Exploring Dimensionality Reduction Techniques for Efficient Surrogate-Assisted Optimization

Sibghat Ullah*, Duc Anh Nguyen*, Hao Wang†, Stefan Menzel§, Bernhard Sendhoff§, and Thomas Bäck*
*Leiden Institute of Advanced Computer Science (LIACS), Leiden University, The Netherlands
Email: {s.ullah,d.a.nguyen,t.h.w.baeck}@liacs.leidenuniv.nl
†Sorbonne Université, CNRS, LIP6, Paris, France
Email: hao.wang@lip6.fr
§Honda Research Institute Europe GmbH (HRI-EU), Offenbach/Main, Germany
Email: {stefan.menzel,bernhard.sendhoff}@honda-ri.de

***Abstract*—Constructing surrogate models of high dimensional optimization problems is challenging due to the computational complexity involved. This paper empirically investigates the practicality of major dimensionality reduction techniques for encapsulating the high dimensional design space into compact representations. Such low dimensional representations of the design space can be utilized for constructing the surrogate models efficiently. Based on historical mainstays and recent developments in deep learning, we study four dimensionality reduction techniques in this paper, namely Principal Component Analysis, Kernel Principal Component Analysis, Autoencoders and Variational Autoencoders. We evaluate and compare these techniques based on quality assessments of the corresponding low dimensional surrogate models on a diverse range of test cases. These test cases are defined on combinations of three dimensionsalities, ten well-known benchmark problems from the continuous optimization domain and two surrogate modeling techniques, namely Kriging and Polynomials. Our results clearly demonstrate the superiority of Autoencoders and Principal Component Analysis on the criteria of modeling accuracy and global optimality respectively.**

***Index Terms*—dimensionality reduction, surrogate-assisted optimization, machine learning, deep latent-variable models, principal component analysis**

I. Introduction

Continuous optimization problems in real-world application domains, e.g., mechanics, engineering, economics and finance, can encompass some of the most complicated optimization setups. Principal obstacles in solving the optimization tasks in these areas involve multimodality [1], high dimensionality [2] and unexpected drifts and changes in the optimization setup [3], [4]. Due to these obstacles and the black-box assumption on the optimization setup, traditional optimization schemes, e.g., gradient descent and Newton methods, are rendered inapplicable [5]. The majority of the optimization schemes applied in these areas now focus on utilizing direct search methods [3], [6], in particular Evolutionary Algorithms (EAs) [7] and Surrogate-Assisted Optimization (SAO) [8]. In this work, we focus on SAO for high dimensional cases.

SAO [8] refers to solving the optimization problem with the help of a surrogate model, which replaces the actual function evaluations by the model prediction. The surrogate model approximates the true values of the objective function under consideration. This is generally done if the objective function is too complex and/or costly, and is therefore hard to optimize directly [9]. The abstraction provided by the surrogate models is useful in a variety of situations. Firstly, it simplifies the task to a great extent in simulation based modeling [8] and optimization. Secondly, it provides the opportunity to evaluate the fitness function indirectly if the exact computation is intractable [9]. In addition, surrogate models can provide practically useful insights, e.g., space visualization and comprehension [8]. Despite these advantages, SAO faces many limitations [10] in constraint handling, dynamic optimization, multi-objective optimization and high dimensional optimization.

Modeling high dimensional optimization problems with SAO is challenging [2] due to two main reasons. Firstly, more training data is required to achieve a comparable level of modeling accuracy as the dimensionality increases [8]. Secondly, training time complexity often increases rapidly w.r.t. both, the dimensionality and the number of training data points [8], [10]. Consequently, constructing the surrogate model becomes costlier. To highlight this issue, upper bounds on the time complexities of the most common surrogate models are presented in Table I. Note that in Table I, D, N, N_{trees}, N_{sv} and K stand for the dimensionality, the number of training data points, the number of trees in Random Forest (RF), the number of support vectors in Support Vector Machines (SVMs) and the number of neighbours in K-Nearest Neighbours (KNNs) respectively. From Table I, it can be deduced that higher dimensionality can severely affect the computational budget in SAO in two ways: directly, i.e., by a higher value of D, and indirectly, i.e., by a higher value of N, N_{trees}, N_{sv} and K.

Various methodologies have been proposed to deal with the issue of high dimensionality in SAO including divide-and-conquer [11], variable screening [12] and mapping the data space to a lower dimensional space [13] using dimensionality reduction techniques (DRTs). One of the most common DRTs is Principal Component Analysis (PCA) [14]. PCA can be defined as the orthogonal projection of the data onto a lower dimensional linear space, known as the "principal subspace", such that the variance of the projected data is maximized [15]. Various generalized extensions of PCA have been established

TABLE I
TRAINING AND PREDICTION TIME COMPLEXITIES OF THE MOST COMMON SURROGATE MODELS. NOTATION: N_{TREES} IS THE NUMBER OF TREES IN RANDOM FOREST, N_{SV} IS THE NUMBER OF SUPPORT VECTORS, AND K THE NUMBER OF NEIGHBOURS IN K-NEAREST NEIGHBOURS.

Model	Training	Prediction
Quadratic Regression	$O(D^4N + D^9 + D^2)$	$O(D^2)$
Random Forest	$O(N^2DN_{\text{trees}})$	$O(NN_{\text{trees}})$
Support Vector Machines	$O(N^2D + N^3)$	$O(DN_{\text{sv}})$
K-Nearest Neighbours	$O(1)$	$O(KD)$
Kriging	$O(N^3D)$	$O(ND)$

in the literature such as Kernel PCA [16], Probabilistic PCA [17], [18] and Bayesian PCA [19]. On the other hand, Autoencoders (AEs) [20], [21] have been contemplated as feed-forward neural networks (FFNNs) trained to attempt to copy their input to their output, so as to learn the useful low dimensional encoding of the data. Like PCA, AEs have also been extended over the years by generalized frameworks such as Sparse Autoencoders [22], Denoising Autoencoders [23], Contractive Autoencoders [24] and Variational Autoencoders (VAEs) [25]–[27]. Besides PCA and AEs, other important DRTs include Isomap [28], Locally-Linear Embedding [29], Laplacian Eigenmaps [30], Curvilinear component analysis [31] and t-distributed stochastic neighbor embedding [32].

In this paper, we evaluate and compare four of the most important DRTs mentioned above, namely PCA, Kernel PCA, AEs and VAEs. We choose PCA and AEs due to their historical significance, since both have been employed regularly for dimensionality reduction, lossy data compression, feature learning and data visualization [15], [20], [33] in machine learning. We incorporate Kernel PCA due to the generalized non-linear extension of the classical PCA algorithm [16]. Similarly, we consider VAEs in this paper due to the presence of the non-linear stochastic encodings [25], [34] of the data space which can be utilized for constructing the surrogate models efficiently. The focal point of this paper is to provide a novel perspective on the applicability of these DRTs in SAO. This is accomplished by performing an extensive quality assessment of the corresponding low dimensional surrogate models (LDSMs) on a diverse range of test cases. For a comprehensive overview on DRTs, please refer to the survey papers [35], [36].

The remainder of this paper is organized as follows. We present the basic introduction to surrogate modeling and the DRTs in section II. Section III provides the blueprint for dimensionality reduction in SAO. In section IV, we present the experimental design to empirically evaluate and compare these techniques based on quality assessment of the corresponding LDSMs. This is followed by our experimental results in section V. Finally, we discuss the conclusions of the paper along side potential future research in section VI.

II. BACKGROUND

In this paper, we are aiming at minimizing real-valued black-box problems $f\colon \mathcal{S} \subseteq \mathbb{R}^D \to \mathbb{R}$, where we only have access to evaluating the function value and any analytical properties, e.g., gradient/Hessian/convexity, are not available to the optimization process [2]. In this section, we first provide a brief introduction to surrogate modeling and then move forward to discuss the DRTs studied in this paper.

A. Surrogate Modeling

By constructing surrogate models [8], we aim to learn an approximation $\hat{f}(\mathbf{x})$ of the target problem f, based on some evaluated data points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ [8]. Various sampling methods are proposed to generate such data points, e.g., Latin hyper-cube sampling [37], Plackett-Burman [38], and Box-Behnken [39] design. The quality of the surrogate models is significantly determined by the training sample size N, the choice of sampling method and the criteria to appraise the surrogate models.

B. Principal Component Analysis

Principal Component Analysis (PCA) is a data preprocessing method, which is commonly employed for dimensionality reduction, feature engineering, and data visualization [14], [15]. PCA learns a linear map $\mathbb{R}^D \to \mathbb{R}^D$ that transforms the original data set to a centered and uncorrelated one, meaning after the transformation, the sample mean is zero and the sample covariance matrix is diagonal [15]. Denoting by $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^\top$ the design matrix, PCA starts with calculating its sample mean $\boldsymbol{\mu} = (\frac{1}{N}\sum_{i=1}^N X_{i1}, \ldots, \frac{1}{N}\sum_{i=1}^N X_{iD})$ and centering the original design matrix: $\bar{\mathbf{X}} = \mathbf{X} - \mathbf{1}_N\boldsymbol{\mu}$ ($\mathbf{1}_N$ is a vector containing N 1's). Then, the first principal component (PC) $\mathbf{u}_1$ can be identified by maximizing the variance of $\bar{\mathbf{X}}$ projected onto $\mathbf{u}_1$, namely $\mathbf{u}_1 = \operatorname{argmax}_{\|\mathbf{u}\|\leq 1} \operatorname{var}\{\bar{\mathbf{X}}\mathbf{u}\}$. Similarly, further components can be obtained by removing the already-computed PCs from $\bar{\mathbf{X}}$ and then solving the same maximization problem, i.e., $\bar{\mathbf{X}}_k = \bar{\mathbf{X}} - \sum_{i=1}^{k-1} \bar{\mathbf{X}}\mathbf{u}_i\mathbf{u}_i^\top$, $\mathbf{u}_k = \operatorname{argmax}_{\|\mathbf{u}\|\leq 1} \operatorname{var}\{\bar{\mathbf{X}}_k\mathbf{u}\}$. Note that the variance of data projections onto each PCs (denoted by $\sigma_i^2 = \operatorname{var}\{\bar{\mathbf{X}}\mathbf{u}_i\}$) is monotonically decreasing concerning the order of PCs. It is not difficult to verify that all PCs are necessarily the eigenvectors of the sample covariance matrix $\bar{\mathbf{X}}^\top\bar{\mathbf{X}}/N$, sorted with respect to the decreasing order of their corresponding eigenvalues. Using PCA for dimensionality reduction, we select a subset of computed PCs according to a user-specific criterion, e.g., to keep the first several PCs such that the variance of data projections onto them sum up to a satisfying percentage of the total variability of the original data. In this work, we shall select the first L PCs where L is the size of the latent dimensionality as described in the experimental section.

C. Kernel Principal Component Analysis

Kernel Principal Component Analysis (KPCA) takes the so-called kernel trick [16], transforms the original data points into a high-dimensional (usually infinite-dimensional) feature space using a non-linear feature map, and performs the linear PCA therein. Formally, we denote the feature map as $\phi\colon \mathbb{R}^D \to \mathcal{H}$, where the feature space $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) equipped with a

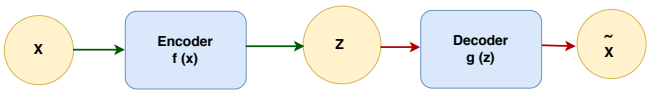

Fig. 1. A schematic diagram of AEs where green arrows indicate the encoding operation and the red ones are for the decoding operation.

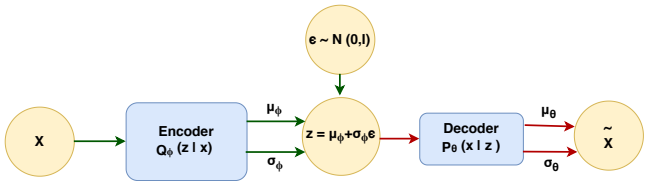

Fig. 2. A schematic diagram of VAEs where green arrows represents the variational inference and the red ones entails the generative part, i.e., the likelihood.

reproducing kernel $k(\mathbf{x}, \cdot) := \phi(\mathbf{x})$ and an inner product $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}} = k(\mathbf{x}, \mathbf{x}')$. For the transformed data points $\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_N)$, we first calculate their mean $\bar{\phi} = \frac{1}{N} \sum_{i=1}^{N} \phi(\mathbf{x}_i)$ and the pairwise inner product $\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{ij}$. Then, by seeking a point $u_1 \in \mathcal{H}$ that maximizes the variability of data projections onto it, we identify the first PC, namely $u_1 = \text{argmax}_{\|u\|_{\mathcal{H}} \leq 1} \frac{1}{N} \sum_{i=1}^{N} \langle u, \phi(\mathbf{x}_i) - \bar{\phi} \rangle_{\mathcal{H}}$. Without further derivations, we state that the solution to this problem is $u_1 = \sum_{i=1}^{N} \alpha_i^{(1)} (\phi(\mathbf{x}_i) - \bar{\phi})$, where $\alpha_i^{(1)}$ is the eigenvector that corresponds to the largest eigenvalue of matrix $\mathbf{HKH}$ ($\mathbf{H} = \mathbf{I}_{N \times N} - N^{-1} \mathbf{1}_N$). As with linear PCA, we proceed to compute further components in the same way after removing data projections onto the already-computed PCs. In essence, all PCs take the same form of u_1, and the coefficients thereof are determined by eigenvectors of $\mathbf{HKH}$.

D. Autoencoders

Autoencoders (AEs) [20], [21] are a family of deep generative models which approximate the data distribution by constructing a lower dimensional representation of the data points. An AE consists of two feed-forward neural networks (FFNNs) for the encoder and decoder respectively. The encoder $\mathbf{z} = f(\mathbf{x})$ takes the input $\mathbf{x}$ and produces a code $\mathbf{z}$ used to represent the input. The decoder takes this code and produces a reconstruction $\mathbf{x}' = g(\mathbf{z})$ of the original data. AEs are restricted in ways that allow them to learn the most salient features of the data. We employ Undercomplete Autoencoders (UAEs) in this paper, which are constrained to have smaller dimensions for $\mathbf{z}$ when compared with the original data $\mathbf{x}$. The learning process in UAEs focuses on minimizing a loss function $\mathcal{L}(\mathbf{x}, g(f(\mathbf{x})))$ such as mean squared error (MSE). A graphical illustration of a standard AE is presented in figure 1.

E. Variational Autoencoders

Variational autoencoders (VAEs) [25], [26] are AEs which provide a principled methodology for employing deep latent-variable models and can be used to learn complex data distributions in a generative fashion. The data $\mathbf{x} \in \mathbb{R}^D$ and the latent variables $\mathbf{z} \in \mathbb{R}^L$ are jointly related by the chain rule,

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) \tag{1}$$

where $L < D$, and θ denote the associated set of parameters. More specifically, a VAE consists of two coupled but independently parameterized models: an inference (also called an encoder or recognition) model and a generative (also called a decoder) model; both of which are implemented by non-linear functions such as neural networks [20], [27]. The encoder encodes the input data $\mathbf{x}$ to the set of latent variables $\mathbf{z}$, and the decoder maps these latent variables $\mathbf{z}$ back to reproduce $\mathbf{x}$. The VAE treats the conditional probability distribution $p_\theta(\mathbf{x}|\mathbf{z})$ as a function approximation of $\mathbf{x}$. However, the non-linear mapping from $\mathbf{z}$ to $\mathbf{x}$ can not be implemented directly because of the intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$ on the latent-variable. The VAE thus introduces the variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ parameterized by ϕ to approximate the intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$. The parameters for the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ are generated by the encoder network. Lastly, the variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$ of $p_\theta(\mathbf{z}|\mathbf{x})$ enables the use of Evidence Lower Bound (ELBO) such as:

$$\log p_\theta(\mathbf{x}) \geq -\text{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) + \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \tag{2}$$

where $\text{KL}(Q||P)$ is the Kullback-Leibler divergence between two probability distributions Q and P. In [25], the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$ is modeled by a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$, where the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are the outputs of the inference network and diag corresponds to the diagonal covariance structure of the Gaussian distribution. The prior $p(\mathbf{z})$ is assumed to be a standard Gaussian distribution. The training process focuses on maximizing ELBO which yields the optimal parameters for the inference and generative networks. A low variance estimator can be substituted with the help of the reparameterization trick $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon$; where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a vector of standard Gaussian variables and $\odot$ denotes the element-wise product:

$$\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] = \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[\log p_\theta(\mathbf{x}|\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon)] \tag{3}$$

In summary, VAEs are AEs which provide a principled framework to learn deep latent-variable models efficiently by combining the Variational Inference (VI) and the reparameterization trick. Due to this reason, they have become a very important tool for dimensionality reduction, lossy data compression, representation learning and generative modeling. Detailed discussion on VAEs, VI and the reparameterization trick can be found in [27], whereas the graphical representation of a VAE is presented in figure 2.

III. Dimensionality Reduction in Surrogate Modeling

As already discussed in section I, the high dimensionality is one of the major difficulties in the application of SAO. In this paper, we propose to conduct empirical comparisons among four major DRTs discussed previously for building low

dimensional surrogate models (LDSMs). In particular, we are interested in evaluating the effectiveness of the low dimensional representations of the data to substitute the features of the original data for constructing the LDSMs. The surrogate models constructed from these encodings can be represented as $\hat{f} : \mathcal{G} \subseteq \mathbb{R}^L \rightarrow \mathbb{R}$, where $\mathcal{G}$ is the low dimensional image of the original domain $\mathcal{S}$. Building the surrogate models in this lower dimensional space is beneficial for two main reasons. Firstly, we can alleviate the curse of dimensionality by controlling the value of L. Secondly, with some tolerance, low dimensional representations in $\mathcal{G}$ contain enough information to reconstruct the original features in $\mathcal{S}$. Together, these rationales allows us to make a compromise on the quality of the surrogate model and the computational budget. Intuitively, if L is very close to D, the performance of the LDSM is expected to be similar to the baseline surrogate with dimensionality D. On the other hand, if $L \ll D$, the performance of the LDSM is expected to be worse than the baseline due to the loss of information. Overall, we believe it is crucial to utilize the DRTs for constructing the surrogate models in high dimensional cases with limited computational resources. We now discuss the experimental setup to evaluate the effectiveness of the DRTs previously introduced to construct the LDSMs.

IV. Experimental Setup

Here, we briefly introduce the benchmark problem set chosen in this work and describe how training and testing data sets are generated from this problem set. Then, we present the performance metric used here and show details of the hyper-parameter optimization (HPO) procedure applied to the surrogate models. A flowchart of the entire experimental procedure is provided in figure 3 for clarification.

A. Test Cases

We select ten unconstrained, noiseless, single-objective optimization problems from the continuous benchmark function test-bed known as "Black-Box-Optimization-Benchmarking" (BBOB) [40]. Note that BBOB provides a total of twenty four such functions divided in five different categories namely "Separable Functions", "Functions with low or moderate conditioning", "Functions with high conditioning and unimodal", "Multi-modal functions with adequate global structure" and "Multi-modal functions with weak global structure" respectively. We select two functions from each of these categories to diversify the landscape of our test cases. The selected functions are $f2$, $f3$, $f7$, $f9$, $f10$, $f13$, $f15$, $f16$, $f20$ and $f24$ respectively. Each of these test functions is evaluated on three different values of dimensionality: 50, 100, and 200 respectively. Additionally, all test functions mentioned here are subject to minimization. Please refer to [40] for more details on the test functions and their characteristics.

B. Generating the Training Data

After the selection of the test cases, we move forward to the data generation and preprocessing. For the purpose of data generation, the choice of training sample size N is problem-dependent. The practical advice however is to begin with $N = \beta D$ [8], where β is usually a low valued scalar and D corresponds to the dimensionality of the problem. Therefore, we proceed with $\beta = 20$. Choosing $\beta = 20$ is based on previous empirical evidence [4] as this results in a training data set of moderate size, which is neither too small to train nor too big to hinder the computational efficiency. Additionally, the test data set with size $M = 0.2\beta D$ is generated to evaluate the modeling accuracy of the LDSMs as prescribed in section IV-E. Notably, we also make sure that the training and test data sets are completely disjoint, i.e., no data point is shared between the two sets. The sampling locations for both data sets are chosen using a maximum-distance Latin hyper-cube sampling [37] scheme. The data preprocessing in this study is a rather straightforward task involving only the rescaling of the features between 0 and 1.

C. Implementation Details

We employ four DRTs in this paper namely PCA, KPCA, AEs and VAEs respectively. For each of these techniques, specifying L is crucial since it may affect the quality of the corresponding LDSM. Therefore, for each distinct value of the original dimensionality D, we choose three values for L as: $L \in \{0.7D, 0.4D, 0.1D\}$. As an example, $L \in \{35, 20, 5\}$ when $D = 50$. An important thing to note is that in AEs and VAEs, both the encoder and the decoder have four hidden layers each with hyperbolic tangent non-linearity. For PCA and KPCA, we perform a linear transformation of the original features before performing the dimensionality reduction. We also choose two surrogate modeling techniques namely Kriging and Polynomial Regression (degree=2 with elastic-net penalty) [41]. Notably, both sets of techniques, i.e., the dimensionality reduction and the surrogate modeling techniques, have some hyper-parameters. Therefore, it is crucial to tune these hyper-parameters to get the best quality surrogate models.

D. Hyper-parameter Optimization

At this stage, however, we have a total of 720 cases due to four DRTs, two surrogate modeling techniques, ten test functions, three values of the original dimensionality D and three different values for L. Therefore, performing HPO for each of these 720 cases is infeasible. Hence, we reduce the number of cases to a total of 72 by aggregating the performance of the LDSMs on all ten test functions. This implies that we optimize the hyper-parameters for each of the 72 cases defined on combinations of three values of the original dimensionality D, three values of L, two surrogate modeling techniques and four DRTs. In each of these 72 cases, we optimize the hyper-parameters for both the dimensionality reduction and the surrogate modeling techniques together based on the aggregated quality of the corresponding LDSMs on all ten test functions. The quality assessment for an individual LDSM,

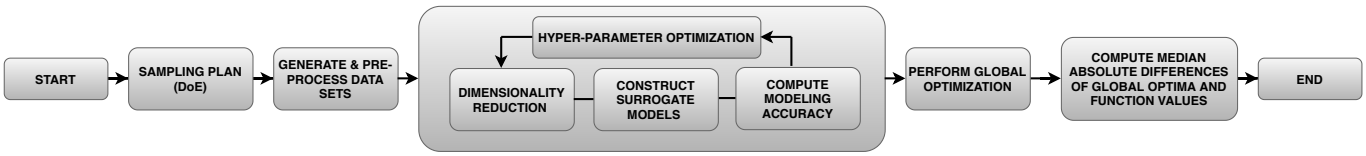

Fig. 3. Flowchart of the experimental setup. Each step of the process is shown in grey rectangles. The central rectangles indicates the hyper-parameter optimization loop based on the modeling accuracy of the surrogates.

i.e., for a particular test function such as $f2$, is measured by taking the so-called relative mean absolute error:

$$\text{RMAE} = \frac{1}{M}\sum_{i=1}^{M} 100\left(\frac{|y_i - \hat{y}_i|}{|y_i|}\right) \tag{4}$$

where y_i and $\hat{y}_i$ are the target and predicted values for the ith test data point and M denotes the size of the test data set. For HPO, we measure this RMAE for all ten test functions by specifying D, L, the dimensionality reduction and the surrogate modeling technique. After this, we take the median of the RMAE values on all ten test functions. The goal of the HPO then becomes to find out the best configuration of the hyper-parameters which minimizes this median. This process is repeated for all 72 cases. Overall, this approach makes the HPO feasible and ensures that the configuration of the hyper-parameters generalizes well across all ten test functions.

We employ Tree Parzen Estimator algorithm (TPE) [42], [43] to perform the HPO for each of the 72 cases discussed above by specifying D, L, the dimensionality reduction and the surrogate modeling technique. TPE algorithm is inspired from Bayesian Optimization and we utilize the Pythonic framework Hyperopt [43] to implement it. The number of function evaluations are restricted to 150 for finding the best configuration of the hyper-parameters using TPE as the maximum number of hyper-parameters in any of the 72 cases is six.

E. Evaluation Criteria

In this work, we choose two criteria to evaluate and compare the LDSMs. The first criterion is that of the modeling accuracy. To compare the LDSMs on this criterion, we first construct the LDSMs in all 720 cases after performing the HPO. This implies that we construct and compare four LDSMs for each distinct value of D, L, the surrogate modeling technique and the test function. These four LDSMs are based on PCA, KPCA, AEs and VAEs respectively. Note that here the LDSMs which share the same test function will also share the same configuration of the hyper-parameters as an implication of the HPO procedure discussed before. Since we vary the surrogate modeling technique, the test function, and the values for D and L, we can perform a comprehensive analysis of the modeling accuracy of the LDSMs based on a particular DRT. We employ RMAE, see Eq. (4), as the performance measure for this criterion.

The second criterion to compare the LDSMs is the quality of the proposed optimal solution. Please note again that all test functions are subject to minimization in this paper. To compare the LDSMs for this criterion, we proceed with the same setup as before. This implies that we construct the LDSM in each of the 720 cases based on the best configuration of the hyper-parameters, and employ the corresponding LDSM to substitute the exact function evaluations within the optimization loop of the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [44] for global optimization. For this, maximum function evaluations are restricted to $1000 \times D$. We perform a total of 30 runs of L-BFGS for each LDSM by varying the starting position, i.e., initial guess.

The resulting optimum in each of these 30 runs is plugged into the test function to achieve the optimal function value. After this procedure, we compare the LDSMs based on two aspects. Firstly, we find the median absolute difference of the globally optimal function value with the proposed optimal function values based on 30 runs of global optimization. This is done for each of the 720 cases. The second aspect to compare the LDSMs is the median absolute difference of the proposed optimums with the global optimum based on 30 runs, i.e., the median absolute difference of the globally optimal point on the search space with the optimal points proposed by the LDSM. Together, these two aspects allow us to make a comprehensive analysis on the performance of the LDSMs w.r.t the criterion of global optiamlity. We now move forward to share the results obtained from this experimental setup.

V. Results

In this section, we present the results obtained from the above experimental setup. We first share the results concerning the criterion of the modeling accuracy. For this, we share the graphs illustrating the modeling accuracy of the LDSMs using Kriging and Polynomial Regression (degree=2 with elastic-net penalty) in figures 4 and 5 respectively. Both figures contain a total of nine subplots each based on three distinct values of D and L respectively. Each subplot contains ten bar charts corresponding to the ten test functions discussed in section IV-A. Furthermore, each bar chart shares the RMAE of the four LDSMs based on the DRTs studied in this paper. From figures 4 and 5, we observe that the LDSMs based on AEs achieve the highest modeling accuracy, i.e., lowest RMAE values, in 132/720 cases. This is clearer to notice for $D \in \{50, 100\}$, and $L \in \{0.7D, 0.4D\}$ in both figures. In most of the remaining cases, the RMAE values are analogous, though there are some exceptional cases where the LDSMs based on other DRTs perform better. Notably, the modeling accuracy of the LDSMs increases with D, i.e., lower RMAE values, but the same is untrue for L. From figures 4 and 5, we can also observe that the LDSMs perform likewise for both surrogate modeling techniques.

Fig. 4. Modeling accuracy of the low dimensional Kriging surrogates for all test cases is presented. The test cases are defined on combinations of ten test functions, four dimensionality reduction techniques and three distinct values for D and L each.

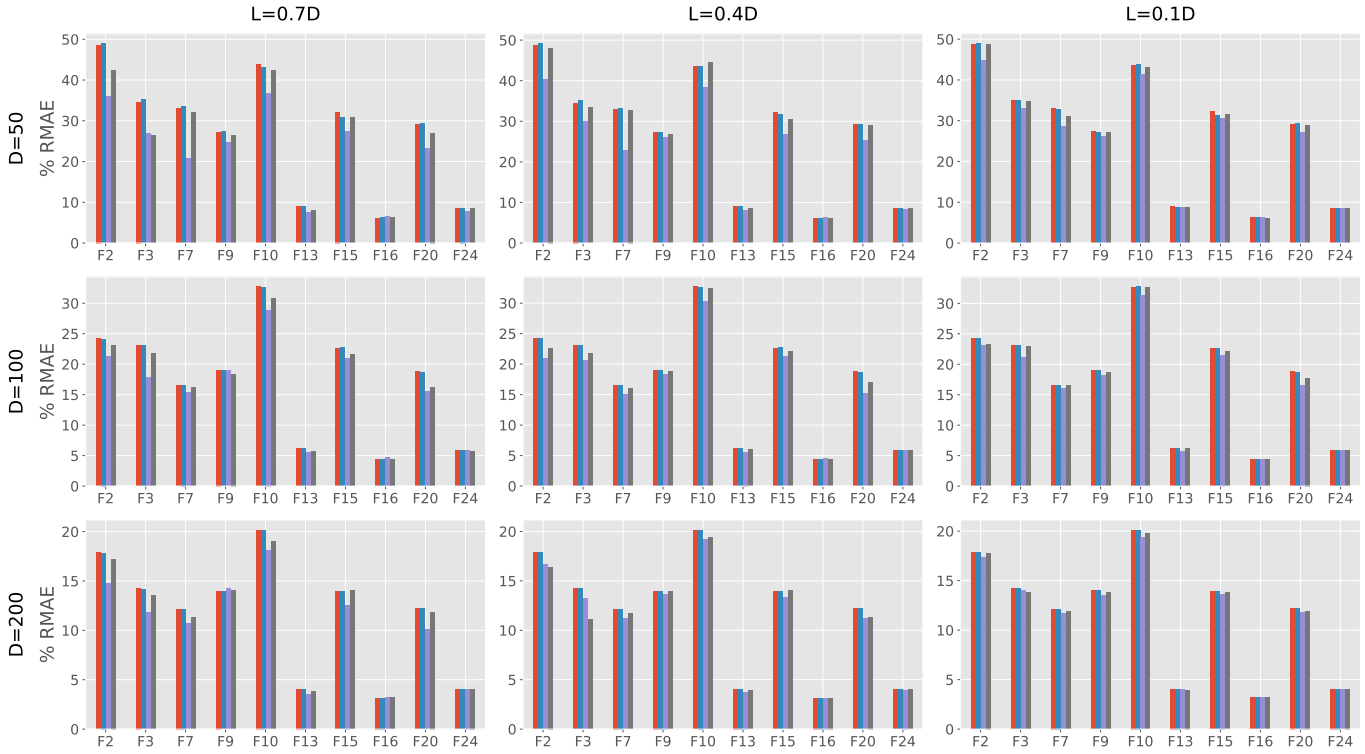

Fig. 5. Modeling accuracy of the low dimensional Polynomial surrogates (with degree=2 and elastic-net penalty) for all test cases is presented. The test cases are defined on combinations of ten test functions, four dimensionality reduction techniques and three distinct values for D and L each.

Fig. 6. Median absolute difference of the globally optimal function values with the proposed optimal function values for all test cases based on Kriging surrogates is presented. The test cases are defined on combinations of ten test functions, four dimensionality reduction techniques and three distinct values for D and L each.

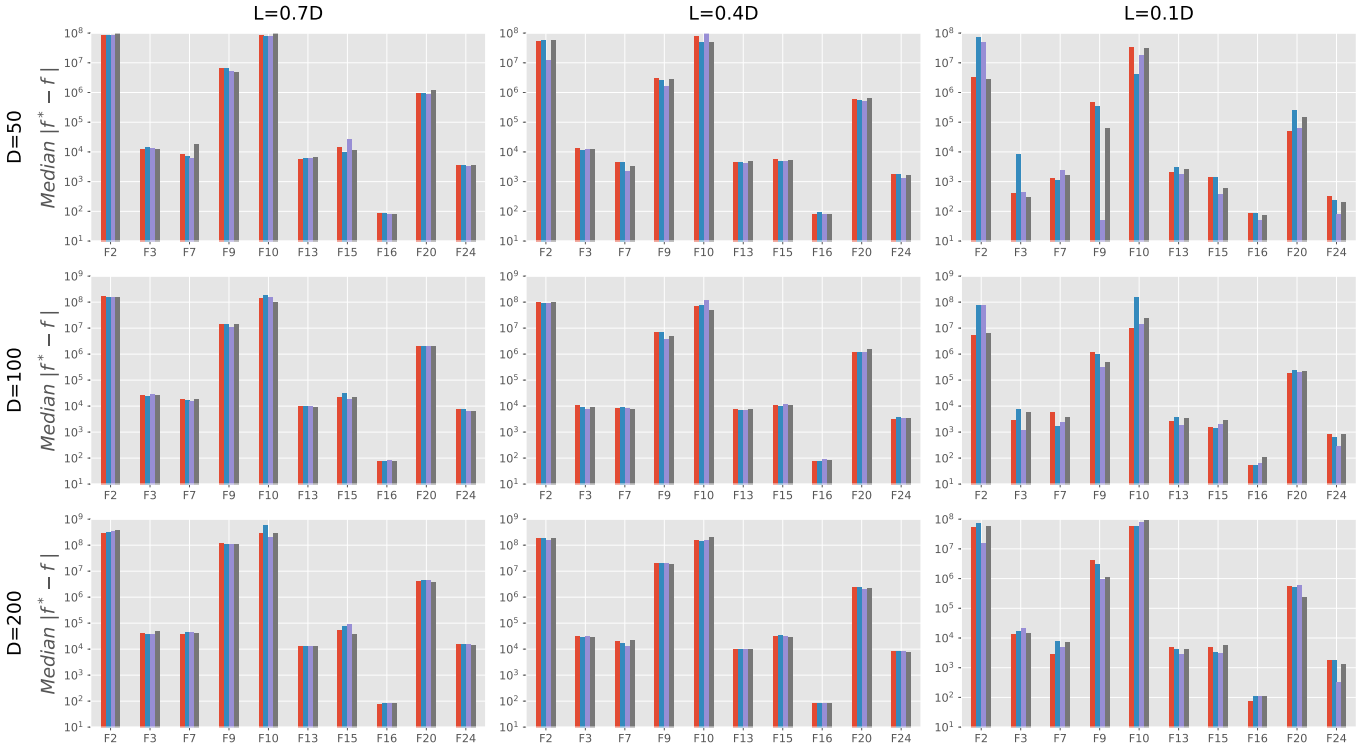

Fig. 7. Median absolute difference of the globally optimal function values with the proposed optimal function values for all test cases based on Polynomial surrogates (with degree=2 and elastic-net penalty) is presented. The test cases are defined on combinations of ten test functions, four dimensionality reduction techniques and three distinct values for D and L each.

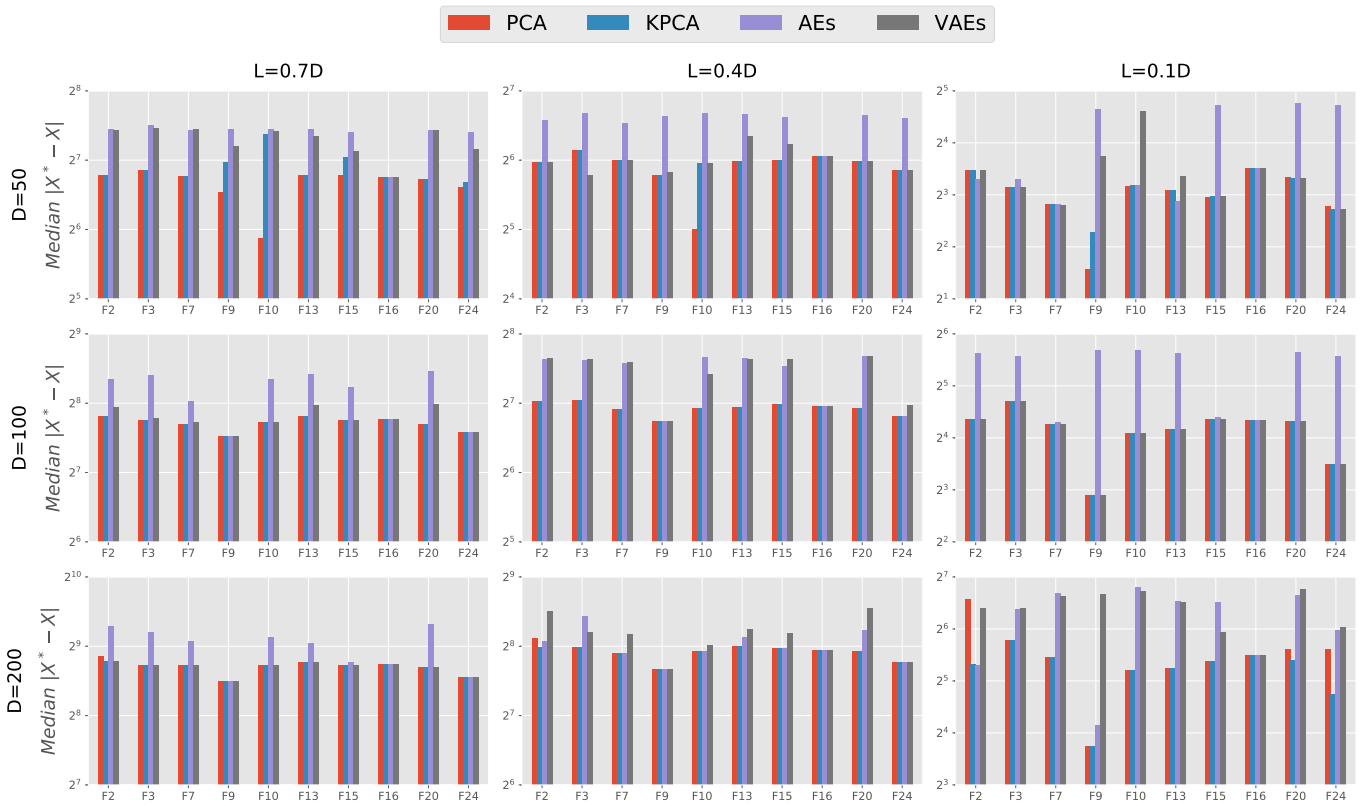

Fig. 8. Median absolute difference of the global optimum with the proposed optimum for all test cases based on Kriging surrogates is presented. The test cases are defined on combinations of ten test functions, four dimensionality reduction techniques and three distinct values for D and L each.

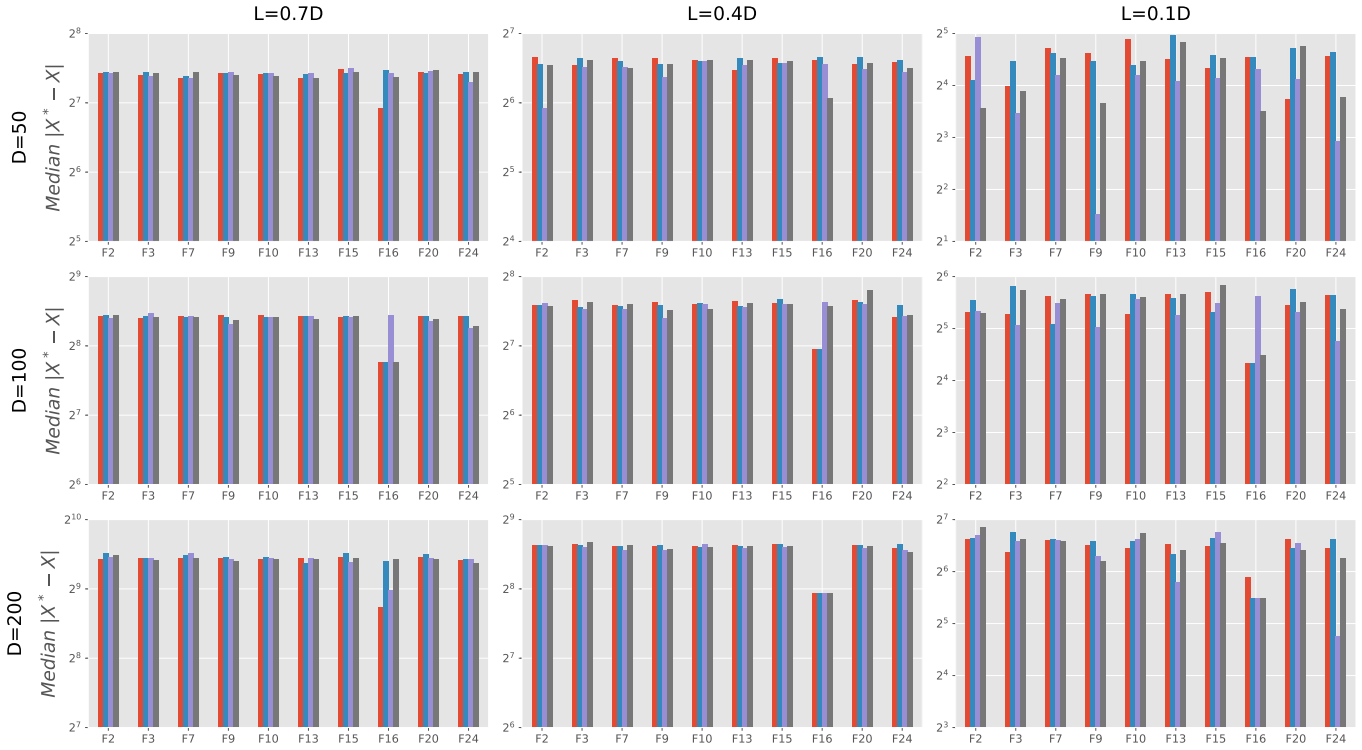

Fig. 9. Median absolute difference of the global optimum with the proposed optimum for all test cases based on Polynomial surrogates (with degree=2 and elastic-net penalty) is presented. The test cases are defined on combinations of ten test functions, four dimensionality reduction techniques and three distinct values for D and L each.

Next, we report the results concerning the criterion of the global optimality. For this, we share the median absolute difference of the globally optimal function values with the proposed optimal function values for all 720 cases in figures 6 and 7 based on the explanation provided in section IV-E. Both figures 6 and 7 contain a total of nine subplots each based on three distinct values of D and L respectively. Each subplot contains ten bar charts corresponding to the ten test functions discussed in section IV-A. Furthermore, each bar chart shares the median absolute difference (lower is better) of the globally optimal function values with the proposed optimal function values. From figure 6 which depicts these results for Kriging; it can be observed that PCA, KPCA and VAEs perform similarly in most cases, whereas AEs perform poorly. The performance of the AEs especially deteriorates for $D = 100$ and $L = 0.1D$. In the majority of the remaining cases, PCA performs better than the other DRTs. In the case of Polynomials in figure 7, all four DRTs perform likewise in most cases except for $L = 0.1D$, where we observe AEs perform better than the rest in 15/30 cases.

Similarly, plots depicting the median absolute difference (lower is better) of the global optima with the proposed optima for all 720 cases are provided in figures 8 and 9. Note that in figures 8 and 9, only the upper portions of the bar plots have been plotted to distinguish between the LDSMs more clearly. From figure 8, we can observe that AEs and VAEs perform poorly, whereas PCA performs better than the other DRTs in most cases. In figure 9, the LDSMs perform similarly in most cases, except when $L = 0.1D$.

VI. Conclusion and Outlook

In this paper, we empirically evaluate and compare four of the most important DRTs for efficiently constructing the LDSMs. The DRTs discussed in the paper are PCA, KPCA, AEs and VAEs respectively. The comparison is made on the basis of the quality assessment of the corresponding LDSMs on a diverse range of test cases. There are a total of 720 test cases based on the combinations of ten test functions, four DRTs, two surrogate modeling techniques and three distinct values for D and L each. Furthermore, the quality assessment of the LDSMs is based on two criteria: modeling accuracy and global optimality. Based on the observations in section V, we believe following conclusions can be drawn:

1) The LDSMs based on AEs have the highest modeling accuracy in 132/720 cases. In most of the remaining cases, the modeling accuracy of the LDSMs is comparable. This demonstrates the efficacy of all four DRTs and provides evidence to suggest AEs as the most competitive DRT in terms of modeling accuracy. However, note that future research is necessary to validate this on more complex cases, e.g., real-world applications and optimization under uncertainty.
2) In terms of the global optimality, the LDSMs based on PCA perform better than the others in most cases for Kriging. This aspect can be verified from figures 6 and 8. For LDSMs based on Polynomials (with degree=2 and elastic-net penalty), the performance on this criterion is comparable in most cases.

Although this study provides a comprehensive analysis on the performance of the LDSMs based on the four DRTs discussed in the paper, there are a few limitations to discuss. Firstly, the study focuses on unconstrained, noiseless, single-objective optimization problems. Therefore, the results can not be generalized to more complex cases. Secondly, the study does not focus on the size of the training sample size which can be crucial in many cases. It is pertinent to maintain that we found this to be infeasible for this particular study and left it for future work. Based on these rationales, we believe that further work is necessary to validate these findings on more complex cases, e.g., multiple-objectives, optimization under uncertainty, constraint handling and real-world applications.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 766186 (ECOLE). Hao Wang acknowledges the support from the Paris Île-de-France Region.

References

[1] D. Beasley, D. R. Bull, and R. R. Martin, "A sequential niche technique for multimodal function optimization," *Evolutionary computation*, vol. 1, no. 2, pp. 101–125, 1993.
[2] S. Shan and G. G. Wang, "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions," *Structural and multidisciplinary optimization*, vol. 41, no. 2, pp. 219–241, 2010.
[3] H.-G. Beyer and B. Sendhoff, "Robust optimization–a comprehensive survey," *Computer methods in applied mechanics and engineering*, vol. 196, no. 33-34, pp. 3190–3218, 2007.
[4] S. Ullah, H. Wang, S. Menzel, B. Sendhoff, and T. Bäck, "An empirical comparison of meta-modeling techniques for robust design optimization," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 819–828, IEEE, 2019.
[5] G. Venter, "Review of optimization techniques," *Encyclopedia of aerospace engineering*, 2010.
[6] R. M. Lewis, V. Torczon, and M. W. Trosset, "Direct search methods: then and now," *Journal of computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 191–207, 2000.
[7] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary computation 1: Basic algorithms and operators*. CRC press, 2018.
[8] A. Forrester, A. Sobester, and A. Keane, *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.
[9] A. Sóbester, A. I. Forrester, D. J. Toal, E. Tresidder, and S. Tucker, "Engineering design applications of surrogate-assisted optimization techniques," *Optimization and Engineering*, vol. 15, no. 1, pp. 243–265, 2014.
[10] J. Stork, M. Friese, M. Zaefferer, T. Bartz-Beielstein, A. Fischbach, B. Breiderhoff, B. Naujoks, and T. Tušar, "Open issues in surrogate-assisted optimization," in *High-Performance Simulation-Based Optimization*, pp. 225–244, Springer, 2020.
[11] P. Yang, K. Tang, and X. Yao, "Turning high-dimensional optimization into computationally expensive optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 143–156, 2017.
[12] H. Wang, "Forward regression for ultra-high dimensional variable screening," *Journal of the American Statistical Association*, vol. 104, no. 488, pp. 1512–1524, 2009.
[13] T. Robinson, M. Eldred, K. Willcox, and R. Haimes, "Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping," *Aiaa Journal*, vol. 46, no. 11, pp. 2814–2822, 2008.

[14] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.

[15] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[16] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[17] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[18] S. T. Roweis, "Em algorithms for pca and spca," in *Advances in neural information processing systems*, pp. 626–632, 1998.

[19] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural computation*, vol. 11, no. 2, pp. 443–482, 1999.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[21] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Advances in neural information processing systems*, pp. 3–10, 1994.

[22] M. Ranzato, C. Poultney, S. Chopra, and Y. L. Cun, "Efficient learning of sparse representations with an energy-based model," in *Advances in neural information processing systems*, pp. 1137–1144, 2007.

[23] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models," in *Advances in neural information processing systems*, pp. 899–907, 2013.

[24] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," 2011.

[25] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[26] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of the 31st International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 1278–1286, 2014.

[27] D. P. Kingma, M. Welling, *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[28] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[29] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[30] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, pp. 585–591, 2002.

[31] P. Demartines and J. Hérault, "Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets," *IEEE Transactions on neural networks*, vol. 8, no. 1, pp. 148–154, 1997.

[32] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[33] T. Rios, B. Sendhoff, S. Menzel, T. Bäck, and B. van Stein, "On the efficiency of a point cloud autoencoder as a geometric representation for shape optimization," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 791–798, IEEE, 2019.

[34] S. Ullah, Z. Xu, H. Wang, S. Menzel, B. Sendhoff, and T. Bäck, "Exploring clinical time series forecasting with meta-features in variational recurrent models," in *2020 IEEE International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2020.

[35] C. O. S. Sorzano, J. Vargas, and A. P. Montano, "A survey of dimensionality reduction techniques," *arXiv preprint arXiv:1403.2877*, 2014.

[36] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative," *J Mach Learn Res*, vol. 10, no. 66-71, p. 13, 2009.

[37] W.-L. Loh *et al.*, "On latin hypercube sampling," *The annals of statistics*, vol. 24, no. 5, pp. 2058–2080, 1996.

[38] K. Vanaja and R. Shobha Rani, "Design of experiments: concept and applications of plackett burman design," *Clinical research and regulatory affairs*, vol. 24, no. 1, pp. 1–23, 2007.

[39] S. C. Ferreira, R. Bruns, H. Ferreira, G. Matos, J. David, G. Brandao, E. P. da Silva, L. Portugal, P. Dos Reis, A. Souza, *et al.*, "Box-behnken design: an alternative for the optimization of analytical methods," *Analytica chimica acta*, vol. 597, no. 2, pp. 179–186, 2007.

[40] N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockhoff, "COCO: A platform for comparing continuous optimizers in a black-box setting," *CoRR*, vol. abs/1603.08785, 2016.

[41] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the royal statistical society: series B (statistical methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[42] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, (USA), pp. 2546–2554, Curran Associates Inc., 2011.

[43] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *ICML*, 2013.

[44] J. L. Morales and J. Nocedal, "Remark on "algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization"," *ACM Transactions on Mathematical Software (TOMS)*, vol. 38, no. 1, pp. 1–4, 2011.