Neural Networks as Black-Box Benchmark Functions Optimized for Exploratory Landscape Features

Raphael Patrick Prager[*]
University of Münster
Münster, NRW, Germany
raphael.prager@wi.uni-muenster.de

Konstantin Dietrich[*]
TU Dresden & Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI)
Dresden, Saxony, Germany
konstantin.dietrich@tu-dresden.de

Lennart Schneider
LMU Munich & Munich Center for Machine Learning (MCML)
Munich, Bavaria, Germany
lennart.schneider@stat.uni-muenchen.de

Lennart Schäpermeier
TU Dresden & Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI)
Dresden, Saxony, Germany
lennart.schaepermeier@tu-dresden.de

Bernd Bischl
LMU Munich & Munich Center for Machine Learning (MCML)
Munich, Bavaria, Germany
bernd.bischl@stat.uni-muenchen.de

Pascal Kerschke
TU Dresden & Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI)
Dresden, Saxony, Germany
pascal.kerschke@tu-dresden.de

Heike Trautmann
University of Münster
Münster, NRW, Germany
University of Twente
Enschede, Overijssel, Netherlands
trautmann@wi.uni-muenster.de

Olaf Mersmann
TH Köln
Gummersbach, NRW, Germany
olaf.mersmann@th-koeln.de

ABSTRACT

Artificial benchmark functions are commonly used in optimization research because of their ability to rapidly evaluate potential solutions, making them a preferred substitute for real-world problems. However, these benchmark functions have faced criticism for their limited resemblance to real-world problems. In response, recent research has focused on automatically generating new benchmark functions for areas where established test suites are inadequate. These approaches have limitations, such as the difficulty of generating new benchmark functions that exhibit exploratory landscape analysis (ELA) features beyond those of existing benchmarks.

The objective of this work is to develop a method for generating benchmark functions for single-objective continuous optimization with user-specified structural properties. Specifically, we aim to demonstrate a proof of concept for a method that uses an ELA feature vector to specify these properties in advance. To achieve this, we begin by generating a random sample of decision space variables and objective values. We then adjust the objective values using CMA-ES until the corresponding features of our new problem match the predefined ELA features within a specified threshold. By iteratively transforming the landscape in this way, we ensure that the resulting function exhibits the desired properties. To create the final function, we use the resulting point cloud as training data for a simple neural network that produces a function exhibiting the target ELA features. We demonstrate the effectiveness of this approach by replicating the existing functions of the well-known BBOB suite and creating new functions with ELA feature values that are not present in BBOB.

CCS CONCEPTS

• **Theory of computation** → **Design and analysis of algorithms**; **Continuous optimization**;

KEYWORDS

Exploratory Landscape Analysis, Benchmarking, Instance Generator, Black-Box Continuous Optimization, Neural Networks

ACM Reference Format:
Raphael Patrick Prager, Konstantin Dietrich, Lennart Schneider, Lennart Schäpermeier, Bernd Bischl, Pascal Kerschke, Heike Trautmann, and Olaf Mersmann. 2023. Neural Networks as Black-Box Benchmark Functions Optimized for Exploratory Landscape Features. In *Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA '23), August 30-September 1, 2023, Potsdam, Germany*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3594805.3607136

[*]Both authors contributed equally to this research.

1 INTRODUCTION

The development of optimization algorithms creates a natural need for test problems to assess their search behaviour, robustness and overall performance. The choice of test problems is far from trivial and will determine the course of development. Ideally, we would like to guide this process by confronting the algorithm with challenging and representative problems that resemble problems we encounter in real-world settings. An impeding factor is that those real-world problems are often very expensive to evaluate and subject to companies' proprietary information. Standardized artificial benchmark suites, like the Black-Box Optimization Benchmark (BBOB) suite [6], the annually changing CEC benchmark (e.g. [37]), and to some degree Nevergrad [31], strive to alleviate this issue by offering a wide range of different problem instances. These suites contain a set of hand-picked artificial problems that are chosen to cover a range of different problem properties such as a varying degree of multi-modality, presence of global and funnel structures, and so forth. While this approach is not without merit, it begs the question whether these suites are sufficiently diverse and resemble prominent real-world problems as a whole. Research endeavours such as [21] and [34] show that this might not always be the case. This in turn may bias the development and evaluation of algorithms where they achieve competitive results on artificial benchmarks but are effectively never used in practice. As algorithm selection and configuration become increasingly automated [17, 18, 29], there is a rising incentive to further increase the heterogeneity of solvers from which a model can choose from. Given these pitfalls concerning artificial benchmarks in general, it is not surprising that several approaches have been developed to address these issues [3, 4, 19, 25].

One of the earliest works to build a tune-able landscape generator was presented by Gallagher and Yuan [4]. The generator uses a set of Gaussian functions and a small number of parameters that can be linked to a problem's geometric properties. This allowed them to achieve new insights on the behavior of some estimation of distribution algorithms, indicating the usefulness and need for a feature driven problem generator.

Another approach has been developed by Lang and Engelbrecht [19]. In their work, they construct a novel benchmark suite by systematically evaluating existing benchmark functions originating from a multitude of different suites. They used exploratory landscape analysis (ELA) [24] to test how diverse each of these functions really is w.r.t. a subset of ELA features. Using self-organized feature maps, they construct a space spanned by the ELA features. They aim to maximize the coverage of this space by selecting the most appropriate functions out of their pool of benchmark functions for each cell map. The only drawback of their approach is that they are not proposing a mechanism to generate entirely new functions.

Muñoz and Smith-Miles [25] presented another approach. In a similar fashion to [19], they use ELA features to evaluate the diversity of a single benchmark (BBOB). This allowed them to characterize every function using an eight-dimensional feature vector. By projecting the vectors into a two-dimensional instance space, they could identify uncovered areas and thereby identify target ELA feature vectors. New functions that exhibit the respective target ELA vector properties are then generated making use of genetic programming. They achieve good results in being able to interpolate within and even extrapolate beyond the convex hull that the BBOB suite spans in the instance space.

Nevertheless, as recently pointed out by Dietrich and Mersmann [3], there are some downsides to the results of Muñoz and Smith-Miles [25]. For one, there is a lack of knowledge about the global optima of genetically programmed functions. But the high computational cost of genetic programming weighs more severely. While Dietrich and Mersmann [3] were able to get rid of both these downsides by using affine recombinations of the BBOB functions as new benchmark problems, this approach could only interpolate within the convex hull of the ELA feature space within BBOB.

The main contribution of this paper is that we address this shortcoming by proposing a new, neural-network based, method for generating novel problem instances with an arbitrary property combination w.r.t. the chosen ELA features. At the same time, we retain the auspicious aspects of [25] and [3]. In other words, our devised approach is not only able to interpolate but also to extrapolate beyond the problem space and thereby can potentially generate truly novel problem instances in less time.

This paper can be compartmentalized into two distinct sets of experiments. The first experiments focus on validating and demonstrating that our method works reasonably well in principle. We accomplish this by trying to emulate certain benchmark functions for which we sample and compute a so-called target ELA feature vector. We start by creating a random sample in the decision space and random objective values corresponding to each observation in our sample. We optimize these objective values until they exhibit the desired ELA values which are determined by the target ELA feature vector. The resulting point cloud contains our anchor points to generate a new benchmark problem. To construct the new problem we then make use of a simple neural network which is trained on this optimized point cloud. In order to show that the resulting functions mimic the existing ones well, we compare the behavior of optimization algorithms on both sets. This gives us the opportunity to investigate where our devised approach excels and where it encounters issues by comparing the emulated landscape with the original one as well as the algorithm rankings between these two. The second set of experiments highlights the potential of our devised approach to create entirely novel functions which are not represented in our selected benchmark suite.

The remainder of this paper is structured as follows. In Section 2, we give a general overview of ELA as well as a justification for the selected features and some technical details for their calculation. Section 3 provides a full account of our devised approach where the construction of the aforementioned point cloud is subject of Subsection 3.1, and the surrogate models are discussed in Subsection 3.2. In Section 4, we validate our approach by imitating functions from the BBOB suite, discussing the results from the landscape perspective in Subsection 4.1 and from an algorithm performance perspective in Subsection 4.2. Our general workflow is then evaluated by generating functions for ELA feature vectors which are not part of the chosen benchmark suite in Subsection 4.3. Finally, we conclude our paper in Section 5 and provide an outlook on future research opportunities based on our findings.

2 EXPLORATORY LANDSCAPE ANALYSIS

While the hardness and properties of an optimization problem, given enough samples, may be characterized visually for up to two-dimensional problems, visualizations of higher-dimensional problems are generally infeasible. Thus, other mechanisms are required to identify problem properties, such as the degree of multi-modality or the presence of global and funnel structures [16, 23, 24]. These properties ultimately define the hardness of a problem. The aforementioned mechanisms manifest themselves in a set of numerical features, which, in the single-objective continuous optimization domain, are consolidated under the term exploratory landscape analysis (ELA) [24].

In essence, ELA is a collection of heterogeneous methods to extract quantitative information about a black-box optimization problem. The majority of ELA features can be computed based on a fixed sample of randomly generated points from the search space, along with their respective (evaluated) objective values. Different sampling procedures have been investigated over the years [32]. The sampling size, on the other hand, often depends on the scope of the analysis. In automated algorithm selection [14], the sampling size must be competitive and, therefore, small, whereas theoretical undertakings can be more lavish (e.g., [3, 11, 33]). Regardless of the scope, the sampling size is typically scaled with the dimensionality d of the problem instance. In this work, we use a sampling size of $250 \cdot d$, which balances between a sufficient coverage of the search space together with keeping the generation of new problem instances computationally less intensive. This size is also recommended by [33] to correctly classify all BBOB problems. For our work, we choose Latin hypercube sampling as the sampling strategy. Furthermore, for any given sample we normalize the objective values to the range of $[0, 1]$ via their respective sample minimum and maximum. This has two desirable effects. Firstly, it recently has been shown that not all ELA features are invariant to linear transformations of the objective space [30]. The proposed method to deal with this issue is to normalize the objective values. Secondly, a lower and upper bound of zero and one alleviates certain problems for our neural networks which act as surrogate models in the consecutive section. This allows us for instance to utilize a sigmoid activation function in the output layer which naturally maps all values in the interval $[0, 1]$.

Up to this point, we only discussed details pertaining to the ELA sample. But ultimately, problem hardness is subject to the properties of the fitness landscape making the selection of suitable ELA features another crucial aspect of this work. We adapt a semi-structured approach. Meaning, we utilize the findings of [33] and [35] and iteratively remove and add certain hand-picked features to improve the landscape properties of our generated functions. The final chosen ELA features are listed below and computed with the Python package `pflacco`*:

- **`ela_meta.lin_simple.adj_r2`** Adjusted coefficient of determination of the linear regression model without variable interactions [24].
- **`ela_meta.lin_w_interact.adj_r2`** Adjusted coefficient of determination of the linear regression model with variable interactions [24].
- **`ela_meta.quad_simple.adj_r2`** Adjusted coefficient of determination of the quadratic regression model without variable interactions [24].
- **`ela_meta.quad_w_interact.adj_r2`** Adjusted coefficient of determination of the quadratic regression model with variable interactions [24].
- **`ela_distr.skewness`** Skewness of the sample's objective values [24].
- **`nbc.nb_fitness.cor`** The correlation between the fitness values of the search points and their indegree in the nearest-better point graph [15].
- **`nbc.nn_nb.sd_ratio`** Ratio of the standard deviation of all nearest neighbor distances to the standard deviation of all nearest better distances [15].
- **`fitness_distance.fitness_std`** Standard deviation of the sample's objective values [12].

3 BLACK-BOX FUNCTION GENERATION

The generation of novel problem instances consists of three steps. Conceptually, we want to (1) **identify a target vector of ELA features**, which occupies sparse regions of an existing benchmark to enhance its diversity. Meaning, this target vector should be constituted of vastly different ELA feature values compared to the values of ELA features of any given benchmark suite. Note that the exact identification of this target vector is not the focus of this work. However, for five novel ELA vectors we demonstrate this generation procedure is able to generate functions with landscape structures arguably different to BBOB. In addition, we compute these target ELA feature vectors on the problem instances of BBOB. This gives us the opportunity to discern how successful our developed approach in general is by comparing our generated function to the existing BBOB instance which has served as a target. Given this target ELA vector, we (2) **generate a sample of points** (i.e., a point cloud) through an optimization process that exhibits the desired ELA vector values. Once the similarity of the values is satisfactory, we (3) **build a surrogate model** based on the found point cloud.

The entire process is depicted in Figure 1. In the following, we will discuss the second and third components in more detail.

3.1 Optimization of Point Clouds

Given a target ELA feature vector $\mathbf{a}$ (which is constituted of the aforementioned 8 features) that we want to emulate, we first create a random sample $X \in \mathbb{R}^{n \times d}$ in the decision space and generate its objective values $\mathbf{y} \in \mathbb{R}^n$ randomly in the interval $[0, 1]$. We refer to the tuple $(X, \mathbf{y})$ as the *point cloud*. For this current iteration of the point cloud, we then compute the corresponding ELA feature vector $\mathbf{b} = h(X, \mathbf{y})$, where h is the set of functions required to calculate the ELA features of interest based on the point cloud $(X, \mathbf{y})$. Within our optimization procedure, we then *only* adjust the objective values $\mathbf{y}$ such that the distance between the ELA feature vectors $\mathbf{b}$ and $\mathbf{a}$ is minimal. Hence, we can formally define the function $f : \mathbb{R}^n \to \mathbb{R}$ to generate an adequate point cloud, where we strive to find a $\mathbf{y}^* \in \arg\min_{\mathbf{y}} f(\mathbf{y})$ such that the distance between the two ELA feature vectors $\mathbf{a}$ and $\mathbf{b}$ is minimal:

$$f(\mathbf{y}) = ||h(X, \mathbf{y}) - \mathbf{a}||_2 \tag{1}$$

*https://github.com/reiyan/pflacco

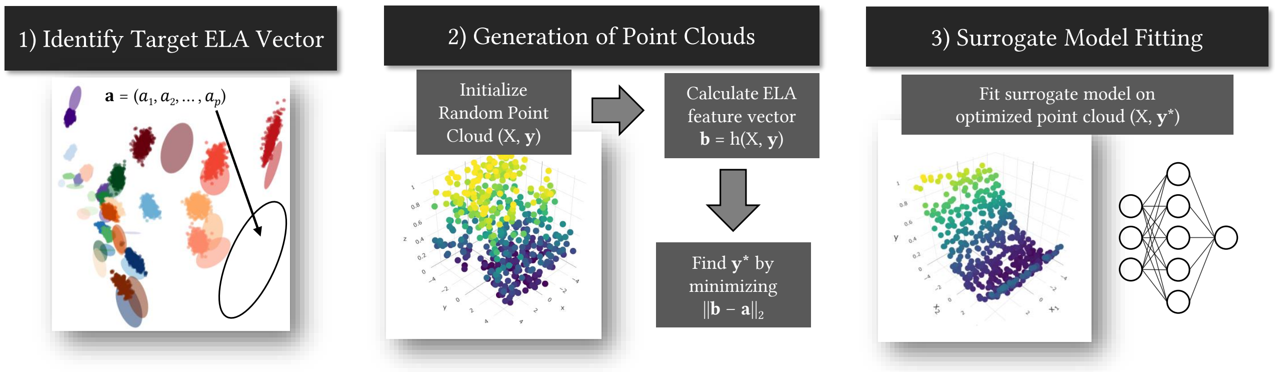

1) Identify Target ELA Vector
$\mathbf{a} = (a_1, a_2, \ldots, a_p)$
2) Generation of Point Clouds
Initialize Random Point Cloud (X, **y**)
Calculate ELA feature vector **b** = h(X, **y**)
Find **y*** by minimizing ||**b** − **a**||$_2$
3) Surrogate Model Fitting
Fit surrogate model on optimized point cloud (X, **y***)

Figure 1: High-level process description of our approach to generate arbitrary black-box problems. In 1), an ELA target vector is identified in sparse regions of an existing benchmark suite. In 2), we generate a random point cloud, calculate the respective ELA features and minimize between the former and the latter by changing y. In 3), we use the optimized point cloud as the basis for fitting our surrogate model.

The global optimum in such a case is known to be zero since distances between two objects w.l.o.g. are in the interval $[0, \infty)$. The dimensionality of this problem is governed by the sample size n of the point cloud. In our experiments, we use a sample size of $250 \cdot d$, which scales linearly with the dimensionality d of the problem we want to create. Meaning if we want to create a two-dimensional problem instance, the dimensionality of our minimization problem is $250 \cdot 2 = 500$. It is apparent that these problems can become extremely high-dimensional. Here, the work of [38] provides valuable insight into this matter. The authors show that the conventional CMA-ES [7] still achieves competitive results even for higher-dimensional problems. While our problem dimension exceeds their limit of up to 320, we argue that their findings are still relevant for and can be extrapolated to our case, i.e., we use the CMA-ES to solve the optimization problem given in Equation 1. Each attempt to generate a point cloud is allocated a budget of $200 \cdot n$, where n is simultaneously the dimensionality of our optimization problem of Equation 1 and by extension the size of our point cloud. The required CPU time is dependent (1) on the time it requires to calculate $f(\mathbf{y})$ (i.e., ELA feature calculation) for each candidate solution and (2) on the time necessary to update the covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$. Generating a single $2d$ point cloud takes up to 4 CPU hours whereas a single $3d$ point cloud demands almost 3 CPU days.

Until now, as hinted at in Section 2, we only normalize the objective values of any given sample by applying min-max normalization. While ELA features value ranges become less prone to having unusually small or large values, it does not account for the different scales *between* distinct ELA features. This biases the search trajectory of the CMA-ES and places more importance on certain ELA features without justifiable reason. Therefore, we experimentally determined the minima and maxima of each ELA feature where no theoretical lower or upper bound can be determined (e.g, the upper bound of any R^2 value is 1) and use these values to apply min-max normalization to our ELA features during the optimization process.

3.2 Surrogate Model Fitting

To construct a novel function based on a given optimized point cloud $(X, \mathbf{y}^*)$, we use neural networks (NNs) as surrogate model $m : \mathbb{R}^{n \times d} \longrightarrow \mathbb{R}, m(\mathbf{x}) = \hat{y}$ fitted on $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{*(i)}\}_{i=1,\ldots,n}$, where $\mathbf{x}^{(i)}$ is the i-th row of X and $y^{*(i)}$ the i-th element of $\mathbf{y}^*$. We use a simple feed forward architecture consisting of one hidden layer with 512 units and a tanh activation function. As a final output we use a sigmoid activation function (due to objective values being normalized via min-max normalization and we want the predictions of the model m to be on the same scale).

On the one hand, our choice of surrogate model is motivated based on theoretical properties, as feed forward NNs are known to be universal function approximators, i.e., already a feed forward NN with a single hidden layer can approximate any continuous function to any desired accuracy, given arbitrary width (number of hidden units) [10]. On the other hand, NNs have desirable practical properties:

(i) They scale well with the number of data points (which will become relevant for higher dimensions).
(ii) They are easy to deploy and integrate in benchmarking suites.
(iii) They induce very little latency overhead during evaluation, i.e., a forward pass.

All in all, this makes NNs very attractive as surrogate models in our scenario compared to other regression models such as generalized additive models, splines, support vector machines, tree-based models or Gaussian Processes. Still, one has to be aware that NNs have a strong inductive bias to learn smooth functions (see, e.g., [5]) – especially when using tanh and sigmoid activation functions – which can only be influenced to a certain degree by architectural design choices.

Our explicit training procedure looks like the following: As our goal is to perfectly interpolate $\mathcal{D}$, we perform gradient descent using the mean squared error as a loss function and AdamW as optimizer with default parameters and a learning rate of 0.001

for 5000 epochs. The objective here is to perfectly interpolate the training data and essentially overfit drastically. Only under these circumstances we can guarantee that the resulting model m will exhibit the same (or reasonably similar) target ELA feature vector $\mathbf{a}$ when evaluated at the anchor points X, i.e., $||h(X, \hat{\mathbf{y}}) - \mathbf{a}||_2$ is sufficiently small[†]. Note that we do not employ mini-batches but use a single batch of all available data - again due to our goal to heavily overfit the training data. We implement our NNs in PyTorch [27]. We provide the required code to replicate our experiments on GitHub[‡].

4 RESULTS

4.1 Landscape Perspective

To validate our devised methodology, we use ELA feature vectors generated based on the functions of the Black-Box Optimization Benchmark (BBOB) suite [6] as targets. The BBOB suite consists of 24 different functions, conveying distinct challenges to numeric optimization algorithms. Each function (FID) belongs to one of five so-called function groups, which share the same major characteristics, and is instantiated using a set of transformations such as rotations and shifts, yielding different function instances with fundamentally the same properties. In this study, we consider only the first instance of each FID in the dimensions $d = \{2, 3\}$, where we denote a problem instance as $p_i := (\text{FID}, d)$. This amounts to 48 distinct problem instances we endeavour to recreate. For each p_i, we run the Python implementation of CMA-ES called pycma with a budget of $200n$ where n is the dimensionality of the point cloud optimization problem and not the dimension d of the function we are trying to generate. We perform ten replications of which all are able to minimize Equation 1 similarly. For every problem p_i, we then select the point cloud that achieved the best objective value to provide our neural networks with the best possible point cloud to fit. All the results presented below, therefore, refer to the fits of the best point clouds.

Exemplary point clouds, where the approximation of the target ELA feature vector works sufficiently well and a counterexample, are given in Figure 2. The left-hand side depicts the case for BBOB function 3. Subfigure 2(a) depicts a random sample of the original BBOB function 3 whereas the adjacent subplot visualizes our point cloud after optimization. We can discern that the two landscapes are rotated, yet in essence they structurally remain the same. The right-hand side showcases an approximation attempt which did not terminate satisfactorily. While some aspects of the original BBOB function (FID 14) are present in our point cloud, the general landscape is far too quadratic and noisy. We believe that the observed phenomenon is attributable to a low resolution of the point cloud.

A full account for every problem instance p_i is given in Figure 3. As pointed out in Section 3.1, each ELA feature is normalized to fall into the interval $[0, 1]$. Hence, we can determine the maximum distance between any two ELA feature vectors $\in \mathbb{R}^8$ is $\sqrt{8}$. This helps to contextualize the reported results because these will be in the interval of $[0, \sqrt{8}]$. The vast majority of cells is colored white which indicates the CMA-ES was able minimize the distance between the two ELA feature vectors up to a precision of 10^{-7}. These results are only tarnished by the functions 6, 9, 14 and 15 for $d = 2$, and for $d = 3$ this list is extended by functions 4, 12, 17 and 20. However, we cannot observe any statistical association which lets us divine the cause of these unsatisfactory instances. Yet, we have to put these values into context, meaning despite their perceptible coloring their values pale in comparison only and not when viewed in isolation. We deem them still sufficient to be used as training data for our surrogate model.

To judge the accuracy of these point clouds further, we create 100 samples for each problem instance p_i (using Latin hypercube sampling) and compute the respective ELA feature vectors. These are aggregated into a single vector $\overline{\mathbf{a}}_i$ via the arithmetic mean. While we make only use of a single dedicated NN architecture for all 48 problem instances, we train a NN individually for a single problem and each p_i receives its own NN model. In general, the training procedure in each individual attempt introduces a stochastic component. Hence, we trained five NNs for each problem instance where every NN is identical except their initial weights. Surprisingly, all five NNs interpolate between samples provided by the point cloud in similar fashion. Ultimately, this means that the resulting fitness landscape of our newly generated function is not subject to the training procedure of our NNs, i.e., the initialization of weights and biases. Similar to creating our target ELA vectors, we generate 100 ELA feature vectors $\mathbf{b}_i^{(j)}$ for each of the five NNs, therefore $j \in \{1, 2, \ldots 500\}$.

We report our results using the same metric of Equation 1. Meaning, the Euclidean distance between the target ELA vector $\overline{\mathbf{a}}_i$ of p_i and the 500 ELA feature vectors $\mathbf{b}_i^{(j)}$ of our constructed function. The distribution of Euclidean distances for each p_i can be found in Figure 4.

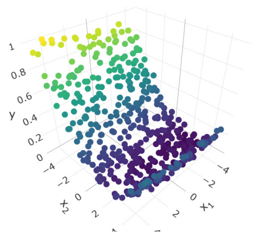
A cursory assessment already reveals that our devised approach works very well for the majority of the 48 considered problem instances. Nevertheless, the results for $d = 2$ are about an order of magnitude better than the results for $d = 3$. The cause for this relatively poor performance can partially be traced back to ill-suited point clouds found by CMA-ES. This is the case for functions 4, 6, 9, 12, 14, 17, and 20. Overall, we anticipate that premature termination of network training due to insufficient number of epochs is another contributing factor to this observation.

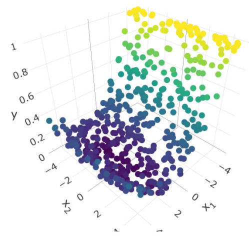
In the following, we provide a more in-depth view into the fitness landscape of a few selected functions. These are functions in which we excel but also functions which did not work out as well. In addition, we present exemplary uni-modal and multi-modal problem instances. These functions are visualized in Figure 5. We contrast the contour plot of the original BBOB problem instance to its corresponding approximation of our approach. This is accompanied by a parallel coordinate plot of the ELA features where the target ELA features are given in black and the values of our five surrogate models are colored.

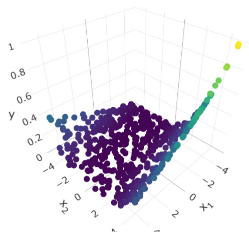
The first row shows the Sphere function (FID 1), which is arguably one of the easiest optimization problems there is. This is also the case for the approximation of its landscape. The only notable difference between the two is that our function is not as artificially smooth as the BBOB function.

[†]Here, $\hat{\mathbf{y}}$ is the vector of predictions of the surrogate model consisting of predictions $\hat{y} = m(\mathbf{x})$for each point $\mathbf{x} \in X$.

[‡]https://github.com/Reiyan/ela_nn_function_generation


(a) BBOB FID 3


(b) Generated Point Cloud for FID 3


(c) BBOB FID 14

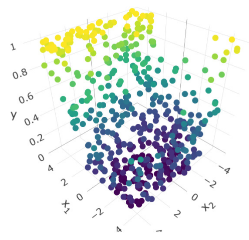

(d) Generated Point Cloud for FID 14. x_1- and x_2-axis have been rotated for accessibility.

Figure 2: Generated point clouds optimized by CMA-ES in contrast to a sample generated from the respective BBOB function directly. Objective values of the respective functions have been scaled to $[0, 1]$ via min-max normalization.

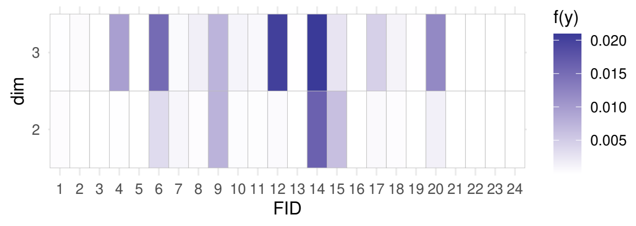


Figure 3: Heatmap of 48 distances in terms of $f(\mathbf{y})$ between a target ELA feature vector and the ELA feature vector calculated on the point cloud found by CMA-ES. The theoretical boundaries of these objective values are in the interval $[0, \sqrt{8}]$.

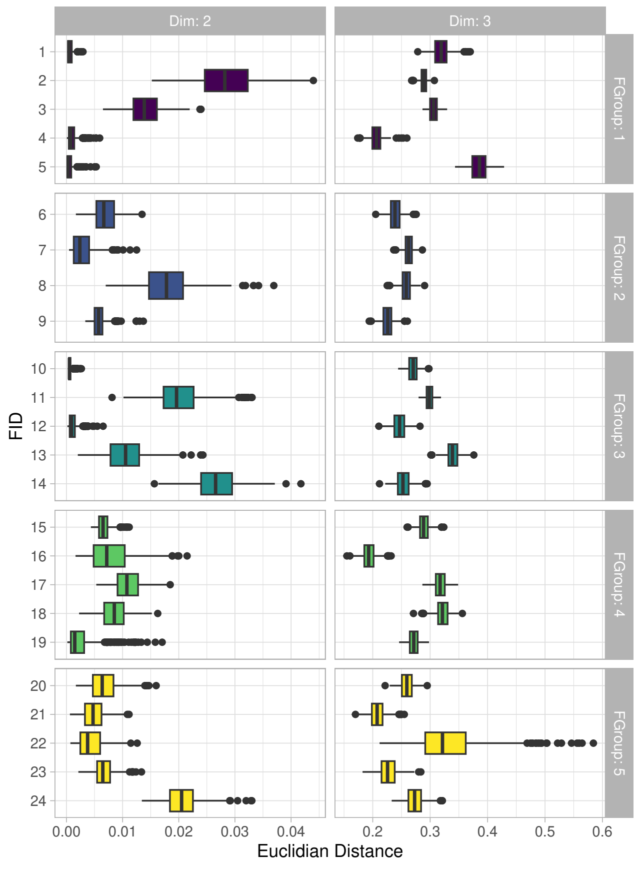


Figure 4: Boxplots of the Euclidean distances between the target ELA vector $\bar{\mathbf{a}}_i$ and each associated problem instance $\mathbf{b}_i^{(j)}$.

BBOB Function 2 displays a similar case. In terms of the captured landscape properties, we are able to emulate the separability of this problem well and also the quadratic structure as a whole. In addition, the generated landscape is rotated but we argue that this fundamentally no issue, since this does not change the characteristics of this problem.

Another function we deem worthy to report individual results on, is the rotated Rosenbrock function (FID 9). This function requires a solver to follow a long path to the global optimum while also accounting for changes in the search direction. Our approximation of it mimics this to a certain degree where we can observe a convergence into the corners of the search space. Yet, we cannot model the strength of this turn as strongly as it is present in the BBOB function.

We also want to critically discuss the inadequacies of our developed approach. This pertains to FID 14. This problem instance exhibits the worst performance in terms of our similarity measure s_i and also in terms of $f(\mathbf{y})$ when optimizing the point cloud. In this case, we can notice that overfitting our NNs leads to a more rugged landscape with drastically increasing and decreasing objective values of the landscape. This is, on the other hand, a property we require to model BBOB functions which are multi-modal and exhibit a mediocre to high conditioning of their respective landscape as will be shown in the following.

Up until now, we only discussed the uni-modal functions with varying degrees of separability and conditioning. The last subplot of Figure 5 depicts the BBOB Function 21. This function is highly multi-modal with no global structure. The basin sizes of attraction

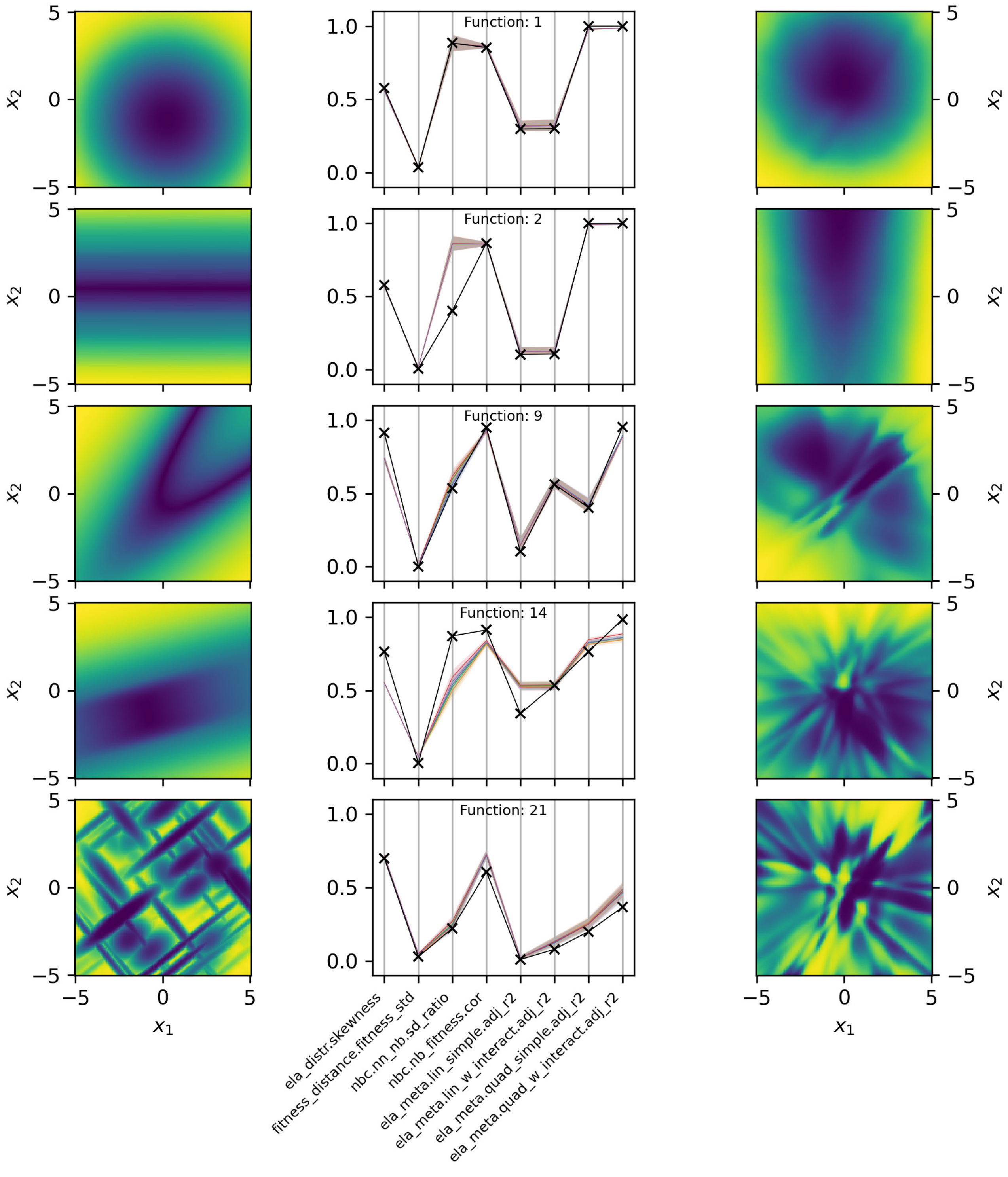


Figure 5: The left column shows contour plots of five selected BBOB functions. The affiliated eight dimensional target ELA vectors are represented by the black lines in the parallel plots of the middle column. The five colored lines show the mean ELA feature vector of the five trained neural networks. The matching colored area around the lines covers the tenth to ninetieth percentile of ELA feature values achieved during sampling. The right column represents the contour plots of one of the trained neural networks.

are are aligned in random directions with different convex shapes. Our approximation is able to capture these properties in general. A meaningful difference is that our basins of attraction more often run parallel to the axes of the search space in contrast to the original BBOB function.

4.2 Algorithm Perspective

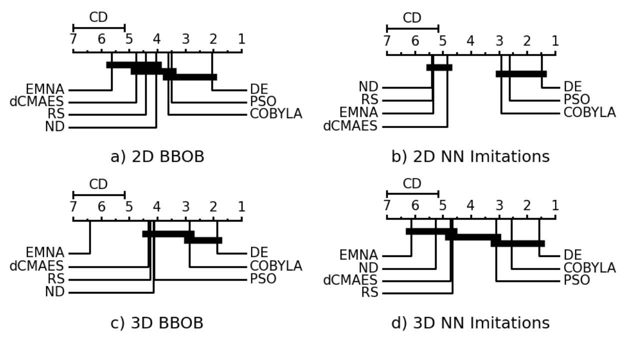


Figure 6: Performance rankings based on the performance of each optimizer with budget $10\,000 \cdot d$ for each dimension separately between BBOB and our imitated versions. Ties are identified by a non-parametric Nemenyi test and indicated by the horizontal lines. Lower rank is better.

It is imprudent to solely rely on the analysis of landscape properties to investigate the similarities between our imitated functions and their original counterparts. A different analysis avenue is provided by observing the behaviour of algorithms on each of these two sets and whether they exhibit a similar performance on the original as well as on the imitation. Hence, we conducted a comprehensive benchmark on BBOB making use of the Nevergrad framework [31]. Nevergrad provides a wide range of gradient-free optimization algorithms but what makes it most useful for our scenario is its interface for implementing and testing custom benchmark functions. As solvers we selected the following seven complementary algorithms:

- estimation of multivariate normal algorithm (EMNA) [20],
- Nelder-Mead algorithm (ND) [26],
- differential evolution (DE) [36],
- particle swarm optimization (PSO) [13],
- diagonal covariance matrix adaptation evolution strategy (dCMA-ES) [8],
- constrained optimization by linear approximation (Cobyla) [28],
- and random search (RS) [1].

Each of these algorithms is is given a budget of $10\,000 \cdot d$ function evaluations and is applied 10 times on every single problem in the dimension two and three.

Figure 6 shows critical distance (CD) plots [2] for the BBOB and NN problems. We compute rankings based on the performance of each optimizer (median of 10 replications) on each of the BBOB and NN problems. A non-parametric Friedman test with Nemenyi post-hoc test is performed to identify ties in the optimizer rankings, which are indicated by the horizontal lines. We rely on the implementation in the Python package `autorank` [9].

In Figure 6, we observe a comparable behavior in the algorithms examined. Specifically, the top three performing algorithms are DE, PSO, and COBYLA. In the $2d$ scenario, there is no statistically significant difference between these three. However, there are slight variations in the ranking of the remaining four algorithms. In our $2d$ imitations of BBOB, these variances are indistinguishable, whereas in the original BBOB suite, other algorithms sometimes tie with the three best performers in terms of their overall performance indicated by the horizontal line. Overall, our imitations of BBOB divide the seven algorithms more strongly into two groups. The $3d$ case provides a more convoluted case. Here, the best performing algorithms on BBOB are DE and COBYLA (which are statistically tied) whereas on our imitation suite this is extended by PSO. These permutations in rankings are to be expected and we think that overall also from an algorithmic perspective these two sets of functions are similar.

4.3 Generation of Novel Benchmark Functions

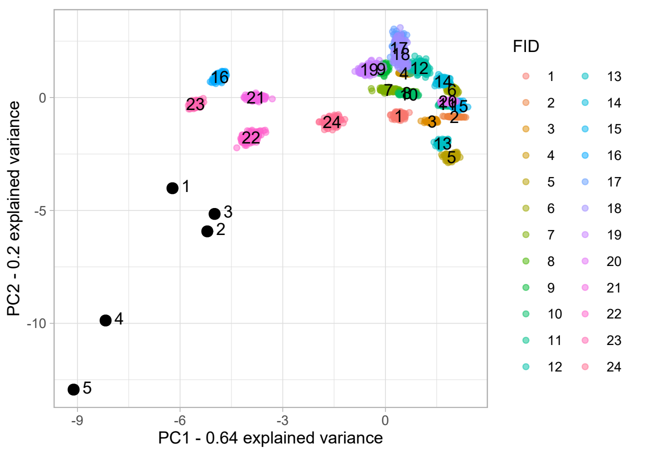


Figure 7: Projection of the ELA feature vectors of BBOB and newly generated functions (represented by a black dot) into a Cartesian plane via PCA.

While the imitation of existing benchmark functions allows us to dissect our approach in isolation and to identify possible shortcomings, we ultimately propose this method to create problem instances which are not covered by an existing benchmark suite. At first glance the creation of an ELA feature vector which is not represented by any problem instance of a given benchmark suite seems simple. This, however, is not the case since interactions between ELA features exist. For example, it is not possible to create an ELA feature vector which has R^2 of 1 for a linear model and simultaneously for a quadratic model (assuming non-zero model coefficients). Thereby, the identification of feasible ELA feature vectors - which are also different to ELA feature vectors of a benchmark - poses a separate research question which is not addressed in this work.

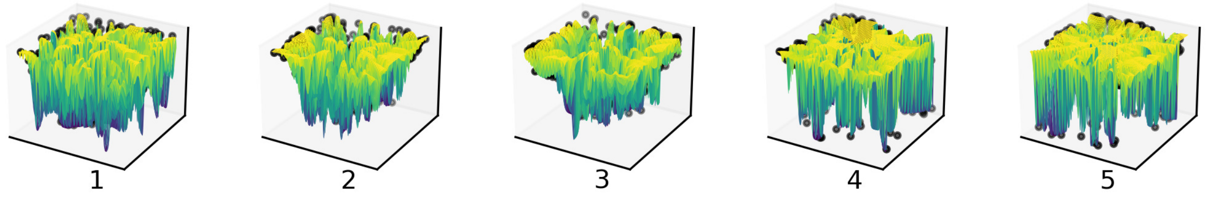


Figure 8: Landscape of the 5 newly created $2d$ Functions where the black points represent the point cloud optimized by the CMA-ES

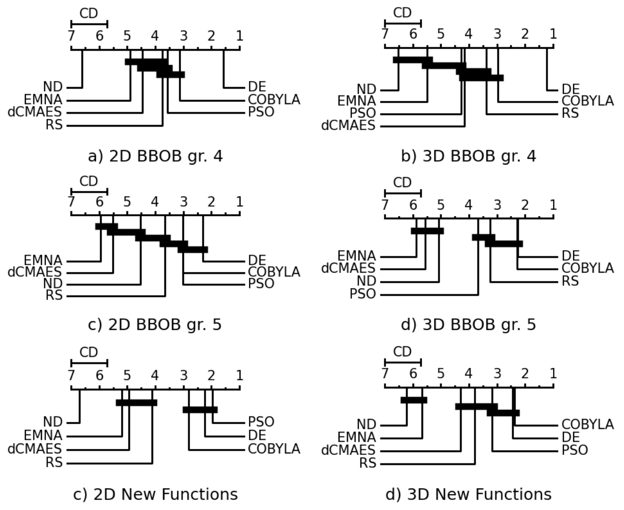


Figure 9: Performance rankings based on the performance of each optimizer with budget $10\,000 \cdot d$ for each dimension separately between BBOB function groups 4 and 5 in contrast to the 5 newly created functions. Ties are identified by a non-parametric Nemenyi test and indicated by the horizontal lines.

Rather, we handpicked five exemplary and feasible ELA feature vectors which have a considerable distance to the existing ELA feature vectors of BBOB to showcase the potential of our method. A projection of these five exemplary ELA feature vectors into a two dimensional plane (using PCA) is depicted in Figure 7.

The generation of the respective point clouds as well as the training of our NN surrogate models is identical to our previous undertaking. The optimization process to generate the point clouds terminates with objective values ranging from $2.37e{-}06$ to $1.26e{-}02$ and an average of $3.01e-03$ for $2d$ and $3d$ problems. We deem this sufficiently well approximated to build our surrogate models upon. Furthermore, we apply the same algorithm benchmark procedure outlined in Subsection 4.2.

Landscape Perspective. The landscape of the newly generated $2d$ functions is depicted in Figure 8. All five functions are highly multi-modal without any or very weak apparent global structure. The conditioning of these functions is also extremely high, i.e., small changes in the decision space generally lead to drastic changes in the objective space. The first three landscapes tend to have the majority of their local and global optima located more in the center of the search space, whereas the latter two follow no observable pattern and have even highly competitive local optima right at the vertices of the box-constraints. Especially the fourth and fifth problem share commonalities with the Katsuura function (FID 23) of BBOB. However, our functions do not exhibit the same regular pattern for local optima which presumably makes it more difficult. This irregular pattern is uncommon for BBOB in general.

Algorithm Perspective. These aforementioned landscape properties make the newly generated functions most similar to the BBOB function group 4 or 5. Hence, we present the algorithm rankings in form of CD-plots for these groups specifically as the CD plots for the remaining function groups do not share any resemblance at all with the CD plots of the newly generated functions.

When assessing Figure 9, there seems to be no conclusive evidence, that our newly generated problems are substantially different in terms of algorithm rankings. ND performs the worst on the new generated functions which is also the case for BBOB group 4. The top performing algorithms are mainly PSO, DE, COBYLA and occasionally even RS. While we would have appreciated a more indicative finding from the algorithm perspective, we are satisfied with the structural differences in the constructed landscapes compared to BBOB.

5 CONCLUSION

In this work, we propose a novel approach for creating new benchmark problems for single-objective continuous black-box optimization. Our approach is mainly based on creating a so-called point cloud which exhibits the desired ELA features and thereby the desired fitness landscape properties. This point cloud is used to train a neural network which acts as a surrogate model and newly created benchmark function. Our approach solves the previously unaddressed issues of [3] by providing the capability to not only interpolate in the ELA feature space of benchmark functions but also extrapolate beyond the convex hull of this region. Furthermore, NNs are fully differentiable in theory (not only with respect to the weights and biases, but also with respect to the input variables) and we hypothesize that we can determine all optima comparably efficiently. This, however, needs to be fully explored. We evaluated our approach on a set of BBOB functions where the goal was to recreate them as closely as possible. Our approach generates problem instances of identical nature. This is supported by a systematic benchmark study of different optimizers which exhibit an overall similar performance ranking on the original BBOB functions and their respective imitations.

In addition, we showcase that it is possible to create functions based on ELA feature vectors which are not represented by any BBOB function. The resulting functions offer interesting irregular structures and landscape property combinations which are not present in the current BBOB suite.

Yet, our devised approach is only the first foray into relatively uncharted territory. An inundating number of research opportunities still remains such as, e.g., improving the scalability to create functions of higher dimensionality. This can be done by employing more CPU-efficient algorithms such as the LM-CMA-ES [22]. Furthermore, a systematic evaluation of different sampling sizes and strategies may conclude that a smaller sample size still produces sufficient results. This would effectively reduce the dimensionality of our point cloud optimization and presumably the complexity of this optimization task.

While we evaluated a multitude of different NN architectures, we still believe that there is potential for improvement in this area. Testing different architectures or investigating the viability of smaller point clouds is only a small excerpt of possible refinements. We could also employ our approach to model real-world problems where a small sample of data exist. While this amount of data may be not sufficient enough to build a surrogate model on it alone, we can use it to calculate ELA features. Thereafter, our procedure can be applied to generate a surrogate function comparatively cheaply. While solving this surrogate function does not lead to the identification of the global optimum of the real-world problem, it can serve as a proxy to benchmark and design algorithms more tailored to real-world problems.

ACKNOWLEDGMENTS

Lennart Schneider is supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the Center for Analytics - Data - Applications (ADACenter) within the framework of BAYERN DIGITAL II (20-3410-2-9-8).

REFERENCES

[1] R. L. Anderson. 1953. Recent Advances in Finding Best Operating Conditions. *J. Amer. Statist. Assoc.* 48, 264 (1953), 789–798. https://doi.org/10.1080/01621459.1953.10501200

[2] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research* 7 (2006), 1–30.

[3] Konstantin Dietrich and Olaf Mersmann. 2022. Increasing the Diversity of Benchmark Function Sets Through Affine Recombination. In *Parallel Problem Solving from Nature – PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part I* (Dortmund, Germany). Springer-Verlag, Berlin, Heidelberg, 590–602. https://doi.org/10.1007/978-3-031-14714-2_41

[4] M. Gallagher and Bo Yuan. 2006. A general-purpose tunable landscape generator. *IEEE Transactions on Evolutionary Computation* 10 (2006), 590–603. Issue 5. https://doi.org/10.1109/TEVC.2005.863

[5] L. Grinsztajn, E. Oyallon, and G. Varoquaux. 2022. Why Do Tree-Based Models Still Outperform Deep Learning on Typical Tabular Data?. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[6] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions*. Research Report RR-6829. INRIA. https://hal.inria.fr/inria-00362633

[7] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. 2003. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evol. Comput.* 11, 1 (mar 2003), 1–18. https://doi.org/10.1162/106365603321828970

[8] N. Hansen and A. Ostermeier. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation* 9 (6 2001), 159–195. Issue 2. https://doi.org/10.1162/106365601750190398

[9] Steffen Herbold. 2020. Autorank: A Python package for automated ranking of classifiers. *Journal of Open Source Software* 5, 48 (2020), 2173. https://doi.org/10.21105/joss.02173

[10] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks* 2, 5 (1989), 359–366.

[11] Anja Janković and Carola Doerr. 2019. Adaptive Landscape Analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 2032–2035. https://doi.org/10.1145/3319619.3326905

[12] Terry Jones and Stephanie Forrest. 1995. Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 184–192.

[13] J. Kennedy and R. Eberhart. 1995. Particle swarm optimization. 4 (1995), 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

[14] Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. 2019. Automated Algorithm Selection: Survey and Perspectives. *Evolutionary Computation* 27, 1 (2019), 3 – 45. https://doi.org/10.1162/evco_a_00242

[15] Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. 2015. Detecting Funnel Structures by Means of Exploratory Landscape Analysis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (Madrid, Spain) *(GECCO '15)*. Association for Computing Machinery, New York, NY, USA, 265–272. https://doi.org/10.1145/2739480.2754642

[16] Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. 2016. Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016* (Denver, Colorado, USA) *(GECCO '16)*. Association for Computing Machinery, New York, NY, USA, 229–236. https://doi.org/10.1145/2908812.2908845

[17] Pascal Kerschke and Heike Trautmann. 2019. Automated Algorithm Selection on Continuous Black-Box Problems By Combining Exploratory Landscape Analysis and Machine Learning. *Evolutionary Computation* 27, 1 (2019), 99 – 127. https://doi.org/10.1162/evco_a_00236

[18] Ana Kostovska, Diederick Vermetten, Sašo Džeroski, Carola Doerr, Peter Korosec, and Tome Eftimov. 2022. The Importance of Landscape Features for Performance Prediction of Modular CMA-ES Variants. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (Boston, MA, USA), Jonathan E. Fieldsend and Markus Wagner (Eds.). ACM, 648 – 656. https://doi.org/10.1145/3512290.3528832

[19] Ryan Dieter Lang and Andries Petrus Engelbrecht. 2021. An Exploratory Landscape Analysis-Based Benchmark Suite. *Algorithms* 14, 3 (2021), 78.

[20] Pedro Larrañaga and Jose A Lazano. 2002. *Estimation of Distribution Algorithms*. Vol. 2. Springer Science & Buisness Media. https://doi.org/10.1007/978-1-4615-1539-5

[21] Fu Xing Long, Bas van Stein, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. 2022. Learning the Characteristics of Engineering Optimization Problems with Applications in Automotive Crash *(GECCO '22)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3512290.3528712

[22] Ilya Loshchilov. 2014. A Computationally Efficient Limited Memory CMA-ES for Large Scale Optimization. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* (Vancouver, BC, Canada) *(GECCO '14)*. Association for Computing Machinery, New York, NY, USA, 397–404. https://doi.org/10.1145/2576768.2598294

[23] Katherine Mary Malan and Andries Petrus Engelbrecht. 2009. Quantifying Ruggedness of Continuous Landscapes Using Entropy. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1440 – 1447. https://doi.org/10.1109/CEC.2009.4983112

[24] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory Landscape Analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (Dublin, Ireland) *(GECCO '11)*. Association for Computing Machinery, New York, NY, USA, 829–836. https://doi.org/10.1145/2001576.2001690

[25] Mario Andrés Muñoz and Kate Smith-Miles. 2020. Generating New Space-Filling Test Instances for Continuous Black-Box Optimization. *Evolutionary Computation* 28 (2020), 379–404. Issue 3. https://doi.org/10.1162/EVCO_A_00262

[26] J. A. Nelder and R. Mead. 1965. A Simplex Method for Function Minimization. *Comput. J.* 7 (1 1965), 308–313. Issue 4. https://doi.org/10.1093/COMJNL/7.4.308

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan

Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[28] M. J. D. Powell. 1994. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation.* Springer Netherlands, Dordrecht, 51–67. https://doi.org/10.1007/978-94-015-8330-5_4

[29] Raphael Patrick Prager, Moritz Vinzent Seiler, Heike Trautmann, and Pascal Kerschke. 2022. Automated Algorithm Selection in Single-Objective Continuous Optimization: A Comparative Study of Deep Learning and Landscape Analysis Methods. In *Proceedings of the 17th International Conference on Parallel Problem Solving from Nature (PPSN XVII)* (Dortmund, Germany), Günter Rudolph, Anna V. Kononova, Hernán E. Aguirre, Pascal Kerschke, Gabriela Ochoa, and Tea Tusar (Eds.). Springer, 3 – 17. https://doi.org/10.1007/978-3-031-14714-2_1

[30] Raphael Patrick Prager and Heike Trautmann. 2023. Nullifying the Inherent Bias of Non-Invariant Exploratory Landscape Analysis Features. In *Applications of Evolutionary Computation*, João Correia, Stephen Smith, and Raneem Qaddoura (Eds.). Springer International Publishing, Cham.

[31] J. Rapin and O. Teytaud. 2018. Nevergrad - A Gradient-Free Optimization Platform. https://GitHub.com/FacebookResearch/Nevergrad.

[32] Quentin Renau, Carola Doerr, Johann Dreo, and Benjamin Doerr. 2020. Exploratory Landscape Analysis is Strongly Sensitive to the Sampling Strategy. *Proceedings of the 16th International Conference on Parallel Problem Solving from Nature (PPSN 2020)* 12270 LNCS (2020), 139–153. https://doi.org/10.1007/978-3-030-58115-2_10

[33] Quentin Renau, Johann Dreo, Carola Doerr, and Benjamin Doerr. 2021. Towards Explainable Exploratory Landscape Analysis: Extreme Feature Selection for Classifying BBOB Functions. *Proceedings of the 24th International Conference, EvoApplications 2021* 12694 LNCS (2021), 17–33. https://doi.org/10.48550/arxiv.2102.00736

[34] Lennart Schneider, Lennart Schäpermeier, Raphael Patrick Prager, Bernd Bischl, Heike Trautmann, and Pascal Kerschke. 2022. HPO × ELA: Investigating Hyperparameter Optimization Landscapes by Means of Exploratory Landscape Analysis. In *Parallel Problem Solving from Nature – PPSN XVII*, Günter Rudolph, Anna V. Kononova, Hernán Aguirre, Pascal Kerschke, Gabriela Ochoa, and Tea Tušar (Eds.). Springer International Publishing, Cham, 575–589.

[35] Moritz Vinzent Seiler, Raphael Patrick Prager, Pascal Kerschke, and Heike Trautmann. 2022. A Collection of Deep Learning-Based Feature-Free Approaches for Characterizing Single-Objective Continuous Fitness Landscapes. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Boston, Massachusetts) *(GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 657–665. https://doi.org/10.1145/3512290.3528834

[36] Rainer Storn and Kenneth Price. 1997. Differential Evolution –A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization* 11, 4 (1997), 341–359. https://doi.org/10.1023/A:1008202821328

[37] Ke Tang, Xın Yáo, Ponnuthurai Nagaratnam Suganthan, Cara MacNish, Ying-Ping Chen, Chih-Ming Chen, and Zhenyu Yang. 2007. Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization. *Nature inspired computation and applications laboratory, USTC, China* 24 (2007), 1–18.

[38] Konstantinos Varelas. 2019. Benchmarking Large Scale Variants of CMA-ES and L-BFGS-B on the Bbob-Largescale Testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 1937–1945. https://doi.org/10.1145/3319619.3326893