, 12, 23, 52], mostly from the UCI Machine Learning Repository [50], but also data concerning locomotion of walking bipedal robots [9], and face patches and hadwritten digits [52]. In [12,23], also classification data were used. However, we are aware of only one application of such an integration, in particular of DGPs, to two very easy 1- and 2-dimensional optimization problems [22]. Hence, there is a gap between the importance of GPs in Bayesian optimization and missing investigations of the suitability of integrated GP-ANN models for surrogate modeling in black-box optimization. That gap motivated the research reported in this paper.

We have performed this novel kind of investigation of GP-ANN integration using data from more than 5000 runs of using GPs as Bayesian optimizers in black-box optimization, i.e. optimization of pointwise observable func-

tions for which no analytic expression is available and the function values have to be obtained either empirically, e.g. through measuring or experiments, or through numerical simulations. Based on that data, our investigation addressed two *research questions*:

1. Does the integration of a GP with neural learning has an added value compared with employing a GP alone or an ANN alone?
2. What is the impact of each of the considered GP covariance functions on the result of GP-ANN integration?

The next section introduces the principles of surrogate modeling as well as of the evolutionary optimization method *Covariance Matrix Adaptation Evolution Strategy (CMA-ES)*, in the context of which our research has been performed. In Section 3, the fundamentals of GPs are recalled and the method we have used for their integration with neural networks is explained. The core part of the paper is Section 4, which attempts to provide experimental answers to the above research questions.

2 Surrogate modeling in Black-Box Optimization

Basically, the purpose of surrogate modeling – to approximate an unknown functional dependence – coincides with the purpose of *response surface modeling* in the design of experiments [24, 40]. Therefore, it is not surprising that typical response surface models, i.e., *low order polynomials*, belong also to the most traditional and most successful surrogate models [1, 2, 20, 29, 45]. Other frequently used kinds of surrogate models are *artificial neural networks* of the kind multilayer perceptron (MLP) or radial basis function network [18, 25–27, 42, 53], and Gaussian processes, in surrogate modeling also known as *kriging* [6, 7, 13–15, 33, 34, 37, 49, 51]. Occasinally encountered are *support vector regression* [10, 36] and *random forests* [5, 44].

2.1 CMA-ES and Its Surrogate-Assisted Variant DTS-CMA-ES

The CMA-ES algorithm performs unconstrained optimization on $\mathbb{R}^d$, by means of iterative sampling of populations sized λ from a d-dimensional Gaussian distribution $N(m,\sigma^2 C)$, and uses a given parent number μ among the sampled points corresponding to the lowest objective function values, to update the parameters of that distribution. Hence, it updates the expected value m, which is used as the current point estimate of the function optimum, the matrix C and the step-size σ. The CMA-ES is invariant with respect to strictly increasing transformations of the objective function. Hence, to make use of the evaluations of the objective function in a set of points, it needs to know only the ordering of those evaluations. Details of the algorithm can be found in [19, 21].

During the more than 20 years of CMA-ES existence, a number of surrogate-assisted variants of this algorithm have been proposed, a survey can be found in [6, 43]. Here, we pay attention only to the most recent GP-based among them, the Doubly Trained Surrogate CMA-ES (DTS-CMA-ES) [6], surrogate-assisted variant of CMA-ES. It employs two GPs f_1 and f_2, trained consecutively, to find an evaluation of the population $x_1,\ldots,x_\lambda$, with f_1 used for active learning of training data for f_2. Due to the CMA-ES invariance with respect to strictly increasing transformations, it evaluates the difference between predictions only according to the difference in the ordering of those predictions, using the ranking difference error (RDE). The RDE of $y \in \mathbb{R}^\lambda$ with respect to $y' \in \mathbb{R}^\lambda$ considering k best components is defined:

$$\underset{\leq k}{\mathrm{RDE}}(y,y') = \frac{\sum_{i,(\rho(y'))_i \leq k} (|\rho(y'))_i - (\rho(y))_i|}{\max_{\pi\in\Pi(\lambda)} \sum_{i=1}^{k} |i-\pi^{-1}(i)|}, \quad (1)$$

where $\Pi(\lambda)$ denotes the set of permutaions of $\{1,\ldots,\lambda\}$ and $\rho(y)$ denotes the ordering of the components of y, i.e., $(\forall y \in \mathbb{R}^\lambda)\ \rho(y) \in \Pi(\lambda)\ \&\ (\rho(y))_i < (\rho(y))_j \Rightarrow y_i \leq y_j$. Because the CMA-ES algorithm uses the surrogate model to select the most promising candidates for true evaluation, the metric considers only k best samples. The range of RDE metric is $[0,1]$, it equals 0 for the exact ordering and 1 for thereverse order.

The algorithm DTS-CMA-ES is described in Algorithm 1, using the following notation:

- A for an *archive* – a set of points that have already been evaluated by the true black-box objective function *BB*;
- $d_{\sigma^2 C}$ for the Mahalanobis distance given by $\sigma^2 C$:
$$d_{\sigma^2 C}(x,x') = \sqrt{(x-x')^\top \sigma^{-2} C^{-1} (x-x')}; \quad (2)$$
- $N_k(x;A)$ for the set of k of $d_{\sigma^2 C}$-nearest neighbours of $x \in \mathbb{R}^d$ with respect to the archive A;
- $f_i(x_1,\ldots,x_\lambda) = (f_i(x_1),\ldots,f_i(x_\lambda))$, for $i = 1,2$;
- T_h for the *training set selection*:
$$T_h = \bigcup_{j=1}^{\lambda} \{x \in N_h(x_j;A) | d_{\sigma^2 C}(x,x_j) < r_{\max}\} \quad (3)$$
with $r_{\max} > 0$ for $h = 1,\ldots,|A|$;
- $k(A) = \max\{h | |T_h| \leq N_{\max}\}$, with $N_{\max} \in \mathbb{N}$;
- ρ_{PoI} for decreasing ordering of $f_1(x_1),\ldots,f_1(x_\lambda)$ according to the probability of improvement with respect to the lowest *BB* value found so far,
$$i < j \Rightarrow \mathrm{PoI}(\rho_{\mathrm{PoI}}(f_1(x_1,\ldots,x_\lambda)))_i;V) \geq \geq \mathrm{PoI}(\rho_{\mathrm{PoI}}(f_1(x_1,\ldots,x_\lambda)))_j;V), \quad (4)$$
where $V = \min_{x\in A} BB(x)$.

Algorithm 1 Algorithm DTS-CMA-ES

Require: $x_1,\ldots,x_\lambda \in \mathbb{R}^d$, μ, A, σ and C – step size and matrix from the CMA-ES distribution, $N_{\max} \in \mathbb{N}$ such that $N_{\max} \geq \lambda$, $r_{\max} > 0$, $\beta, \varepsilon_{\min}, \varepsilon_{\max}, \varepsilon_{\text{initial}}, \alpha_{\min}, \alpha_{\max}, \alpha_{\text{initial}} \in (0,1)$

if this is the 1st call of the algorithm in the current CMA-ES run **then**
  $\alpha = \alpha_{\text{initial}}, \varepsilon = \varepsilon_{\text{initial}}$
else
  take over the returned values of α, ε from its previous call in the run
end if
Train a Gaussian process f_1 on $T_{k(A)}$, estimating m_{GP}, σ_n, σ_f, ℓ through maximization of the likelihood (7)
Evaluate $BB(x_j)$ for x_j not yet BB-evaluated and such that $(\rho_{\text{PoI}}(f_1(x_1,\ldots,x_\lambda)))_j \leq \lceil \alpha\lambda \rceil$
Update A to $A \cup \{(x_j | (\rho_{\text{PoI}}(f_1(x_1),\ldots,f_1(x_\lambda)))_j \leq \lceil \alpha\lambda \rceil\}$
Train a Gaussian process f_2 on $T_{k(A)}$, estimating m_{GP}, σ_n, σ_f, ℓ through maximization of the likelihood (7)
For x_j such that $(\rho_{\text{PoI}}(f_1(x_1,\ldots,x_\lambda)))_j \leq \lceil \alpha\lambda \rceil$, update $f_2(x_j) = BB(x_j)$
Update ε to $\beta\,\text{RDE}_\mu(f_1(x_1,\ldots,x_\lambda),(f_2(x_1,\ldots,x_\lambda)) + (1-\beta)\varepsilon$ and α to $\alpha_{\min} + \max(0, \min(1, \frac{\varepsilon - \varepsilon_{\min}}{\varepsilon_{\max} - \varepsilon_{\min}}))$
For j fulfilling $(\rho_{\text{PoI}}(f_1(x_1,\ldots,x_\lambda))_j > \lceil \alpha\lambda \rceil$, update $f_2(x_j)$ to $f_2(x_j) - \min\{f_2(x_{j'}) | (\rho_{\text{PoI}}(f_1(x_1,\ldots,x_\lambda))_{j'} > \lceil \alpha\lambda \rceil\} + \min\{f_2(x_{j'}) | (\rho_{\text{PoI}}(f_1(x_1,\ldots,x_\lambda))_{j'} \leq \lceil \alpha\lambda \rceil\}$
Return $f_2(x_1),\ldots,f_2(x_\lambda), \varepsilon, \alpha$

3 Gaussian Processes and Their Integration with Neural Networks

3.1 Gaussian processes

A *Gaussian process* on a set $X \subset \mathbb{R}^d, d \in \mathbb{N}$ is a collection of random variables $(f(x))_{x\in X}$, any finite number of which has a joint Gaussian distribution [47]. It is completely specified by a *mean function* $\mu : X \to \mathbb{R}$, typically assumed constant, and by a *covariance function* $\kappa : X \times X \to \mathbb{R}$ such that for $x, x' \in X$,

$$\mathbb{E}f(x) = \mu, \text{cov}(f(x), f(x')) = \kappa(x,x'). \tag{5}$$

Therefore, a GP is often denoted $GP(\mu, \kappa(x,x'))$ or $GP(\mu,\kappa)$.

The value of $f(x)$ is typically accessible only as a *noisy observation* $y = f(x) + \varepsilon$, where ε is a zero-mean Gaussian noise with a variance $\sigma_n > 0$. Then

$$\text{cov}(y,y') = \kappa(x,x') + \sigma_n^2 \mathbb{I}(x = x'), \tag{6}$$

where $\mathbb{I}(\text{proposition})$ equals for a true proposition 1, for a false proposition 0.

Consider now the prediction of the random variable $f(x_\star)$ in a point $x_\star \in X$ if we already know the observations $y_1,\ldots,y_n$ in points $x_1,\ldots,x_n$. Introduce the vectors $x = (x_1,\ldots,x_n)^\top$, $y = (y_1,\ldots,y_n)^\top = (f(x_1) + \varepsilon, \ldots f(x_n) + \varepsilon)^\top$, $k_\star = (\kappa(x_1,x_\star),\ldots,\kappa(x_n,x_\star))^\top$ and the matrix $K \in \mathbb{R}^{n\times n}$ such that $(K)_{i,j} = \kappa(x_i,x_j) + \sigma_n^2 \mathbb{I}(i=j)$. Then the probability density of the vector y of observations is

$$p(y;\mu,\kappa,\sigma_n^2) = \frac{\exp\left(-\frac{1}{2}(y-\mu)^\top K^{-1}(y-\mu)\right)}{\sqrt{(2\pi)^2 \det(K + \sigma_n^2 I_n)}}, \tag{7}$$

where $\det(M)$ denotes the determinant of a matrix M. Furthermore, as a consequence of the assumption of Gaussian joint distribution, also the conditional distribution of $f(x_\star)$ conditioned on y is Gaussian, namely

$$N(\mu(x_\star) + k_\star K^{-1}(y-\mu), \kappa(x_\star,x_\star) - k_\star^\top K^{-1} k_\star). \tag{8}$$

According to (6), the relationship between the observations y and y' is determined by the covariance function κ. In the reported research, we have considered 6 kinds of covariance functions, listed below. In their definitions, the notation $r = \|x' - x\|$ is used, and among the parameters of κ, aka hyperparameters of the GP, frequently encountered are $\sigma_f^2, \ell > 0$, called *signal variance* and *characteristic length scale*, respectively. Other parameters are introduced for each covariance function separately.

(i) *Linear*: $\kappa_{\text{LIN}}(x,x') = \sigma_0^2 + x^\top x'$, with a bias σ_0^2.

(ii) *Quadratic* is the square of the linear covariance: $\kappa_{\text{Q}}(x,x') = (\sigma_0^2 + x^\top x')^2$.

(iii) *Rational quadratic*: $\kappa_{\text{RQ}}(x,x') = \sigma_f^2 \left(1 + \frac{r^2}{2\alpha\ell^2}\right)^{-\alpha}$, with $\alpha > 0$.

(iv) *Squared exponential*: $\kappa_{\text{SE}}(x,x') = \sigma_f^2 \exp\left(-\frac{r^2}{2\ell^2}\right)$.

(v) *Matérn* $\frac{5}{2}$:
$\kappa_{\text{Matérn}}^{\frac{5}{2}}(x,x') = \sigma_f^2 \left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}r}{\ell}\right)$.

(vi) One *composite covariance function*, namely the sum of κ_{SE} and κ_{Q}:
$\kappa_{\text{SE+Q}}(x,x') = \kappa_{\text{SE}}(x,x') + \kappa_{\text{Q}}(x,x')$.

3.2 GP as the Output Layer of a Neural Network

The approach integrating a GP into an ANN as its output layer has been independently proposed in [9] and [52]. It relies on the following two assumptions:

1. If n_I denotes the number of the ANN input neurons, then the ANN computes a *mapping* net *of n_I-dimensional input values into the set X* on which is the GP. Consequently, the number of neurons in the last hidden layer equals the dimension d, and the ANN maps an input v into a point $x = \text{net}(v) \in X$, corresponding to an observation $f(x+\varepsilon)$ governed by GP (Figure 1). From the point of view of the ANN inputs, the GP is now $GP(\mu(\text{net}(v)), \kappa(\text{net}(v), \text{net}(v')))$.

2. The GP mean μ *is assumed to be a known constant*, thus not contributing to the GP hyperparameters and independent of net.

Due to the assumption 2., the only hyperparameters of the GP are the parameters θ^κ of the covariance function.

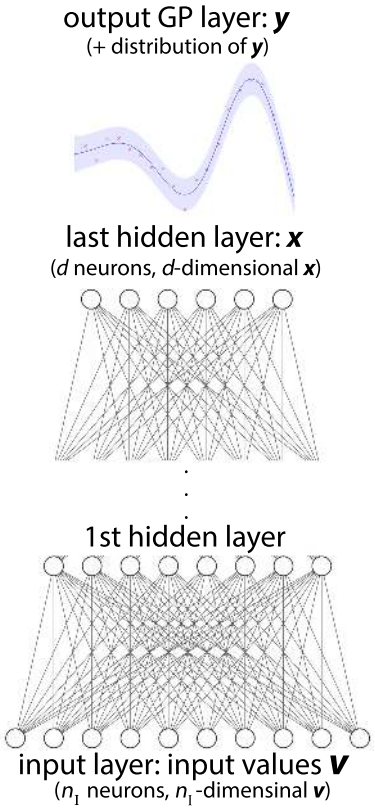

Figure 1: Schema of the integration of a GP into an ANN as its output layer. Taken over from [32].

As to the ANN, it depends on the one hand on the vector θ^W of its weights and biases, on the other hand on the parameters θ^A of its architecture (such as the number of hidden layers, the numbers of neurons in each of them, the activation functions). Altogether, the integrated GP-ANN model depends on the vector $(\theta^\kappa, \theta^W, \theta^A)$, which we in accordance with [9,52] and with the terminology common for GPs also call hyperparameters, although in the context of ANNs, this term is typically used only for θ^A.

Consider now n inputs to the neural network, $v_1, \ldots, v_n$, mapped to the inputs $x_1 = \mathrm{net}(v_1), \ldots, x_n = \mathrm{net}(v_n)$ of the GP, corresponding to the observations $y = (y_1, \ldots, y_n)^\top$. Then the log-likelihood of θ is

$$L(\theta) = \ln p(y; \mu, \kappa, \sigma_n^2), \tag{9}$$

where μ is the constant assumed in Assumption 2., and

$$(K)_{i,j} = \kappa(\mathrm{net}(v_i), \mathrm{net}(v_j)). \tag{10}$$

To find the combination of values of θ with the maximal likelihood, methods for smooth optimization can be applied only with respect to θ^κ and θ^W, with respect to which is L continuous. With respect to θ^A, we have two possibilities. First, to fix θ^A in advance, thus effectively restricting the hyperparameters of the integrated model to $(\theta^\kappa, \theta^W)$. Second, to perform the optimization with respect to θ^A in an outer loop (using, e.g. grid search combined with cross-validation) and for each considered combination of values of θ^A perform the optimization with respect to $(\theta^\kappa, \theta^W)$ in an inner loop.

Finally, let the smooth optimization be performed in the most simple but, in the context of neural networks, also most frequent way – as gradient descent. The partial derivatives forming $\nabla_{(\theta^\kappa, \theta^W)} L$ can be computed as:

$$\begin{aligned} \frac{\partial L}{\partial \theta_\ell^\kappa} &= \sum_{i,j=1}^{n} \frac{\partial L}{\partial K_{i,j}} \frac{\partial K_{i,j}}{\partial \theta_\ell^\kappa}, \frac{\partial L}{\partial \theta_\ell^W} = \\ &= \sum_{i,j,k=1}^{n} \frac{\partial L}{\partial K_{i,j}} \frac{\partial K_{i,j}}{\partial x_k} \frac{\partial \, \mathrm{net}(v_k)}{\partial \theta_\ell^W}. \end{aligned} \tag{11}$$

In (11), the partial derivatives $\frac{\partial L}{\partial K_{i,j}}$, $i, j = 1, \ldots, n$, are components of the matrix derivative $\frac{\partial L}{\partial K}$, for which the calculations of matrix differential calculus [38] together with (7) and (9) yield

$$\frac{\partial L}{\partial K} = \frac{1}{2} \left(K^{-1} y y^\top K^{-1} - K^{-1} \right). \tag{12}$$

4 Experiments

For the evaluation of the aforementioned models, we used the open-source Python library GPytorch and our implementation is available at [31].

4.1 Employed Data

As a dataset to compare different configurations of the combined GP-ANN model, we used recorded DTS-CMA-ES runs from previous experiments. This allowed us to effectively evaluate the surrogate model on its own without having to perform the whole optimization. The open-source Matlab implementation of DTS-CMA-ES, used to obtain the data, is available at [4]. The underlying surrogate models were Gaussian process with 8 different covariance functions implemented using the GPML Toolbox [46]. The optimization runs were collected on the platform for comparing continuous optimizers COCO. The employed black-box optimization benchmark set contains 24 noiseless functions scalable to any input dimension. We used dimensions 2, 3, 5, 10, and 20 in 5 different instances, which consist in random rotation and translation in the input space. Therefore, for each combination of benchmark function and dimension, 40 runs are available. More important than the number of runs, however, is the number of their generations because data from each generation apart from the first can be used for testing all those surrogate models that could be trained with data from the previous generations. The number of generations in a particular run of CMA-ES algorithm is unknown before the run and depends on the objective function and its particular instance.

Table 1: Noiseless benchmark functions of the platform comparing continuous optimizers (COCO) [11, 16] and the number of available generations of DTS-CMA-ES runs for each of them in each considered dimension

	Benchmark function name	Available generations in dimension				
		2	3	5	10	20
	Separable					
1.	Sphere	4689	6910	11463	17385	25296
2.	Separable Ellipsoid	6609	9613	15171	25994	55714
3.	Separable Rastrigin	7688	11308	17382	27482	42660
4.	Büche-Rastrigin	8855	13447	22203	31483	49673
	Moderately Ill-Conditioned					
6.	Attractive Sector	16577	25200	38150	45119	72795
7.	Step Ellipsoid	7103	9816	24112	34090	56340
8.	Rosenbrock	7306	11916	21191	32730	71754
9.	Rotated Rosenbrock	7687	12716	24084	35299	71017
	Highly Ill-Conditioned					
10.	Ellipsoid with High Conditioning	6691	9548	15867	25327	59469
11.	Discus	6999	9657	15877	25478	45181
12.	Bent Cigar	10369	18059	28651	34605	56528
13.	Sharp Ridge	7760	11129	20346	30581	48154
14.	Different Powers	6653	10273	17693	31590	61960
	Multi-modal with Global Structure					
15.	Non-separable Rastrigin	7855	11476	19374	28986	44446
16.	Weierstrass	9294	13617	24158	27628	40969
17.	Schaffers F7	9031	13960	24244	34514	56247
18.	Ill-Conditioned Schaffers F7	9598	13404	25802	31609	53836
19.	Composite Griewank-Rosenbrock	9147	16268	24456	34171	53536
	Multi-modal Weakly Structured					
20.	Schwefel	9081	13676	24219	33753	53104
21.	Gallagher's Gaussian 101-me Points	7645	12199	18208	25366	43186
22.	Gallagher's Gaussian 21-hi Points	7629	11086	17881	26626	44971
23.	Katsuura	8751	11233	17435	25030	37366
24.	Lunacek bi-Rastrigin	8983	13966	19405	29762	44420

The benchmarks and their total numbers of available generations for individual dimensions are listed in Table 1. We skipped the linear function (benchmark function number 5), which is easy to optimize, and the recorded runs did not provide enough data samples to train and evaluate the surrogate models.

4.2 Experimental Setup

We compare the combinations of neural networks with Gaussian processes using six different covariance functions listed in Subsection 3.1. All models were trained on the same set of training data $T_{k(A)}$ as was used in steps 6 and 9 of Algorithm 1. Due to the condition (3), this set is rather restricted and allows training only a restricted ANN to prevent overfitting. Therefore, we decided to use a multilayer perceptron with a single hidden layer, thus a topology (n_I, n_H, n_O), where n_I is the dimension of the training data, i.e. $n_I \in \{2,3,5,10,20\}$, and

$$n_H = n_O = \begin{cases} 2 & \text{if } n_I = 2, \\ 3 & \text{if } n_I = 3,5, \\ 5 & \text{if } n_I = 10,20. \end{cases} \tag{13}$$

As the activation function for both the hidden and output layer, we chose the logistic sigmoid. We trained the weights and biases of the neural network together with the parameters of the Gaussian process as proposed in [52] and outlined in Subsection 3.2. As a loss function, we used the Gaussian log-likelihood and optimized the parameters with Adam [30] for a maximum of 1000 iterations. We also kept a 10 % validation set out of the training data to monitor overfitting, and we selected the model with the lowest L_2 validation error during the training.

4.3 Results

To evaluate different covariance functions, we used the recorded data in a similar way as it would be used by DTS-CMA-ES algorithm. We took each generation of points except the first one and we used it as testing data. Every point evaluated by the true objective function in all previous generations is then available as a training sample. We filter these samples using the *training set selection* method $T_{k(A)}$ and trained the surrogate model on it.

This way, we evaluated six different models on every generation of samples listed in Table 1. The results are presented first in a function-method view in Table 4, then in a dimension-method view in Table 5. The metric used in both tables is RDE, which shows how precise the model is in ordering of the predicted values, therefore the lower is the error value, the better. The first table contains results for every benchmark function separately, averaged over every input dimension, function instance and DTS-CMA-ES generation as well as the average over the whole

group of functions. The second table provides a view on how the dimension of the input space affects the approximation error. The results for a particular dimension are averaged over all functions in a specific group, instances, and generations. We also visualized with boxplots the summarized RDE values for the function groups in Figure 2.

Moreover, we verified the results by performing multiple comparison tests. A Non-parametric Friedman test was conducted on RDEs across all results for particular functions and function types in Table 4, and for a particular combination of dimensions and function types in Table 5. If the null hypothesis of the equality of all six considered methods was declined, the Wilcoxon signed-rank test was performed for all pairs of covariance functions, and its results were corrected for multiple hypotheses testing using the Holm method. We summarized the results of statistical testing in Tables 2 and 3.

Table 2: Statistical comparison of six covariance functions from the function-method view in Table 4. It shows for how many functions (23 in total) was the kernel in a row significantly better than the one in a column.

	κ_{LIN}	κ_{Q}	κ_{RQ}	κ_{SE}	κ_{Mat}	$\kappa_{\text{SE+Q}}$	Σ
κ_{LIN}	–	20	19	21	20	20	100
κ_{Q}	0	–	3	2	3	16	24
κ_{RQ}	1	5	–	1	1	13	21
κ_{SE}	2	6	1	–	1	14	24
κ_{Mat}	2	6	2	0	–	13	23
$\kappa_{\text{SE+Q}}$	2	1	2	0	1	–	6

Table 3: Statistical comparison of six covariance functions from the dimension-method view in Table 5. It shows for how many combinations of input dimension and a specific function group (25 in total) was the kernel in a row significantly better than the one in a column.

	κ_{LIN}	κ_{Q}	κ_{RQ}	κ_{SE}	κ_{Mat}	$\kappa_{\text{SE+Q}}$	Σ
κ_{LIN}	–	25	23	21	20	24	113
κ_{Q}	0	–	4	0	3	16	23
κ_{RQ}	0	4	–	2	1	13	20
κ_{SE}	0	4	5	–	5	13	27
κ_{Mat}	0	2	3	1	–	9	15
$\kappa_{\text{SE+Q}}$	0	0	0	0	0	–	0

4.4 Discussion and Further Research

The results show that the combined model performs best with the simplest linear kernel. We compared the GP-ANN with a linear kernel with pure Gaussian processes in paper [32]. We found out that if we compare the combined GP-ANN with GP both with linear kernels, the neural extensions can bring better results in some cases. However, using more complex covariance functions with GP is still better. Therefore, we tried to apply it also to the GP-ANN combination. Unfortunately, the results with other kernels ended up much worse.

In our future research, we would like to try to overcome this. We want to systematically investigate different ANN topologies, including the direction of deep Gaussian processes [8, 12, 22, 23], in which only the topology is used from an ANN, but all neurons are replaced by GPs. Moreover, in addition to the selection of the training set used in DTS-CMA-ES and described in Algorithm 1, we want to consider also alternative ways of training set selection, allowing to train larger networks. Finally, we intend to perform research into transfer learning of surrogate models: An ANN-GP model with a deep neural network will be trained on data from many optimization runs, such as those employed in this paper, and then the model used in a new run of the same optimizer will be obtained through additional fine tuning restricted only to the GP and last 1-2 layers of the ANN.

We would also like to try to change the way how the model is trained. In paper [52] the parameters of Gaussian process are learned together with the parameters of the neural network. We think that this might not work well in the domain of surrogate modeling in black-box optimization. Therefore, we will try to train the network and Gaussian process separately.

5 Conclusion

In this paper, we examined an extension of Gaussian processes used in surrogate modeling. At the beginning, we described the CMA-ES algorithm for black-box optimization and its surrogate-assisted variant DTS-CMA-ES. Then we outlined Gaussian processes, various covariance functions, and their neural network extension. The presented research is our first attempt in the application of the combination of Gaussian processes with neural networks as a surrogate model in black-box optimization. We implemented the combined model and compared the results obtained with six different covariance functions. By looking at the results presented in the previous section, we can conclude that the combined model yields the best results with the simplest linear kernel. The other covariance functions produce much larger errors and the worst performing one seems to be the composite kernel $\kappa_{\text{SE+Q}}$. Unfortunatelly the results showed up to be worse than using Gaussian processes alone, therefore we discuss possible ideas of further improvements at the end of the paper.

Acknowledgement

The research reported in this paper has been supported by SVV project number 260 575 and was also partially supported by Czech Science Foundation (GAČR) grant 18-18080S. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

Table 4: Comparison of average RDE values for six different covariance functions depending on a particular benchmark function. The values are averaged over different dimensions and instances.

	f	κ_{LIN}	κ_{Q}	κ_{RQ}	κ_{SE}	κ_{Mat}	$\kappa_{\mathrm{SE+Q}}$
Separable Functions	1	0.242	0.528	0.486	0.469	0.476	0.591
	2	0.128	0.386	0.475	0.486	0.503	0.491
	3	0.300	0.452	0.449	0.462	0.443	0.546
	4	0.352	0.480	0.459	0.462	0.462	0.548
	all	0.255	0.461	0.467	0.469	0.471	0.544
Moderate conditioning	6	0.351	0.446	0.410	0.451	0.451	0.421
	7	0.202	0.461	0.485	0.476	0.499	0.529
	8	0.330	0.463	0.410	0.406	0.460	0.527
	9	0.326	0.465	0.463	0.435	0.430	0.530
	all	0.302	0.458	0.441	0.442	0.459	0.501
High conditioning and unimodal	10	0.142	0.458	0.505	0.457	0.484	0.487
	11	0.147	0.417	0.537	0.527	0.525	0.488
	12	0.284	0.436	0.506	0.498	0.510	0.451
	13	0.216	0.434	0.455	0.418	0.433	0.532
	14	0.289	0.498	0.452	0.405	0.416	0.551
	all	0.215	0.448	0.491	0.461	0.473	0.501
Multi-modal adequate global structure	15	0.321	0.462	0.467	0.446	0.427	0.545
	16	0.551	0.530	0.516	0.486	0.489	0.504
	17	0.391	0.450	0.487	0.474	0.483	0.550
	18	0.380	0.486	0.489	0.472	0.480	0.553
	19	0.400	0.540	0.524	0.521	0.526	0.536
	all	0.408	0.493	0.496	0.479	0.480	0.537
Multi-modal weak global structure	20	0.213	0.508	0.467	0.456	0.472	0.547
	21	0.387	0.470	0.467	0.476	0.468	0.532
	22	0.347	0.480	0.484	0.467	0.475	0.532
	23	0.589	0.589	0.537	0.544	0.547	0.549
	24	0.448	0.492	0.509	0.516	0.506	0.506
	all	0.396	0.507	0.492	0.491	0.493	0.533

Table 5: Comparison of average RDE values for six different covariance functions depending on the type of benchmark function and the input space dimension. The values are averaged over different functions in particular group and instances.

	f	κ_{LIN}	κ_{Q}	κ_{RQ}	κ_{SE}	κ_{Mat}	$\kappa_{\mathrm{SE+Q}}$
Dimension 2	SEP	0.252	0.452	0.448	0.445	0.418	0.581
	MOD	0.341	0.450	0.450	0.483	0.455	0.471
	HC	0.204	0.449	0.459	0.474	0.441	0.502
	MMA	0.416	0.485	0.472	0.475	0.484	0.550
	MMW	0.443	0.517	0.419	0.471	0.431	0.522
	all	0.334	0.473	0.450	0.470	0.447	0.525
Dimension 3	SEP	0.194	0.439	0.414	0.455	0.447	0.527
	MOD	0.262	0.449	0.383	0.407	0.429	0.512
	HC	0.174	0.425	0.425	0.451	0.464	0.489
	MMA	0.353	0.486	0.449	0.461	0.453	0.536
	MMW	0.387	0.512	0.467	0.485	0.464	0.527
	all	0.278	0.464	0.430	0.454	0.453	0.518
Dimension 5	SEP	0.227	0.445	0.473	0.479	0.478	0.526
	MOD	0.260	0.458	0.444	0.458	0.468	0.502
	HC	0.202	0.439	0.513	0.489	0.486	0.492
	MMA	0.373	0.481	0.491	0.500	0.482	0.517
	MMW	0.363	0.513	0.496	0.504	0.499	0.532
	all	0.289	0.468	0.486	0.487	0.484	0.514
Dimension 10	SEP	0.278	0.479	0.476	0.500	0.494	0.536
	MOD	0.285	0.460	0.453	0.417	0.443	0.502
	HC	0.206	0.464	0.515	0.438	0.478	0.507
	MMA	0.422	0.498	0.525	0.499	0.478	0.549
	MMW	0.363	0.500	0.530	0.525	0.530	0.549
	all	0.313	0.481	0.503	0.477	0.486	0.529
Dimension 20	SEP	0.327	0.493	0.525	0.469	0.518	0.551
	MOD	0.362	0.475	0.479	0.447	0.504	0.522
	HC	0.293	0.467	0.543	0.454	0.500	0.519
	MMA	0.478	0.518	0.546	0.464	0.508	0.537
	MMW	0.429	0.496	0.551	0.475	0.544	0.536
	all	0.381	0.490	0.531	0.462	0.515	0.533

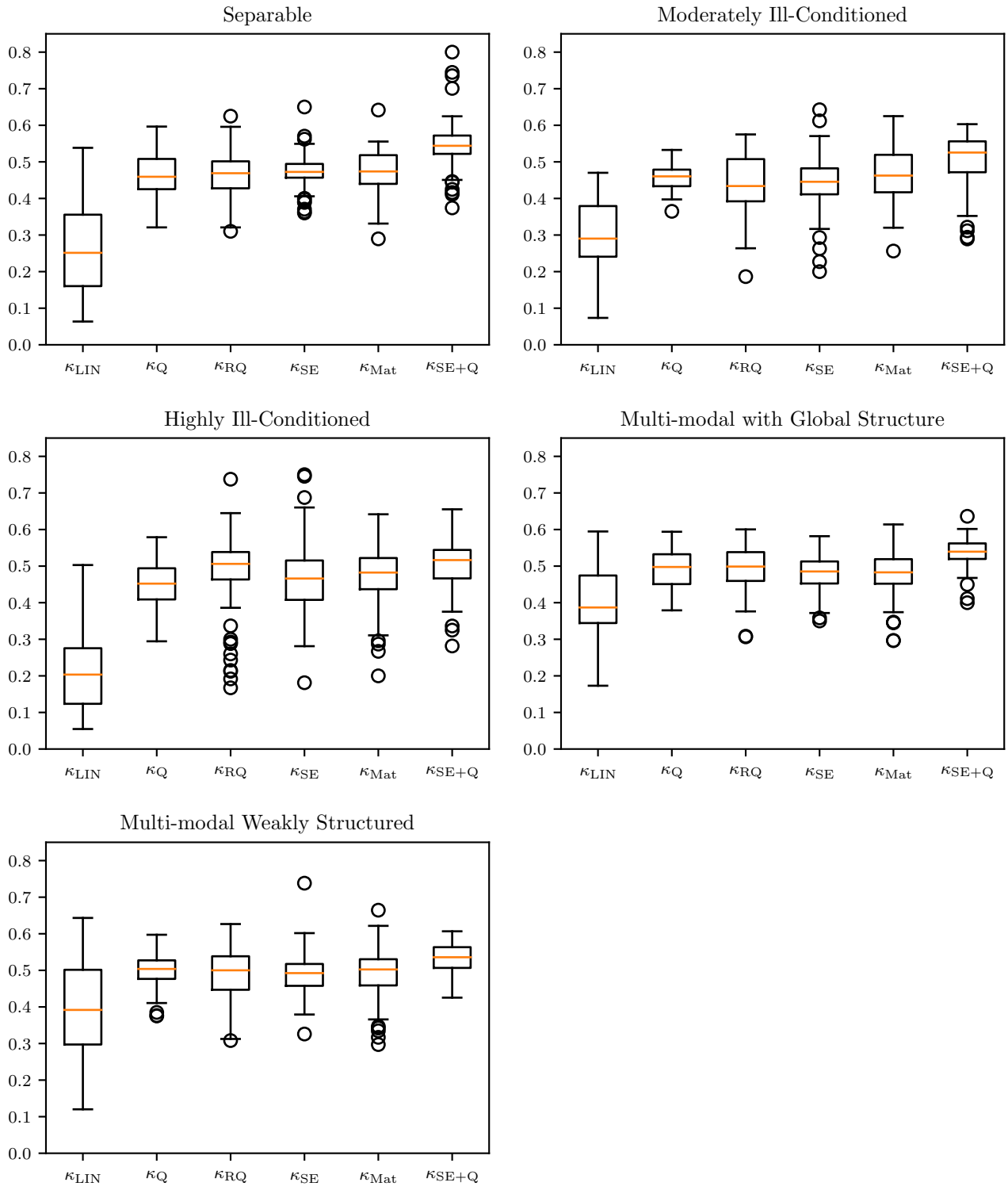

Figure 2: Visualization of the distribution of RDE values for a specific group of functions. Each boxplot corresponds to the aggregated mean value in the grey rows in Table 4.

References

[1] A. Auger, D. Brockhoff, and N. Hansen. Benchmarking the local metamodel cma-es on the noiseless BBOB'2013 test bed. In *GECCO'13*, pages 1225–1232, 2013.

[2] A. Auger, M. Schoenauer, and N. Vanhaecke. LS-CMA-ES: A second-order algorithm for covariance matrix adaptation. In *Parallel Problem Solving from Nature - PPSN VIII*, pages 182–191, 2004.

[3] M. Baerns and M. Holeňa. *Combinatorial Development of Solid Catalytic Materials. Design of High-Throughput Experiments, Data Analysis, Data Mining*. Imperial College Press / World Scientific, London, 2009.

[4] L. Bajer and Z. Pitra. Surrogate CMA-ES. `https://github.com/bajeluk/surrogate-cmaes`, 2021.

[5] L. Bajer, Z. Pitra, and M. Holeňa. Benchmarking Gaussian processes and random forests surrogate models on the BBOB noiseless testbed. In *GECCO'15 Companion*, pages 1143–1150, 2015.

[6] L. Bajer, Z. Pitra, J. Repický, and M. Holeňa. Gaussian process surrogate models for the CMA evolution strategy. *Evolutionary Computation*, 27:665–697, 2019.

[7] A.J. Booker, J. Dennis, P.D. Frank, D.B. Serafini, Torczon V., and M. Trosset. A rigorous framework for optimization by surrogates. *Structural and Multidisciplinary Optimization*, 17:1–13, 1999.

[8] T. Bui, D. Hernandez-Lobato, J. Hernandez-Lobato, Y. Li, and R. Turner. Deep Gaussian processes for regression using approximate expectation propagation. In *ICML*, pages 1472–1481, 2016.

[9] R. Calandra, J. Peters, C.E. Rasmussen, and M.P. Deisenroth. Manifold Gaussian processes for regression. In *IJCNN*, pages 3338–3345, 2016.

[10] S.M. Clarke, J.H. Griebisch, and T.W. Simpson. Analysis of support vector regression for approximation of complex engineering analyses. *Journal of Mechanical Design*, 127:1077–1087, 2005.

[11] The COCO platform, 2016. http://coco.gforge.inria.fr.

[12] K. Cutajar, E.V. Bonilla, P. Michiardi, and M. Filippone. Random feature expansions for deep Gaussian processes. In *ICML*, pages 884–893, 2017.

[13] M.A. El-Beltagy, P.B. Nair, and A.J. Keane. Metamodeling techniques for evolutionary optimization of computaitonally expensive problems: Promises and limitations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 196–203. Morgan Kaufmann Publishers, 1999.

[14] M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and multi-objective evolutionary optimization assisted by Gaussian random field metamodels,. *IEEE Transactions on Evolutionary Computation*, 10:421–439, 2006.

[15] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In *PPSN VII*, pages 361–370. ACM, 2002.

[16] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Presentation of the noisy functions. Technical report, INRIA, Paris Saclay, 2010.

[17] A. Forrester, A. Sobester, and A. Keane. *Engineering Design via Surrogate Modelling: A Practical Guide*. John Wiley and Sons, New York, 2008.

[18] H.M. Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19:201–227, 2001.

[19] N. Hansen. The CMA evolution strategy: A comparing review. In *Towards a New Evolutionary Computation*, pages 75–102. Springer, 2006.

[20] N. Hansen. A global surrogate assisted CMA-ES. In *GECCO'19*, pages 664–672, 2019.

[21] N. Hansen and A. Ostermaier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9:159–195, 2001.

[22] A. Hebbal, L. Brevault, M. Balesdent, E.G. Talbi, and N. Melab. Efficient global optimization using deep Gaussian processes. In *IEEE CEC*, pages 1–12, 2018.

[23] G. Hernández-Muñoz, C. Villacampa-Calvo, and D. Hernández Lobato. Deep Gaussian processes using expectation propagation and Monte Carlo methods. In *ECML PKDD*, pages 1–17, paper no. 128, 2020.

[24] S. Hosder, L. Watson, and B. Grossman. Polynomial response surface approximations for the multidisciplinary design optimization of a high speed civil transport. *Optimization and Engineering*, 2:431–452, 2001.

[25] Y. Jin, M. Hüsken, M. Olhofer, and Sendhoff B. Neural networks for fitness approximation in evolutionary optimization. In Y. Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, pages 281–306. Springer, 2005.

[26] Y. Jin, M. Olhofer, and B. Sendhoff. Managing approximate models in evolutionary aerodynamic design optimization. In *CEC 2001*, pages 592–599, 2001.

[27] Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6:481–494, 2002.

[28] D.R. Jones, M. Schonlau, and W.J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[29] S. Kern, N. Hansen, and P. Koumoutsakos. Local metamodels for optimization using evolution strategies. In *PPSN IX*, pages 939–948, 2006.

[30] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. Preprint arXiv:1412.6980, 2014.

[31] J. Koza and J. Tumpach. Surrogate networks. `https://github.com/c0zzy/surrogate-networks`, 2021.

[32] J. Koza, J. Tumpach, Z. Pitra, and M. Holeňa. Using past experience for configuration of Gaussian processes in black-box optimization. In *15th Learning and Intelligent Optimization Conference*, page accepted for publication, 2021.

[33] J.W. Kruisselbrink, M.T.M. Emmerich, A.H. Deutz, and T. Bäck. A robust optimization approach using kriging metamodels for robustness approximation in the CMA-ES. In *IEEE CEC*, pages 1–8, 2010.

[34] S.J. Leary, A. Bhaskar, and A.J. Keane. A derivative based surrogate model for approximating and optimizing the output of an expensive computer simulation. *Journal of Global Optimization*, 30:39–58, 2004.

[35] J. Lee, Y. Bahri, R. Novak, S.S. Schoenholz, J. Pennington, et al. Deep neural networks as Gaussian processes. In *ICLR*, pages 1–17, 2018.

[36] I. Loshchilov, M. Schoenauer, and M. Sebag. Intensive surrogate model exploitation in self-adaptive surrogate-assisted CMA-ES (saACM-ES). In *GECCO'13*, pages 439–446, 2013.

[37] J. Lu, B. Li, and Y. Jin. An evolution strategy assisted by an ensemble of local Gaussian process models. In *GECCO'13*, pages 447–454, 2013.

[38] J.R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley and Sons, Chichester, 2007.

[39] A.G.G. Matthews, J. Hron, M. Rowland, and R.E. Turner. Gaussian process behaviour in wide deep neural networks. In *ICLR*, pages 1–15, 2019.

[40] R.H. Myers, D.C. Montgomery, and C.M. Anderson-Cook. *Response Surface Methodology: Proces and Product Optimization Using Designed Experiments*. John Wiley and Sons, Hoboken, 2009.

[41] R. Novak, L. Xiao, J. Lee, Y. Bahri, G. Yang, et al. Bayesian deep convolutional networks with many channels are Gaussian processes. In *ICLR*, pages 1–35, 2019.

[42] Y.S. Ong, P.B. Nair, A.J. Keane, and K.W. Wong. Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In Y. Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, pages 307–331. Springer, 2005.

[43] Z. Pitra, M. Hanuš, J. Koza, J. Tumpach, and M. Holeňa. Interaction between model and its evolution control in surrogate-assisted CMA evolution strategy. In *GECCO'21*, page paper no. 358, 2021.

[44] Z. Pitra, J. Repický, and M. Holeňa. Boosted regression forest for the doubly trained surrogate covariance matrix adaptation evolution strategy. In *ITAT 2018*, pages 72–79, 2018.

[45] K. Rasheed, X. Ni, and S. Vattam. Methods for using surrogate modesl to speed up genetic algorithm oprimization: Informed operators and genetic engineering. In Y. Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, pages 103–123. Springer, 2005.

[46] C.E Rasmussen and N. Hansen. GPML 4.0. matlab toolbox. `http://www.gaussianprocess.org/gpml/code/matlab/doc/`.

[47] E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, 2006.

[48] A. Ratle. Kriging as a surrogate fitness landscape in evolutionary optimization. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 15:37–49, 2001.

[49] H. Ulmer, F. Streichert, and A. Zell. Evolution strategies assisted by Gaussian processes with improved pre-selection criterion. In *IEEE CEC*, pages 692–699, 2003.

[50] University of California in Irvine. Repository of machine learning databases. http://www.ics.uci.edu/∼mlearn, 2016.

[51] V. Volz, G. Rudolph, and B. Naujoks. Investigating uncertainty propagation in surrogate-assisted evolutionary algorithms. In *GECCO'17*, pages 881–888, 2017.

[52] A.G. Wilson, Z. Hu, R. Salakhutdinov, and E.P. Xing. Deep kernel learning. In *ICAIS*, pages 370–378, 2016.

[53] Z.Z. Zhou, Y.S. Ong, P.B. Nair, A.J. Keane, and K.Y. Lum. Combining global and local surrogate models to accellerate evolutionary optimization. *IEEE Transactions on Systems, Man and Cybernetics. Part C: Applications and Reviews*, 37:66–76, 2007.