NEURAL EXPLORATORY LANDSCAPE ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent research in Meta-Black-Box Optimization (MetaBBO) have shown that meta-trained neural networks can effectively guide the design of black-box optimizers, significantly reducing the need for expert tuning and delivering robust performance across complex problem distributions. Despite their success, a paradox remains: MetaBBO still rely on human-crafted Exploratory Landscape Analysis features to inform the meta-level agent about the low-level optimization progress. To address the gap, this paper proposes Neural Exploratory Landscape Analysis (NeurELA), a novel framework that dynamically profiles landscape features through a two-stage, attention-based neural network, executed in an entirely end-to-end fashion. NeurELA is pre-trained over a variety of MetaBBO algorithms using a multi-task neuroevolution strategy. Extensive experiments show that NeurELA achieves consistently superior performance when integrated into different and even unseen MetaBBO tasks and can be efficiently fine-tuned for further performance boost. This advancement marks a pivotal step in making MetaBBO algorithms more autonomous and broadly applicable. The source code of NeurELA can be accessed at `https://anonymous.4open.science/r/Neur-ELA-303C`.

1 INTRODUCTION

Black-Box Optimization (BBO) is a discipline in mathematical optimization characterized by the absence of both the closed-form expression of the target objective function and its derivative during the optimization process. Traditional BBO methods, such as Genetic Algorithm (Holland, 1975), Evolutionary Strategy (Hansen & Ostermeier, 2001a), and Bayesian Optimization (Shahriari et al., 2015), have been extensively studied in the past half century. They are applied to address a broad range of black-box scenarios, including automated machine learning (Akiba et al., 2019), bioinformatics (Tsaban et al., 2022), prompt tuning in large language models (Guo et al., 2024b). However, these methods often require manual tuning with deep expertise to ensure the optimal performance on unseen tasks. Recent advances in Meta-Black-Box Optimization (MetaBBO) (Xue et al., 2022; Lange et al., 2023; Chen et al., 2024; Guo et al., 2024a; Lange et al., 2024) leverage meta-learning (Finn et al., 2017) to streamline the traditional labour-intensive tuning process across different problem distributions. This effort not only reduces the need for expert-level knowledge, but also enhances generalization towards unseen problems. As depicted on the left side of Figure 1 MetaBBO generally employs a bi-level optimization paradigm (Ma et al., 2023). In the meta level, a control policy (often implemented as a neural network) is maintained to dictate timely algorithm configuration for the low-level BBO optimizer according to the low-level optimization status. Once obtained the algorithm configuration, the low-level BBO optimizer optimizes the target BBO problem by that configuration and returns a feedback to the meta-level policy the performance gain feedback. The meta-objective of MetaBBO is to meta-learn an optimal control policy which cound achieve maximum accumulated performance gain across a problem distribution.

One of the core challenges to ensure the generalization performance of MetaBBO algorithms is to design an information-rich profiling system for the low-level optimization status (highlighted in yellow box on the left side of the Figure 1). Firstly, the optimization status must include the identification features to differentiate various optimization problems. This design is crucial to enable the meta-level neural policy to be aware of *what is the target optimization problem*. Secondly, the optimization status should align with the low-level optimization dynamics, which helps to inform the meta-level neural policy about *where the low-level optimization progress has reached*. Existing MetaBBO algorithms widely adopt Exploratory Landscape Analysis (ELA) (Wright et al., 1932;

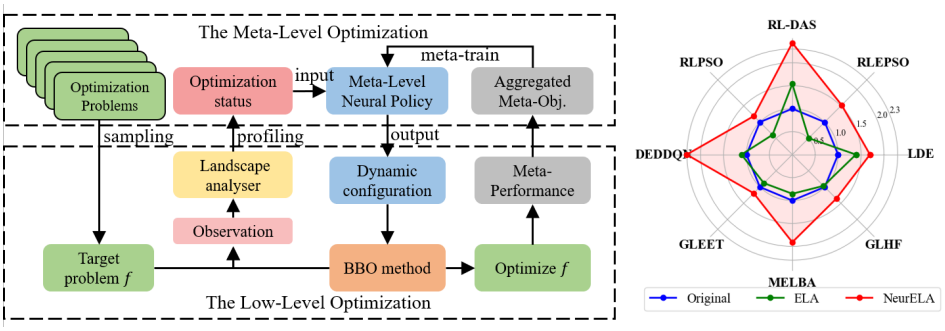

Figure 1: **Left**: The general workflow of MetaBBO algorithms, which follows a bi-level optimization paradigm. The landscape analyser timely profiles the low-level optimization progress, informing the meta-level neural policy to dynamically output desired configuration for the low-level BBO method. **Right**: The average optimization performance (larger is better) of integrating our pre-trained landscape analyser into diverse MetaBBO tasks (red line). The blue line denotes the original performance of the MetaBBO algorithms, while the green line denotes the performance of integrating the traditional exploratory landscape analysis features into the MetaBBO algorithms.

Mersmann et al., 2011) to profile the low-level optimization status. While the approaches provide a comprehensive understanding of problem and algorithm characteristics, they still face several limitations. 1) Many features derived from ELA are highly correlated with each other (Renau et al., 2019) and exhibit strong sensitivity to random sampling strategies (Renau et al., 2020). This necessitates an additional step of feature selection for the given MetaBBO scenario, requiring a certain level of expert knowledge. 2) The landscape features given by ELA are more like an "offline" portray of the target optimization problem. This is somewhat incompatible with the dynamic nature of MetaBBO paradigm, which requires both the "offline" problem properties and the "online" optimization progress. 3) Computing some of these features, particularly those related to convexity and local optimum properties, consume additional function evaluations (FEs). The consumption of the computational resource, initially reserved for the low-level optimization, ultimately degrades the final performance of the MetaBBO algorithms.

To address the above limitations, we propose Neural Exploratory Landscape Analysis (NeurELA), a novel, learnable framework designed to dynamically profile optimization status for the MetaBBO algorithms in an end-to-end manner. Our NeurELA encounters certain challenges: on the one hand, we need to carefully design a universal neural network-based landscape analyser that is compatible with a variety of MetaBBO algorithms. On the other hand, it is crucial to enable effective learning within this analyser to ensure it robustly generalizes not only to unseen optimization problems but also to previously unencountered MetaBBO algorithms. We address the challenges as follows.

1) Our research begins by designing a comprehensive operating space, Ω, within which our landscape analyser Λ functions. This space encompasses a collection of MetaBBO tasks, each designated as $\mathbb{T} := \{\mathbb{A}, \mathbb{D}\}$. Here, $\mathbb{A}$ specifies a particular MetaBBO algorithm, and $\mathbb{D}$ represents the associated set of optimization problems to meta-train and test the algorithm. The primary objective is to optimize Λ such that it enhances the performance of various $\mathbb{A}$ across their respective problem sets in Ω. Our introduction of Λ within the comprehensive operating space Ω marks a significant advancement in MetaBBO methodologies. Unlike traditional approaches that treat each optimization algorithm and problem in isolation, Λ leverages a holistic view, enhancing performance across a diverse set of tasks.

2) We instantiate the landscape analyser Λ through a neural network model parameterized by θ. Specifically, Λ_θ is implemented using a two-stage attention-based neural network. Our design enables flexible integration with most existing MetaBBO algorithms as an effective replacement for their traditional hand-crafted landscape analysis mechanisms. Given the non-differential nature of our objective in BBO, we adopt an Evolution Strategy (Hansen & Ostermeier, 2001b; Li et al., 2018) to train Λ_θ. This training is carried out within a multitask neuroevolution (Miikkulainen et al., 2024) paradigm, aiming to optimize θ to maximize the expected performance across MetaBBO tasks.

3) We propose using the trained Λ_θ in two distinct ways. First, having been trained on a diverse range of tasks, Λ_θ is capable of being directly applied to new, unseen MetaBBO algorithms, achieving zero-

shot generalization. Second, Λ_θ also supports fine-tuning in conjunction with the meta-training of new MetaBBO algorithms, allowing it to adapt its performance for specific optimization challenges.

Through extensive experiments, we validate the effectiveness of NeurELA, as illustrated on the right side of Figure 1. These results demonstrate that it is possible to pre-train a universal neural landscape analyser Λ_θ over a group of MetaBBO tasks and then generalize it to unseen ones. Our work pioneers the integration of automatic feature extraction in MetaBBO, transforming these methods into fully end-to-end systems. This advancement significantly streamlines optimization processes, enhancing efficiency and adaptability across related fields, and reducing the need for manual expertise.

2 Related Works

Meta-Black-Box Optimization. Existing MetaBBO paradigms can be classified into following categories with respect to role they play at the low-level optimization process: 1) *Hyper-parameters configurator*: this line of works meta-trained neural network-based policy to dynamically configure the hyper-parameters of the backend BBO optimizer, e.g., the scale factor (F) and crossover rate (Cr) in Differential Evolution (DE) algorithm (Sun et al., 2021; Tan et al., 2022). Initial researches (e.g., RLPSO (Wu & Wang, 2022)) employed simple MLP structures to parameterize the meta-level policy, which was enhanced by the latest works such as GLEET (Ma et al., 2024), which facilitated Transformer (Vaswani et al., 2017) architecture to boost the sequential decision making along the low-level optimization. 2) *Update rule selector*: works in this line flexibly select desired update rules for the low-level optimization process. Several initial works learns to select one mutation operator from a pre-defined DE mutation operator pool to attain diverse optimization behaviours (Sharma et al., 2019; Tan & Li, 2021; Tan et al., 2022). Then the research scope extends to the refinement of the operators pool for optimizing more complex problems (Lian et al., 2024), and dynamically selecting the entire algorithm(Guo et al., 2024a). 3) *Complete optimizer generator*: works in this line aims to meta-learning the neural network-based policy as a complete optimizer. Lange et al. (Lange et al., 2023; 2024) proposed meta-training several attention blocks as the complete updating process of CMAES (Hansen et al., 2003) , which was followed by Song et al. (Song et al., 2024), proposing meta-training Transformer-based language model as an entire BBO method. Such end-to-end deep learning systems encountered interpretability issue. To this end, SYMBOL (Chen et al., 2024) proposed generating symbolic update rules for the low-level optimization process, promoting the interpretable and automatic discovery of black-box method. GLHF (Li et al., 2024) carefully design a Transformer-based neural network that imitates the evolutionary operators in DE algorithm. As for the designs of optimization status extraction, a great majority of existing MetaBBO algorithms borrow the idea from well-known landscape analysis researches (Wright et al., 1932; Mersmann et al., 2011). To make those landscape features compatible with the MetaBBO paradigm, existing MetaBBO algorithms more or less facilitated feature selection and feature aggregation procedure. Take a recent work GLEET (Ma et al., 2024) as an example, it aggregates the representative ELA features (Mersmann et al., 2011) into a 9-dimensional high-level profiling features, which helps reduce the training complexity and enhance the overall optimization performance. However, we have to note that such hand-crafted feature processing requires deep expertise in optimization.

Landscape Analysis. Landscape Analysis originates from the emergence of Automated Algorithm Selection (AAS) (Rice, 1976; Kerschke et al., 2019), which selects an optimal algorithm out of a collection of algorithms for any given optimization problem. Generally, Landscape analysis is formulated as a function: $\Lambda : (X, y) \to s$, that maps the sampled candidates X and their corresponding objective values y to a feature vector s, each dimension of which indicates some problem-specific properties of the target optimization problem. The feature vector s can be subsequently fed into machine learning algorithms (e.g., Support Vector Machines (Cortes & Vapnik, 1995) and Random Forests (Breiman, 2001)) to address some specific algorithmic tasks (Kerschke & Trautmann, 2019a). Over the years, a wide range of Landscape Analysis feature groups have been developed, and structurally summarized in (Wright et al., 1932; Pitzer & Affenzeller, 2012; Malan & Engelbrecht, 2013; Kerschke & Trautmann, 2019b), of which the mostly utilized ones are: 1) *Fitness Distance Correlation* (Jones et al., 1995) includes six features profiling the distances of the sampled candidates both in decision space and the objective space, as well as the correlation in-between the two spaces. 2) *Dispersion* (Lunacek & Whitley, 2006) features describe the degree of dispersity for the best quantiles of the sampled candidates. 3) *Classical ELA* (Mersmann et al., 2011) proposed using six groups of low-level metrics such as local search, skewness of objective space and approximated

curvature in (second) first-order as a comprehensive abstract of basic landscape properties (e.g., global structure, multimodality, separability). Generally, classic ELA requires adequate samples and enormous computational efforts to conclude all of its fifty low-level feature values. 4) *Information Content* (Muñoz et al., 2014) comprises five features about the smoothness, ruggedness and neutrality of an optimization problem. It first applies random walk sampling (e.g., Metropolis Random Walk (Hastings, 1970)) for each of the sampled candidates, then calculates its features by statistic metrics. 5) *Nearest Better Clustering* (Kerschke et al., 2015) is a collection of features that describe the distribution of the nearest better neighbors and the nearest neighbors of the sampled candidates.

While these hand-crafted landscape analysis features provide a systematic understanding of optimization problems, they inevitably hold high computing complexity hence consume additional computational resources. Besides, the intricate correlations in-between these feature groups (Renau et al., 2019) and their sensitivities to the random sampling strategy (Renau et al., 2020) necessitate the expert-level feature selection and feature aggregation procedures. To address these issues, several early-stage studies have recently proposed learning neural networks (e.g., convolutional neural network (Prager et al., 2021a) and point cloud transformer (Seiler et al., 2022a)) as the feature mapping function. Prager et al. (2021b) first proposed using Deep-CNN to extract landscape features from fitness map of moderate initial sample points. The neural network is trained to correctly predict algorithm configuration of modular CMA-ES algorithms. However, solely feeding the fitness map might introduce information loss. To address such issue, Seiler et al. (2022b) propose a novel point cloud transformer (PCT) based approach, which embeds sample points information and the fitness map together as the input. This work generally achieves superior landscape feature prediction accuracy. Another technical bottleneck of (Prager et al., 2021b) is the supervised training, which might leads the learned landscape representation overfit to a certain AAS/AAC task. To refine this issue, van Stein et al. (2023) propose self-supervised learning through AutoEncoder, which adopted fitness map as the target to be reconstructed. More recently, Seiler et al. (2024) propose DeepELA to adrress both the infomation loss and the supervised trainng issues. It introduces MHA and trains it by a contrastive learning loss InfoNCE to ensure aligned representation across different sampling strategies. In this paper, our NeurELA presents several novelty compared with the above learning-based ELA frameworks: a) NeurELA aims for MetaBBO tasks yet the previous works generally aim for classical AS or AC tasks. MetaBBO tasks are more akin to Dynamic AS or AC tasks (DAC, DAS), where the configuration decision have to be customized in each generation of optimization of a given problem. b) Unlike previous works, which provide static and invariant problem properties, NeurELA offers dynamic optimization status information. c) compared with (Seiler et al., 2024), our architecture designs do not rely on a pre-defined dimension upper bound, making it more generic and versatile for various MetaBBO tasks, see more detailed differences in Appendix B.2.

3 Neural Exploratory Landscape Analysis

3.1 Framework

We introduce a multi-task operating space, denoted as Ω, which serves as the foundational basis for our landscape analyser, Λ. The space Ω is mathematically defined as:

$$\Omega = \{\mathbb{T}_k = (\mathbb{A}_k, \mathbb{D}_k) \mid k = 1, 2, \ldots, K\} \tag{1}$$

where each $\mathbb{T}_k$ represents a task configuration consisting of a MetaBBO algorithm $\mathbb{A}_k$ and a corresponding set of optimization problems $\mathbb{D}_k$. This structured collection of tasks is meticulously designed to optimize our proposed landscape analyser Λ_θ over a variety of algorithms and problems.

In accordance with the MetaBBO protocol, each algorithm $\mathbb{A}_k$ is composed of three fundamental elements: the meta-level neural policy Π_ϕ, the low-level optimizer Γ, and the initial landscape feature analyser, denoted as Λ_0. As illustrated in Figure 1, the meta-training of $\mathbb{A}_k$ adheres to a bi-level optimization paradigm. Given a problem instance f sampled from $\mathbb{D}_k$, at each optimization step t, the low-level BBO method Γ interacts with f to attain a raw observation $\{X_i^t, y_i^t\}_{i=1}^m$ of the current population of candidates solutions, denoted as o^t, where m is the population size in population-based search methods Storn & Price (1997); Kennedy & Eberhart (1995); Holland (1975). Subsequently, the observation o^t is further processed by the original landscape analyser, Λ_0, to extract optimization status features $s^t = \Lambda_0(o^t)$. Based on these features, the meta-level neural policy Π_ϕ dictates a design choice c^t from the configuration space $\mathbb{C}$ conditioned on the current optimization status:

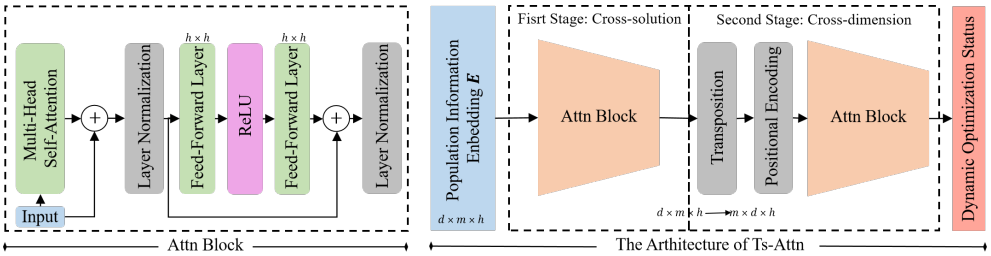

Figure 2: **Left**: The architecture of the basic attention block ($Attn$). **Right**: The computation graph of the two-stage attention mechanism (Ts-Attn).

$c^t = \Pi_\phi(s^t)$. Then, the low-level BBO method Γ loads c^t to optimize f for next time step $t+1$. The meta-performance r^t is calculated by evaluating the performance gain achieved by implementing c_t. The ultimate goal of the meta-training phase is to maximize the overall meta-objective F_{metaBBO}, defined as:

$$\max_{\phi} F_{\text{metaBBO}} = \mathbb{E}_{f\sim\mathbb{D}_k,\Pi_\phi}\left[\sum_{t=0}^{T} r^t\right], \tag{2}$$

where the expectation is over all problem instances from $\mathbb{D}_k$ and spans the total length T of the low-level optimization horizon.

In our NeurELA, we parameterize a neural network-based landscape analyser Λ_θ (where θ denote the learnable parameters), which substitutes the original landscape analyser Λ_0, to produce the optimization status features as $s^t = \Lambda_\theta(o^t)$. Our aim is to develop a universal Λ_θ for diverse MetaBBO algorithms that perform comparably to, or better than, the original hand-crafted Λ_0 in the corresponding MetaBBO tasks. To objectively measure the performance of Λ_θ, we define an evaluation metric $\Upsilon(\Lambda_\theta|\mathbb{T}_k)$ as the relative performance of Λ_θ against Λ_0 in the context of MetaBBO task $\mathbb{T}_k = (\mathbb{A}_k, \mathbb{D}_k)$. Concretely, we first meta-train two versions of the MetaBBO algorithms on $\mathbb{D}_k$: $\mathbb{A}_k(\Pi_\phi, \Gamma, \Lambda_0)$ and $\mathbb{A}'_k(\Pi_\phi, \Gamma, \Lambda_\theta)$, and then evaluate $\Upsilon(\Lambda_\theta|\mathbb{T}_k)$ as:

$$\Upsilon\left(\Lambda_\theta \mid \mathbb{T}_k\right) = \frac{1}{P\times Q}\sum_{p=1}^{P}\sum_{q=1}^{Q} Z_{p,q}, \text{ where } Z_{p,q} = -\frac{f^*_{p,q}-\mu_p}{\sigma_p} \tag{3}$$

where $P = |\mathbb{D}_{\text{test}}|$ is the number of test problems (the dataset $\mathbb{D}_k$ is divided into two distinct subsets: $\mathbb{D}_{\text{train}}$ for training and $\mathbb{D}_{\text{test}}$ for testing), $Q = 50$ represents the number of independent runs to obtain average performance, $f^*_{p,q}$ denotes the objective value achieved by testing $\mathbb{A}'_k$ on the p-th problem instance and q-th run, and μ_p and σ_p are the mean and standard deviation of the objective values obtained by testing $\mathbb{A}_k$ on the p-th problem over 50 independent runs. Note that we follow the Z-score normalization design in Ma et al. (2023) to evaluate the performance gain $Z_{p,q}$.

The meta-objective of our NeurELA framework is defined as maximizing the expected relative performance of the neural network-based landscape analyser, Λ_θ, across all MetaBBO tasks defined within the multi-task space Ω. Mathematically, this is expressed as:

$$\max_{\theta} F_{\text{metaELA}} = \mathbb{E}_{\mathbb{T}_k\sim\Omega}\left[\Upsilon\left(\Lambda_\theta \mid \mathbb{T}_k\right)\right] = \frac{1}{K}\sum_{k=1}^{K}\Upsilon\left(\Lambda_\theta \mid \mathbb{T}_k\right) \tag{4}$$

3.2 Architecture of Λ_θ

Our architecture is designed for compatibility with various MetaBBO algorithms. We introduce a Population Information Embedding (PIE) module that preprocesses observations $o^t = \{X_i^t, y_i^t\}_{i=1}^m$, allowing Λ_θ to adapt to different search ranges and objective scales. Additionally, a two-stage attention-based encoder (Ts-Attn) enhances Λ_θ 's information extraction capabilities and scalability.

PIE. PIE normalizes observation o^t using two min-max normalization operations: first on the candidate solutions $\{X_i^t\}_{i=1}^m$ against the search range, and second on the objective values $\{y_i^t\}_{i=1}^m$ using the extremum values at time step t. This ensures unified representation and generalization by

Algorithm 1 Pseudo code of training NeurELA

1: **Input**: The MetaBBO task space Ω, Evolution Strategy ES, optimization horizon $maxGen$.
2: **Output**: Optimal neural landscape analyser Λ_{θ^*}
3: Initialize a group of $\{\Lambda_{\theta_i}\}_{i=1}^N$ by $ES.init()$;
4: **for** $generation = 1$ **to** $maxGen$ **do**
5: **for** each Λ_{θ_i} **do**
6: **for** each MetaBBO task $\mathbb{T}_k \in \Omega$ **do**
7: Construct $\mathbb{A}_k(\Pi_\phi, \Gamma, \Lambda_0)$ and $\mathbb{A}'_k(\Pi_\phi, \Gamma, \Lambda_\theta)$;
8: Meta-train $\mathbb{A}_k$ and $\mathbb{A}'_k$ on $\mathbb{D}_{train}$;
9: Test meta-trained $\mathbb{A}_k$ and $\mathbb{A}'_k$ on $\mathbb{D}_{\text{test}}$;
10: **end for**
11: Evaluate the fitness f_i of Λ_{θ_i} by F_{metaELA} in Eq. (4);
12: **end for**
13: Set Λ_{θ^*} as the one with the highest fitness so far;
14: Update the distributional parameters in ES by $ES.update(\{\Lambda_{\theta_i}\}_{i=1}^N, \{f_i\}_{i=1}^N)$;
15: Sample a new generation of $\{\Lambda_{\theta_i}\}_{i=1}^N$ by $ES.sample()$;
16: **end for**

scaling all values to $[0, 1]$. For a d-dimensional optimization problem, the normalized observations o^t are then reorganized into per-dimensional tuples $\{\{(X_{i,j}^t, y_i^t)\}_{i=1}^m\}_{j=1}^d$, with a shape of $d \times m \times 2$. These tuples undergo a linear transformation via a mapping matrix $W_{\text{emb}} \in \mathbb{R}^{2\times h}$, producing the population information encoding E^t, which has the shape $d \times m \times h$. Here, h represents the hidden dimension used in the subsequent two-stage attention module.

Ts-Attn. The computation graph of the Ts-Attn is illustrated on the right side of Figure 2, with a basic component being the attention block ($Attn$) of hidden dimension h. As illustrated in the left side of Figure 2, the $Attn$ block follows the design of the original Transformer Vaswani et al. (2017), except that the layer normalization Ba et al. (2016) is used instead of batch normalization Ioffe & Szegedy (2015). Ts-Attn receives E^t and then advances the information sharing at both cross-solution and cross-dimension levels. **1) Cross-solution attention:** We utilize an *Attn* block to enable same dimensions shared by different candidate solutions within the population to exchange information. **2) Cross-dimension attention:** A second *Attn* block is implemented to further promote the sharing of information across different dimensions within each candidate. To achieve this, the output from the cross-solution phase must be transposed into the shape $m \times d \times h$ and augmented with cosine/sine positional encodings to maintain the dimensional order within a candidate. After these two phases of information processing, the resulting output s^t provides comprehensive optimization status features, suitable for different MetaBBO algorithms. This highly parallelizable, attention-based architecture significantly boosts the scalability of NeurELA as the number of candidates m or dimensions d increases. Additional technical details about the architecture of Λ_θ can be found in Appendix A.1.

3.3 Training Method

Neuroevolution of Λ_θ. The meta-objective of our NeurELA, as defined in Eq. (4), is non-differentiable, since the evaluation of the relative performance $\Upsilon(\Lambda_\theta \mid \mathbb{T}_k)$ in Eq. (3) is not directly derived from the parameter θ of Λ_θ. We in turn employ an Evolution Strategy, denoted as ES, to neuroevolve Λ_θ towards maximizing its expected relative performance for multiple MetaBBO tasks in Ω. The pseudo code of the training is presented in Algorithm 1. The ES first initializes a group of N landscape analysers, then iteratively searches for θ^* through $maxGen$ generations of optimization (lines 4 to 16). Note that the meta-training of the K MetaBBO tasks, which involves training their neural-policies over the training problem set, is nested within the optimization loop (lines 5 to 12) and is extremely time-consuming. We hence employ **Ray** (Moritz et al., 2018), an open-source framework for parallel processing in machine learning applications, to distribute the meta-training pipelines to multiple CPUs. This parallelization significantly enhances the overall efficiency of training NeurELA.

Zero-shot Generalization. The pre-trained Λ_{θ^*} demonstrates robust performance boosts across K representative MetaBBO tasks, enabling seamless integration into unseen MetaBBO tasks. Specifically, Λ_{θ^*} can replace the original Λ_0 in an unseen MetaBBO algorithm, $\mathbb{A}_{\text{unseen}}$. This enables

$\mathbb{A}_{\text{unseen}}$ to concentrate on refining its meta-level neural policy Π_ϕ during training on the designated problem set, utilizing the capabilities of Λ_{θ^*} without requiring any adaptation.

Fine-tuning Adaptation. If Λ_{θ^*} does not perform well in zero-shot generalization, we provide a flexible alternative: fine-tuning it for unseen MetaBBO tasks. In this case, Λ_{θ^*} is integrated into $\mathbb{A}_{\text{unseen}}$ and co-trained with its Π_ϕ during its meta-training process. This method adjusts Λ_{θ^*} to better align with the specific characteristics of $\mathbb{T}_{\text{unseen}}$.

4 Experimental Results

In this section, we discuss the following research questions: **RQ1**: Can NeurELA generalize the pre-trained landscape analyser Λ_{θ^*} to unseen MetaBBO tasks? **RQ2**: If the zero-shot performance is not as expected, is it possible to efficiently fine-tune NeurELA for the underperforming MetaBBO task? **RQ3**: What does NeurELA have learned? **RQ4**: How efficient is the pre-trained landscape analyser in computing landscape features? **RQ5**: How to choose the Evolutionary Strategy for training our NeurELA? **RQ6**: What is the relationship between the model complexity and the generalization performance? Below, we first introduce the experimental settings and then address RQ1 $\sim$ RQ6.

Training Setup. The operating space Ω comprises $K = 3$ MetaBBO tasks. In specific, we first select three representative MetaBBO algorithms LDE Sun et al. (2021), RLEPSO Yin et al. (2021)and RL-DAS Guo et al. (2024a) from existing MetaBBO literature. Then we set the associated problem set $\mathbb{D}_k$ for these MetaBBO algorithms as the BBOB testsuites in COCO Hansen et al. (2021), which includes a variety of optimization problems with diverse landscape properties. We have to note that the selected algorithms cover diverse MetaBBO scenarios such as auto-configuration for control parameters and auto-selection for evolutionary operators, hence ensuring the generalization of our NeurELA. We follow the original settings of these MetaBBO algorithms during their meta-training, as suugested in their papers. For the settings about the neural network, we set the hidden dimension $h = 16$ for the neural network modules in the landscape analyser Λ_θ. With a single-head attention in $Attn$, Λ_θ possess a total number of 3296 learnable parameters. We adopt Fast CMA-ES Li et al. (2018) as ES for its searching efficiency and robust optimization performance. During the training, we employ a population of $N = 10$ Λ_θs within a generation and optimize the population for $maxGen = 50$ generations. For each generation, we parallel the $N \times K = 30$ meta-training pipelines across 30 independent CPU cores using **Ray** Moritz et al. (2018). For each low-level optimization process, we set the maximum function evaluations as 20000. The experiments are run on a computation node of a Slurm CPU cluster, with two Intel Sapphire Rapids 6458Q CPUs and 256 GB memories. Due to the limitation of the space, we present more technical details such as the train-test split $\{\mathbb{D}_{\text{train}}, \mathbb{D}_{\text{test}}\}$, the control parameters of Fast CMAES and the use of open-sourced software in Appendix A.2 $\sim$ A.5.

Baselines. We consider three baselines for comparison: **1) Original**, which denotes the original hand-crafted landscape analyser functions (Λ_0) in the MetaBBO algorithms. **2) ELA**, which denotes the traditional ELA feature groups summarized in Section 2 and defined in (Mersmann et al., 2011), which we implement by the open-sourced Pflacco package (Kerschke & Trautmann, 2019b). **3) DeepELA** (Seiler et al., 2024), which is a recent work which trains an attention-based neural network as an alternative of traditional ELA features through self-supervised contrastive learning.

4.1 Zero-shot Performance (RQ1)

Recall that a MetaBBO task is composed of an MetaBBO algorithm and an associated optimization problem set (i.e., $\mathbb{T}_k = (\mathbb{A}_k, \mathbb{D}_k)$). We hence generalize the pre-trained landscape analyser Λ_{θ^*} towards three zero-shot cases: 1) five unseen MetaBBO algorithms DEDDQN Sharma et al. (2019), RLPSO Wu & Wang (2022), GLEET Ma et al. (2024), MelBA (Chaybouti et al., 2022) and GLHF (Li et al., 2024); 2) two unseen problem set BBOB-Noisy Hansen et al. (2021) and Protein-Docking benchmark Hwang et al. (2010); 3) both the unseen MetaBBO algorithms and problem sets. Those tested MetaBBO tasks cover a wide range of meta-learned black-box optimizers. Besides, the problem sets include challenging realistic application, e.g., Protein-Docking. We evaluate the performance of the ELA, DeepELA and our NeurELA by calculating the relative optimization performance (Eq. (3)) against the Original baseline in the given MetaBBO tasks. Since the performance of Original baseline would be 0 within this setting, we report the exponential of the calculated relative performances

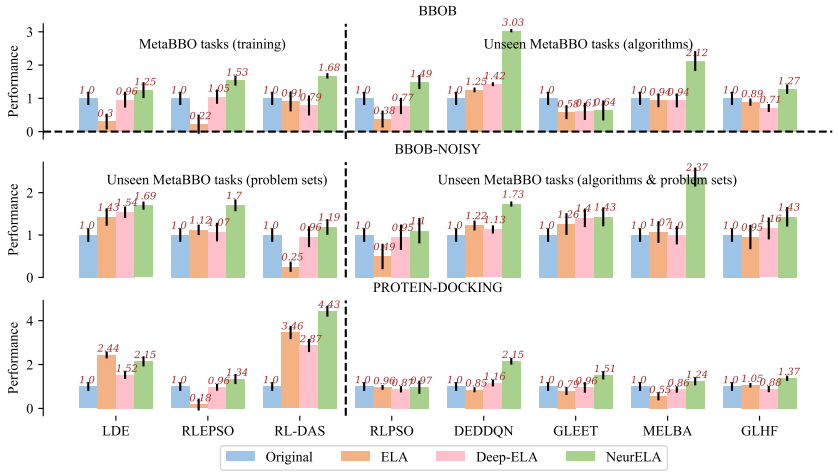

Figure 3: Zero-shot performance of NeurELA on unseen MetaBBO algorithms and problem sets.

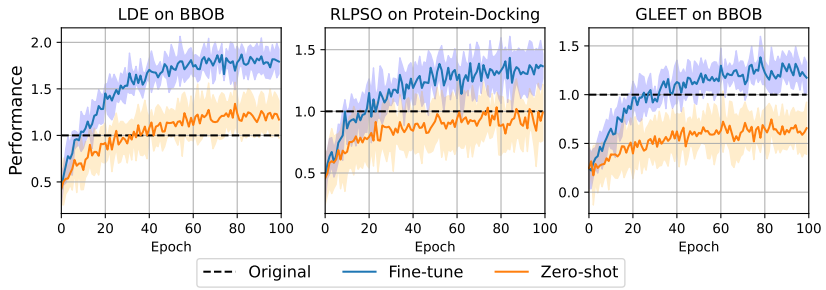

Figure 4: Performance gain curves of fine-tuning NeurELA for a specific MetaBBO task.

in Figure 3, where the bars denote the average relative performance, the error bars denote the 95% bootstrapped confidence interval, for 10 runs of meta-training. For BBOB-Noisy set, we set the maximum function evaluations as 20000, while for Protein-Docking set, we set it to 1000. The three zero-shot cases are split by the dashed lines.

The results in Figure 3 show that: 1) Over all of the three zero-shot cases, NeurELA generally achieves superior performance against original feature extraction designs (Original) in these MetaBBO algorithms, as well as the traditional landscape analysis features (ELA) and DeepELA. 2) The results of integrating ELA into the evaluated MetaBBO tasks are quite noisy, which might be due to the intricate correlations in-between the feature groups. Such strong correlations might challenge the information processing ability of the neural policies Π_ϕ in some MetaBBO algorithms. 3) We observe that our NeurELA significantly boosts the performance of Original baseline on unseen MetaBBO algorithms & problem sets case, which not only validates the effectiveness of our design, but also indicates that the specific feature extraction designs (Origin) in existing MetaBBO algorithms might overfits to the problem set it is meta-trained on. 4) Compared with traditional ELA, DeepELA show similar performance, which is in accordance with its the training objective to serve as automated alternative of traditional ELA features. 5) Our NeurELA consistently outperforms the traditional ELA features and DeepELA on the BBOB-Noisy testsuites and Protein-Docking benchmark, which comprises a wide range of optimization problems with either different noise models and levels (e.g., Gaussian noise and Cauchy noise Hansen et al. (2009)) or intricate realistic problem features. It indicates that our NeurELA shows robust generalization ability to unseen MetaBBO tasks, which involve both unseen MetaBBO algorithm and unseen realistic optimization problems.

4.2 Fine-tuning Performance (RQ2)

We select three representative MetaBBO task from the zero-shot performance analysis (Figure 3) to validate fine-tuning adaption efficiency of our NeurELA. The selected MetaBBO tasks includes: 1) LDE on BBOB problem set, where NeurELA surpasses the Original baseline in LDE. 2) RLPSO

on Protein-Docking problem set, where NeurELA equally performs against the Original baseline in RLPSO. 3) GLEET on BBOB problem set, where NeurELA does not achieves the expected zero-shot performance. Concretely, we integrate the pre-trained Λ_{θ^*} into the corresponding MetaBBO algorithm, then fine-tune it in conjunction with the meta-training over the given problem set. We depict the performance gain curves during the meta-training in Figure 4. We additionally plot the performance gain curves of the meta-training during the zero-shot generalization for comparison. The results showcase the flexible adaption potential of our NeurELA: 1) fine-tuning NeurELA on specific MetaBBO tasks where our NeurELA outperforms, equally performs or underperforms against the Original baseline would consistently introduce significant performance boosts. 2) the fine-tuning strategy shows remarkable efficiency in adapting NeurELA for new MetaBBO algorithm dynamics and optimization problem structures, which only requires less than 20% meta-training epochs to achieve the similar zero-shot performance.

4.3 What has NeurELA learned? (RQ3)

In this section, we presents the advantage of the landscape features learned by NeurELA against traditional ELA features under the MetaBBO settings. Concretely, we take LDE (Sun et al., 2021) as a showcase. In LDE, the meta-level policy (a LSTM-based RL agent) outputs the mutation strength factor $F^t \in (0, 1)$ for the low-level DE per optimization step t. F^t indicates the exploration-exploitation tradeoff during the optimization (larger value indicates exploration, vice versa). We first record the per step populations during the optimization process of LDE (10 runs on 10-D Rastrigin in CoCo-BBOB testsuites). We then simply classify the recorded populations as exploration population ($F^t > 0.5$) and exploitation population ($F^t \leq 0.5$). We now illustrate the low-rank visualization of the ELA feature and our NeurELA feature for the recorded populations in Figure 5. The data points denoted the 2-dimensional features obtained by first feeding the populations into ELA/NeurELA and then using PCA (Shlens, 2014) transformation on the output landscape features. Red points and blue points represent the features for the exploration and exploitation population respectively. The results validate that NeurELA is more sensitive than traditional ELA features on the dynamics of low-level optimization. Since traditional ELA is mainly used for describing an optimization problem, it might falls short in providing informative low-level optimization status for MetaBBO methods. This explains the consistent superior performance of our NeurELA (see Figure 3). Compared with the traditional ELA features, our NeurELA can be meta-trained on MetaBBO tasks to provide sensitive and accurate low-level optimization status for the meta-level policy. We also provide a further interpretation analysis which reveals the correlation of features learned by NeurELA and traditional ELA features, refer to Appendix B.3 for detailed discussion.

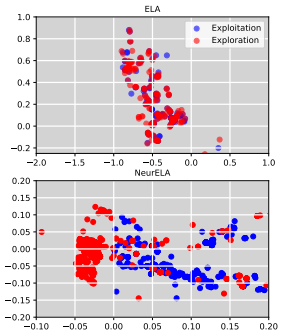

Figure 5: Low-level optimization features obtained by traditional ELA and our NeurELA, through PCA dimension reduction.

4.4 In-depth Study (RQ4 $\sim$ RQ6)

Computational Efficiency Comparison (RQ4). We compare the computational efficiency of our NeurELA with Original and ELA as either the amount of candidate samples m or the dimension of the optimization problem d scales. We report the average wall-time required by each baseline and our NeurELA to produce the landscape features (for 1000 runs). The results show that: 1) Our NeurELA achieves comparative computational efficiency with the specific hand-crafted designs in existing MetaBBO algorithms, while introducing significant performance boosts; 2) When the sample size m increases, our NeurELA becomes more time-efficient than the original feature extraction designs in existing MetaBBO algorithms. This is primarily owning to the two-stage attention architecture in our NeurELA, which can be efficiently computed on CPUs in parallel. 3) The traditional ELA features encounters the curse of dimensionality, which makes itself impractical for MetaBBO algorithms to solve high-dimensional optimization problems.

Evolution Strategy Selection (RQ5). Within this paper, we employ a neuroevolution process to optimize the parameters of the landscape analyzer Λ_θ, which presents a large-scale optimization challenge as Λ_θ comprises 3296 learnable parameters. To identify an effective ES, we implement

Table 1: The average wall time (in seconds) for computing features.

	$m = 100$		
	$d = 10$	$d = 100$	$d = 1000$
Original	**4.99E-03**	**5.61E-03**	**6.38E-03**
ELA	1.49E-01	1.48E+01	3.00E+02
NeurELA	5.42E-03	6.24E-03	7.11E-03
	$d = 10$		
	$m = 100$	$m = 500$	$m = 1000$
Original	**4.99E-03**	2.53E-02	5.16E-02
ELA	1.49E-01	1.69E-01	2.02E-01
NeurELA	5.42E-03	**7.97E-03**	**9.38E-03**

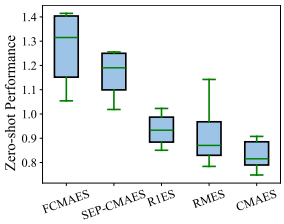

Figure 6: Zero-shot performance when using different evolution strategies.

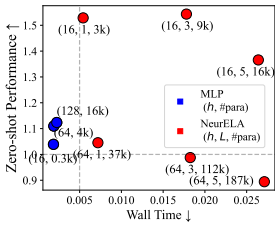

Figure 7: The influence of model complexity on the zero-shot performance.

several candidate evolution strategy variants: Fast CMAES Li et al. (2018), SEP-CMAES Ros & Hansen (2008), R1ES, RMES Li & Zhang (2017), and the original CMAES Hansen & Ostermeier (2001b), and report the overall zero-shot performance (unseen algorithms & unseen problem sets) achieved by training our NeurELA with these candidate strategies for 10 runs in Figure 6. The results reveal that Fast CMAES dominates the other candidates, and is hence selected as the neural network optimizer ES for our NeurELA. The convergence of these ES baselines are shown in Appendix B.1.

Model Complexity (RQ6). We discuss the relationship between the model complexity and the zero-shot performance (unseen MetaBBO algorithm & problem sets) of our NeurELA. Concretely, We pre-train NeurELA under 6 different model complexities, with various hidden dimensions, i.e., $h = (16, 64)$, and the number of the Ts-Attn module, i.e., $l = (1, 3, 5)$. We additionally pre-train three MLP baselines, which substitute the Ts-Attn module in NeurELA with a linear feed-forward layer, which holds a shape of $h \times h, h = (16, 64, 128)$. We report both the zero-shot performance (y-axis) and the computational efficiency (x-axis, presented as the consumed wall time for computing the landscape features) in Figure 7, where the dashed lines denotes the performance and wall-time of the Original baseline. $\#para$ denotes the number of the learnable parameters. The results show that: 1) a significant performance gap is observed between the MLP baselines and our Ts-Attn module ($h = 16, l = 1$). It validates the effectiveness of our Ts-Attn design, which enhance the feature extraction of our NeurELA by encouraging the information sharing at both the cross-solution and cross-dimension levels; 2) As the model complexity increases, the performance of the Ts-Attn module drops rapidly. It reveals that the increased number of learnable parameters challenges the optimization ability of the backend ES. Given the limited computational resources, it is difficult to identify the optimal parameters θ^*.

Limitations. As a pioneer work on using deep neural network for landscape analysis in MetaBBO, our NeurELA show possibility of learning an universal landscape analyser with minimal expertise requirement and appealing optimization performance. However, a major limitation is that the training efficiency of NeurELA is not very satisfactory. As discussed in the Model Complexity part, the zero-shot performance drops rapidly when the model complexity increases. This is due to the relatively limited large-scale optimization ability of existing evolution strategies. Unfortunately, the meta-objective of NeurELA (Eq. (4)) is not differentiable hence cannot benefit from back propagation methods. We denote this limitation as an important future work to address.

5 Conclusion

In this paper, we introduce NeurELA to complement *the last mile* in making the MetaBBO paradigms entirely end-to-end: the automatic extraction of optimization status features. To this end, we propose a landscape analyser parameterized by a two-stage attention-based neural network. The analyser enables flexible replacement for the original hand-crafted landscape analysis mechanisms in existing MetaBBO algorithms. We employ a neuroevolution paradigm to optimize it by maximizing performance over a multi-task MetaBBO operating space. The experimental results showcase that our NeurELA serves two key functions: 1) it can be seamlessly integrated into unseen MetaBBO algorithms for impressive zero-shot performance, and 2) it can also be fine-tuned together with new MetaBBO algorithms to adapt its performance for specific optimization challenges. While it possesses certain limitation (e.g., low training efficiency), our work represents a significant step toward realizing a fully automatic MetaBBO algorithm—a paradigm that learns to optimize without the need for manual expertise.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *International Conference on Knowledge Discovery & Data mining*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Leo Breiman. Random forests. *Machine Learning*, 2001.

Sofian Chaybouti, Ludovic Dos Santos, Cedric Malherbe, and Aladin Virmaux. Meta-learning of black-box solvers using deep reinforcement learning. In *NeurIPS 2022, MetaLearn Workshop*, 2022.

Jiacheng Chen, Zeyuan Ma, Hongshu Guo, Yining Ma, Jie Zhang, and Yue-Jiao Gong. SYMBOL: Generating flexible black-box optimizers through symbolic equation learning. In *International Conference on Learning Representations*, 2024.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 1995.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Hongshu Guo, Yining Ma, Zeyuan Ma, Jiacheng Chen, Xinglin Zhang, Zhiguang Cao, Jun Zhang, and Yue-Jiao Gong. Deep reinforcement learning for dynamic algorithm selection: A proof-of-principle study on differential evolution. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2024a.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations*, 2024b.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 2001a.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 2001b.

Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 2003.

Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. *Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions*. PhD thesis, INRIA, 2009.

Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. Coco: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 2021.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

John H Holland. *Adaptation in Natural and Artificial Systems*. 1975.

Howook Hwang, Thom Vreven, Joël Janin, and Zhiping Weng. Protein–protein docking benchmark version 4.0. *Proteins: Structure, Function, and Bioinformatics*, 2010.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

Terry Jones, Stephanie Forrest, et al. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *ICGA*, 1995.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of the International Conference on Neural Networks*, 1995.

Pascal Kerschke and Heike Trautmann. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolutionary Computation*, 2019a.

Pascal Kerschke and Heike Trautmann. Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the r-package flacco. *Applications in Statistical Computing*, 2019b.

Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. Detecting funnel structures by means of exploratory landscape analysis. In *Annual Conference on Genetic and Evolutionary Computation*, 2015.

Pascal Kerschke, Holger H Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary Computation*, 2019.

Robert Lange, Tom Schaul, Yutian Chen, Tom Zahavy, Valentin Dalibard, Chris Lu, Satinder Singh, and Sebastian Flennerhag. Discovering evolution strategies via meta-black-box optimization. In *Companion Conference on Genetic and Evolutionary Computation*, 2023.

Robert Tjarko Lange, Yingtao Tian, and Yujin Tang. Evolution transformer: In-context evolutionary optimization. *arXiv preprint arXiv:2403.02985*, 2024.

Xiaobin Li, Kai Wu, Yujian Betterest Li, Xiaoyu Zhang, Handing Wang, and Jing Liu. Glhf: General learned evolutionary algorithm via hyper functions. *arXiv preprint arXiv:2405.03728*, 2024.

Zhenhua Li and Qingfu Zhang. A simple yet efficient evolution strategy for large-scale black-box optimization. *IEEE Transactions on Evolutionary Computation*, 2017.

Zhenhua Li, Qingfu Zhang, Xi Lin, and Hui-Ling Zhen. Fast covariance matrix adaptation for large-scale black-box optimization. *IEEE Transactions on Cybernetics*, 2018.

Hongqiao Lian, Zeyuan Ma, Hongshu Guo, Ting Huang, and Yue-Jiao Gong. Rlemmo: Evolutionary multimodal optimization assisted by deep reinforcement learning. In *Genetic and Evolutionary Computation Conference Companion*, 2024.

Monte Lunacek and Darrell Whitley. The dispersion metric and the cma evolution strategy. In *Annual Conference on Genetic and Evolutionary Computation*, 2006.

Zeyuan Ma, Hongshu Guo, Jiacheng Chen, Zhenrui Li, Guojun Peng, Yue-Jiao Gong, Yining Ma, and Zhiguang Cao. Metabox: A benchmark platform for meta-black-box optimization with reinforcement learning. In *Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Zeyuan Ma, Jiacheng Chen, Hongshu Guo, Yining Ma, and Yue-Jiao Gong. Auto-configuring exploration-exploitation tradeoff in evolutionary computation via deep reinforcement learning. In *Genetic and Evolutionary Computation Conference Companion*, 2024.

Katherine M Malan and Andries P Engelbrecht. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 2013.

Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Annual Conference on Genetic and Evolutionary Computation*, 2011.

Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. 2024.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging ai applications. In *Symposium on Operating Systems Design and Implementation*, 2018.

Mario A Muñoz, Michael Kirley, and Saman K Halgamuge. Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE transactions on evolutionary computation*, 2014.

Erik Pitzer and Michael Affenzeller. A comprehensive survey on fitness landscape analysis. *Recent Advances in Intelligent Engineering Systems*, 2012.

Raphael Patrick Prager, Moritz Vinzent Seiler, Heike Trautmann, and Pascal Kerschke. Towards feature-free automated algorithm selection for single-objective continuous black-box optimization. In *IEEE Symposium Series on Computational Intelligence*, 2021a.

Raphael Patrick Prager, Moritz Vinzent Seiler, Heike Trautmann, and Pascal Kerschke. Towards feature-free automated algorithm selection for single-objective continuous black-box optimization. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, 2021b. doi: 10.1109/SSCI50451.2021.9660174.

Quentin Renau, Johann Dreo, Carola Doerr, and Benjamin Doerr. Expressiveness and robustness of landscape features. In *Genetic and Evolutionary Computation Conference Companion*, 2019.

Quentin Renau, Carola Doerr, Johann Dreo, and Benjamin Doerr. Exploratory landscape analysis is strongly sensitive to the sampling strategy. In *International Conference on Parallel Problem Solving from Nature*, 2020.

John R Rice. The algorithm selection problem. In *Advances in Computers*. 1976.

Raymond Ros and Nikolaus Hansen. A simple modification in cma-es achieving linear time and space complexity. In *International Conference on Parallel Problem Solving from Nature*, 2008.

Moritz Vinzent Seiler, Raphael Patrick Prager, Pascal Kerschke, and Heike Trautmann. A collection of deep learning-based feature-free approaches for characterizing single-objective continuous fitness landscapes. In *Genetic and Evolutionary Computation Conference Companion*, 2022a.

Moritz Vinzent Seiler, Raphael Patrick Prager, Pascal Kerschke, and Heike Trautmann. A collection of deep learning-based feature-free approaches for characterizing single-objective continuous fitness landscapes. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2022b.

Moritz Vinzent Seiler, Pascal Kerschke, and Heike Trautmann. Deep-ela: Deep exploratory landscape analysis with self-supervised pretrained transformers for single-and multi-objective continuous optimization problems. *arXiv preprint arXiv:2401.01192*, 2024.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 2015.

Mudita Sharma, Alexandros Komninos, Manuel López-Ibáñez, and Dimitar Kazakov. Deep reinforcement learning based parameter control in differential evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019.

Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

Lei Song, Chenxiao Gao, Ke Xue, Chenyang Wu, Dong Li, Jianye Hao, Zongzhang Zhang, and Chao Qian. Reinforced in-context black-box optimization. *arXiv preprint arXiv:2402.17423*, 2024.

Rainer Storn and Kenneth Price. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 1997.

Jianyong Sun, Xin Liu, Thomas Bäck, and Zongben Xu. Learning adaptive differential evolution algorithm from optimization experiences by policy gradient. *IEEE Transactions on Evolutionary Computation*, 2021.

Zhiping Tan and Kangshun Li. Differential evolution with mixed mutation strategy based on deep reinforcement learning. *Applied Soft Computing*, 2021.

Zhiping Tan, Yu Tang, Kangshun Li, Huasheng Huang, and Shaoming Luo. Differential evolution with hybrid parameters and mutation strategies based on reinforcement learning. *Swarm and Evolutionary Computation*, 2022.

Tomer Tsaban, Julia K Varga, Orly Avraham, Ziv Ben-Aharon, Alisa Khramushin, and Ora Schueler-Furman. Harnessing protein folding neural networks for peptide–protein docking. *Nature Communications*, 2022.

Bas van Stein, Fu Xing Long, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. Doe2vec: Deep-learning based features for exploratory landscape analysis. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, GECCO '23 Companion, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Sewall Wright et al. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. 1932.

Di Wu and G Gary Wang. Employing reinforcement learning to enhance particle swarm optimization methods. *Engineering Optimization*, 2022.

Ke Xue, Jiacheng Xu, Lei Yuan, Miqing Li, Chao Qian, Zongzhang Zhang, and Yang Yu. Multi-agent dynamic algorithm configuration. *Advances in Neural Information Processing Systems*, 2022.

Shiyuan Yin, Yi Liu, GuoLiang Gong, Huaxiang Lu, and Wenchang Li. Rlepso: Reinforcement learning based ensemble particle swarm optimizer. In *International Conference on Algorithms, Computing and Artificial Intelligence*, 2021.