Surrogate Learning in Meta-Black-Box Optimization: A Preliminary Study

Zeyuan Ma
scut.crazynicolas@gmail.com
South China University of Technology
Guangzhou, Guangdong, China

Zhiyang Huang
scut.hzy@gmail.com
South China University of Technology
Guangzhou, Guangdong, China

Jiacheng Chen
jackchan9345@gmail.com
South China University of Technology
Guangzhou, Guangdong, China

Zhiguang Cao
zhiguangcao@outlook.com
Singapore Management University
Singapore, Singapore

Yue-Jiao Gong*
gongyuejiao@gmail.com
South China University of Technology
Guangzhou, Guangdong, China

Abstract

Recent Meta-Black-Box Optimization (MetaBBO) approaches have shown possibility of enhancing the optimization performance through learning meta-level policies to dynamically configure low-level optimizers. However, existing MetaBBO approaches potentially consume massive function evaluations to train their meta-level policies. Inspired by the recent trend of using surrogate models for cost-friendly evaluation of expensive optimization problems, in this paper, we propose a novel MetaBBO framework which combines surrogate learning process and reinforcement learning-aided Differential Evolution algorithm, namely Surr-RLDE, to address the intensive function evaluation in MetaBBO. Surr-RLDE comprises two learning stages: surrogate learning and policy learning. In surrogate learning, we train a Kolmogorov-Arnold Networks (KAN) with a novel relative-order-aware loss to accurately approximate the objective functions of the problem instances used for subsequent policy learning. In policy learning, we employ reinforcement learning (RL) to dynamically configure the mutation operator in DE. The learned surrogate model is integrated into the training of the RL-based policy to substitute for the original objective function, which effectively reduces consumed evaluations during policy learning. Extensive benchmark results demonstrate that Surr-RLDE not only shows competitive performance to recent baselines, but also shows compelling generalization for higher-dimensional problems. Further ablation studies underscore the effectiveness of each technical components in Surr-RLDE. We open-source Surr-RLDE at https://github.com/GMC-DRL/Surr-RLDE.

CCS Concepts

• **Computing methodologies → Control methods**.

Keywords

Meta-Black-Box-Optimization, Dynamic Algorithm Configuration

1 Introduction

Black-Box Optimization (BBO) [1] holds a wide range of applications and requires effective optimization techniques such as Evolutionary Computation (EC) to address them. Recently, Meta-Black-Box Optimization (MetaBBO) [27, 43] has gained incremental research interests by showing a feasible paradigm of combining

*Corresponding Author

learning-based techniques (e.g., reinforcement learning [16]) and EC to reduce expertise dependent tuning of human-crafted EC methods when addressing BBO problems. MetaBBO comprises bi-level optimization [26]: a meta-level policy interacts with a low-level BBO process by configuring the low-level EC method for optimization and collecting feedback signal from the performance improvement during the low-level optimization process. The meta-level policy can then be meta-learned by maximizing the accumulated feedback signals over a problem distribution to attain a generalizable and flexible policy capable of optimizing novel BBO problems, with minimal expertise requirement.

Although MetaBBO methods [9, 10, 21, 25, 33] have shown promising performance in enhancing traditional EC methods, their training processes unavoidably consume massive computational resources. This attributes to the inherent workflow of existing MetaBBO methods, where the optimization trajectory is repeatedly sampled through the interaction of meta-level policy and low-level optimization which involves numbers of function evaluations on the training optimization problems. Considering the development of surrogate models in recent data-driven evolutionary algorithms [15], the core research question of this paper comes out: can surrogate modeling techniques be combined with MetaBBO approach to reduce the massive number of function evaluations consumed during the meta learning process?

To explore this research question, in this paper, we propose the use of surrogate models as substitutes for the original low-level function evaluation of MetaBBO. Specifically, we adopt Kolmogorov-Arnold Network (KAN) [24] as the surrogate model for its validated robustness in formula representation [44], and train each training problem instance a separate KAN model. To improve the fitting accuracy during the surrogate learning, we additionally design a relative-order-aware loss function to ensure that surrogate model maintains the relative objective value ranks between different sample points. As a preliminary study, we integrated the learned KAN-based surrogate model into a MetaBBO framework to evaluate its performance. In our setup, the MetaBBO framework is designed to learn a reinforcement learning-based meta-level policy capable of dynamically configuring Differential Evolution (DE [7]), namely Surr-RLDE. Specifically, we construct a comprehensive configuration space including both the operator selection and the parameter tuning for DE mutation operator. The meta-level policy is meta-trained to provide flexible mutation configurations for

the low-level DE hence maximizing the optimization performance sorely dependent on KAN-based surrogate models of the training problem instances. Extensive benchmark results demonstrate that incorporating surrogate models with meta-learning achieves comparable or even superior optimization performance to existing MetaBBO baselines, while saving all function evaluations needed during training. Additional ablation studies further validate that design choices in Surr-RLDE contribute to the satisfactory performance indeed, e.g., the proposed novel Relative-Order-Aware(ROA) loss function and the selection of a KAN as the surrogate model.

2 Related Works

2.1 Meta-Black-Box Optimization

Meta-Black-Box Optimization (MetaBBO) [27, 43] aims to reduce the manual effort typically required for adaptive tuning in traditional optimization algorithms. Although there is a variety of MetaBBO approaches that adopt different learning techniques in the meta-level, in this paper, we focus on a particular kind of MetaBBO approach, those training their policies with RL. Specifically, MetaBBO can be modeled as a Markov Decision Process (MDP), $\mathcal{M} := \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$. At each time step t, the meta-level RL-Agent π_ω takes current optimization state $s^t \in \mathcal{S}$ as input and accordingly outputs action $a^t = \pi_\omega(s^t) \in \mathcal{A}$. In MetaBBO, a^t is applied to low-level strategy λ to construct a completed optimizer, which is to interact with optimization tasks directly. After a^t is executed, next state s^{t+1} is return according to transition dynamic $\mathcal{T}$ and reward r^t can be calculated. Given a dataset of problem instances $\mathbb{P}$, the meta-level objective of π_ω is defined to maximize the expectation of r^t over $\mathbb{P}$. Mathematically:

$$\pi_\omega^* = \underset{\pi_\omega}{argmax} \left[\mathbb{E}_{f \in \mathbb{P}} \left[\sum_{t=1}^{T} r^t \right] \right] \qquad (1)$$

where r^t quantifies the performance gain at each time step, which usually demonstrates the relative performance gain between consecutive steps, which is highly dependent on the relative fitness among candidate solutions.

Among existing MetaBBO methods that utilize RL as the meta-level learning algorithm, the action space varies widely, including operator selection (e.g., DEDDQN [32], DEDQN [36]), algorithm selection (e.g., RLDAS [9]), hyperparameter tuning (e.g., LDE [33], GLEET [25]), and even algorithm generation (e.g., SYMBOL [5]). In addition to RL-based approaches, other methods have also been investigated. For instance, LES [21] and LGA [20] adopt CMA-ES [11] as the meta-level strategy, while B2Opt [23] and POM [22] employ supervised learning for meta-level training.

Despite these advances, current MetaBBO methods are predominantly applied to synthetic optimization tasks [26], such as CEC benchmark functions [29], particularly during the training phase. This limitation arises from the high computational cost of MetaBBO training, which often requires a large number of additional fitness evaluations. For example, the authors of SYMBOL reported the need for an extra 528M evaluations during training, rendering it impractical for many real-world applications where evaluation is costly. To address this issue, incorporating surrogate models as substitutes for fitness evaluations during training could significantly expand the applicability of MetaBBO to more complex and expensive scenarios.

2.2 Surrogate Model Learning

Surrogate models are usually used as simplified approximations of more complex and computationally expensive functions [2, 19]. The motivation for using surrogate models comes from the prohibitive expense of direct evaluations in many real-world scenarios, such as engineering design optimization [38], scientific simulations [3], and hyperparameter tuning for machine learning algorithms [42]. Training surrogate models typically involves collecting a set of samples from the original function, consisting of input-output pairs, and using these samples to fit the surrogate model. This process often employs regression techniques to minimize the prediction error in the training data. Common approaches to surrogate model learning include Gaussian Processes (GP) [41] and various deep-learning-based methods such as neural networks [39]. Each of these models has unique strengths: GPs excel in providing uncertainty estimates, making them particularly useful for small-scale optimization tasks, while neural networks are favored for their scalability and ability to handle high-dimensional inputs in large datasets.

Surrogate models have demonstrated success in diverse applications. For instance, in design optimization [4], surrogate models are used to approximate expensive finite element simulations, enabling rapid prototyping and testing. In evolutionary algorithms [15], surrogate models are used to replace the expensive fitness evaluation function or serve as an approximation when only historical data are available. In our method, we utilized KAN to build surrogate models, which are employed to replace the real fitness evaluation during the training phase of MetaBBO.

2.3 Kolmogorov-Arnold Network

Formulation. KAN (Kolmogorov-Arnold Network) [24] is a type of neural network inspired by the Kolmogorov-Arnold representation theorem. This theorem states that any multivariate continuous function can be expressed as the composition of a finite number of continuous univariate functions. Specifically, the representation is given by:

$$f(\mathbf{x}) = f(x_1, \cdots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right) \qquad (2)$$

where Φ and ϕ are univariate functions. When applied in a multi-layer fashion, the formula for a KAN becomes:

$$\text{KAN}(x) = \Phi^L \left(\Phi^{L-1} \left(\dots \left(\Phi^2 \left(\Phi^1(x) \right) \right) \right) \right) \qquad (3)$$

where x denotes the input vector and Φ_k represents the transformation matrix of the k-th layer. For example, the calculation in layer 1 is:

$$\Phi^1(x) = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{m1} & \phi_{m2} & \cdots & \phi_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad (4)$$

To make ϕ_{ij} trainable and easy to implement, the authors adopt the idea of B-splines. Each $\phi(x)$ is represented as:
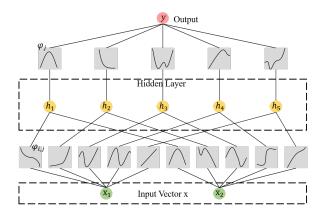
$$\phi(x) = w_b b(x) + w_s \, \text{spline}(x) \qquad (5)$$

Figure 1: Neural network architecture of KAN used in this paper, where input and hidden layer dimension is limited to 2 **and** 5 **respectively for the convenience of presentation.**

where w_b and w_s are both trainable scalar parameters, and $b(x)$ is defined as:

$$b(x) = \text{silu}(x) = x/(1 + e^{-x}) \tag{6}$$

$b(x)$ plays a role similar to residual connections. The $\text{spline}(x)$ component is parameterized as a linear combination of B-splines:

$$\text{spline}(x) = \sum_i c_i B_i(x) \tag{7}$$

where c_i are trainable parameters, and $B_i(x)$ are the B-spline basis functions.

Application. KAN has demonstrated significant success in various domains, particularly in tasks involving high-dimensional data and complex function approximation. For instance, KANs have been effectively applied in areas such as image and video processing [6], as well as function approximation [8]. In these contexts, KANs have been utilized to model intricate processes by decomposing high-dimensional relationships into manageable, low-dimensional representations.

However, the application of KAN to MetaBBO methods remains an area of active exploration. One related study [13] employs KAN within a Surrogate-Assisted Evolutionary Algorithm (SAEA), where KAN serves as a surrogate model to guide the selection of promising solutions during the search process.

3 Methodology

As we elaborated in previous sections, a key bottleneck of existing MetaBBO approaches is that they unavoidably consume massive number of function evaluations to train the meta-level policy, which might be neither practical nor efficient in many realistic BBO scenarios. The core motivation of Surr-RLDE is to explore an interesting research question: Can MetaBBO be meta-trained with surrogate models of the low-level optimization problems (which might introduces biased learning signal), while maintaining the learning effectiveness and the final optimization performance? To find the answer, we design two learning stages: Surrogate Learning Stage (SLS) and Policy Learning Stage (PLS) in Surr-RLDE, of which the problem definitions and detailed technical designs are elaborated in Section 3.1 and Section 3.2 respectively.

Algorithm 1: Surrogate Model Training Process.

Input: Dataset D_f, Initialized model F_θ, Batch size N_{batch},
Training epochs T_{mse} and T_{roa}, Learning rate η.
Output: Optimal surrogate model $F_{\theta*}$.

1 **for** $epoch \leftarrow 1$ **to** T_{mse} **do**
2   **for** *each batch in* D_f **do**
3        $\mathcal{L}(\theta) \leftarrow \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} MSE(f(x_i), F_\theta(x_i))$;
4        $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$;
5   **end**
6 **end**
7 Set λ in Eq. 9 as 1;
8 **for** $epoch \leftarrow 1$ **to** T_{roa} **do**
9   **for** *each batch in* D_f **do**
10   Sort x_i in the batch by $f(x_i)$ in descending order;
11   Compute ROA loss $\mathcal{L}(\theta)$ described in Eq. 9;
12       $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$;
13   **end**
14    $\lambda \leftarrow \lambda \cdot (1 - \frac{epoch}{T_{mix}})$
15 **end**
16 **return** *The trained* F_θ

3.1 Surrogate Learning Stage

3.1.1 Problem Definition. The first stage of Surr-RLDE is Surrogate Learning Stage (SLS). SLS aims to approximate the objective functions of the low-level BBO problems used to meta-train the meta-level policy, hence saving function evaluations on these problems during the training. Intuitively, the primary objective in this stage is to learn a surrogate model F_θ (a neural network parameterized with θ) that regresses a given D-dimensional BBO problem $f : \mathbb{R}^D \rightarrow \mathbb{R}$ with minimal MSE regression error [18]:

$$\text{Minimize} : \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} (f(x_i) - F_\theta(x_i))^2, x_i \in D_f \tag{8}$$

where $D_f : \{x_i, f(x_i)\}_{i=1}^{N}$ is a collection of N solutions uniformly and randomly sampled from the definition domain of the decision variables in BBO problem f, and their corresponding function values. In surrogate learning literature [2, 19], D_f is regarded as the training dataset and the above MSE regression error term is regarded as the training loss function to learn optimal θ^* of the surrogate model F_θ. Once trained, $F_{\theta*}$ is expected to serve as an accurate surrogate function on those solution positions out of D_f.

3.1.2 Surrogate Model Design. In Surr-RLDE, we employ a KAN [24] neural network for SLS. We illustrate a simple architecture example of KAN (with a hidden layer) in Fig. 1, where the input dimension L_{in} is 2, the hidden dimension L_{hidden} is 5, the output dimension L_{out} is 1. The computational workflow is that: first, given an input x with L_{in} dimensions, for each i-th dimension x_i, we connect it with each j-th dimension h_j in the hidden layer, through a transformation function $\phi_{i,j}$ (see Eq. 5). The final post-activation value of each hidden unit h_j is the summation of all connections with the input x: $h_j = \sum_{i=1}^{L_{in}} \phi_{i,j}(x_i)$. To attain the output value y, we further connect each h_j in the hidden layer with it, through a transformation function ϕ_j. Then y is attained by summing up all connections: $y = \sum_{j=1}^{L_{hidden}} \phi_j(h_j)$. In the rest of this paper, we use F_θ to refer to
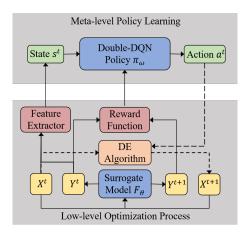
Figure 2: Workflow of Policy Learning Stage in Surr-RLDE.

the KAN-based surrogate model, where θ is the trainable parameters in the model, including the combination weights (w_b, w_s and c_k in Eq. 5 and Eq. 7) within the transformation function ϕ of each connections. Given an input x, we attain $y = F_\theta(x)$ directly.

We have to note that there are two core insights that motivate us to use KAN as the surrogate model rather than MLP neural network: a) KAN replaces the fixed activation functions in MLP by learnable spline function combinations, while maintaining the fully-connected structures. This helps KAN learn both the generalized compositional structure of an optimization problem and the non-linearity within each univariate function components. b) Some recent studies [30, 31] indicate that KAN often outperforms MLP in diverse machine learning tasks, considering both accuracy and efficiency.

3.1.3 Relative-Order-Aware Loss. During our preliminary experimental study, we found that naively training the proposed KAN network by the MSE regression loss function (see Eq. 8) on the training dataset D_f can not ensure stability of the subsequent MetaBBO policy learning stage. We delve into this observation to explore the tricky part behind, and found that the learning failure comes from the regression process itself. For a complex function f with diverse and intricate local landscapes [28], the regression loss is hard to be minimized to 0, which leads to the relative orders of the predicted function values of some data points in D_f violate the ground truth of their function values on f, which is also observed in recent surrogate learning studies [14, 35]. We provide an example here: suppose there are five data points in D_f, which are $[x_1, x_2, x_3, x_4, x_5]$, sorted by descending order of their true objective values $f(x)$: $[9, 5.1, 5, 3, 1]$. However, after training our KAN-based surrogate model F_θ, the predicted objective values $F_\theta(x)$ are $[9, 5, 5.1, 3, 1]$. In this case, although F_θ achieves ideal accuracy on most of the data points, it misjudges the relative order of x_2 and x_3. Although such misjudgment might not interfere classical machine learning tasks, it is indeed problematic for MetaBBO approaches. As we introduced in Section 2.1, in MetaBBO, the training of its meta-level policy relies on the correct feedback from the low-level optimization process which indicates the relative performance improvement. In the case that we substitute the original optimization problem f by our trained KAN model F_θ, the low-level optimization

process might return an incorrect relative performance improvement due to the misjudgment of the relative order of two solution points. Apparently, this might lead to potential training failure.

To address the risk of misjudgment of the relative order, we propose a novel Relative-Order-Aware (ROA) loss function, which combines the strength of the MSE regression loss to improve the model's fitting accuracy and an order correction (OC) term that forces F_θ to maintain the ground truth of the relative orders among training data. Given a batch of N_{batch} input data points sampled from D_f and sorted by their actual objective values in descending order, the proposed ROA loss function is formulated as below:

$$\mathcal{L}(\theta) = \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} [\lambda \cdot MSE(f(x_i), F_\theta(x_i)) + OC(F_\theta(x_i))]$$

$$OC(F_\theta(x_i)) = \frac{1}{2} [|f(x_{i-1}) - F_\theta(x_i)| + |F_\theta(x_i) - f(x_{i+1})|]$$

$$(9)$$

where the total loss is a weighted combination of the original MSE loss and the OC term. For the i-th data point x_i in the sorted data batch, we force the prediction value of the surrogate model, say $F_\theta(x_i)$, to locate between the ground truth objective values of x_{i-1} and x_{i+1}. The OC term is minimized only when the relative orders of the predicted values in the data batch is consistent with the ground truth value.

We present the overall training workflow of SLS in Algorithm 1. It first goes through a pre-training phase where the loss function is the naive MSE (line $1 \sim 6$). This phase aims to let the model be aware of the overall structure of the target function f. Then it goes through the second phase, where the loss function is our proposed ROA loss in Eq. 9 (line $7 \sim 14$). The combination weights λ in this phase is initially set to 1 and then decay linearly, which strikes a good balance in learning both the prediction accuracy and the relative order accuracy.

3.2 Policy Learning Stage

Our proposed Policy Learning Stage (PLS) mirrors existing MetaBBO methods [10, 25, 32, 36] which leverage RL [34] techniques for dynamic algorithm configuration (DAC) in Evolutionary Algorithms (EAs). Since this paper serves as a preliminary study on the positive effects of surrogate learning techniques in MetaBBO, we define a relatively simple MetaBBO task as the experimental operation basis, which is illustrated in Figure 2.

Specifically, a Double-DQN [40] agent is employed in the meta-level to serve as the meta-level policy π_ω, dictating flexible mutation operator configurations for the low-level DE algorithm dynamically. In the low-level optimization, given an optimization problem distribution $\mathbb{P}$, we first sample a collection of training problem instances $\mathbb{P}_{train}$ used for policy learning. For each instance $f \in \mathbb{P}_{train}$, we follow the procedure in SLS and train a surrogate model F_θ. Then these surrogate models are used as the substitution of $\mathbb{P}_{train}$ in the low-level optimization. We have to note that the overall workflow of PLS in Surr-RLDE shows consistency with what we have elaborated in Section 2.1. The only difference is that we replace the evaluation function in the low-level optimization by surrogate models, which plays two key roles in Surr-RLDE:

a) **Feature Extraction**: at each time step t of the low-level optimization, the population of the DE algorithm, X^t, is fed into surrogate model F_θ to obtain the objective values Y^t. Then a feature extractor module processes X^t and Y^t into the optimization state s^t in this time step. To ensure an accurate profiling of the current optimization state, the surrogate model need to be as accurate as possible, compared with the original function f.

b) **Reward Computation**: Once the meta-level policy π_ω receives s^t, it accordingly outputs an action a^t (in Surr-RLDE, a^t denotes different mutation operator configurations) for the low-level DE. Then DE algorithm load the dictated mutation configuration a^t and evolves X^t towards next generation X^{t+1}, with corresponding Y^{t+1} evaluated by F_θ. A reward signal in computed by a reward function, according to the relative performance improvement between Y^{t+1} and Y^t. Then the meta-level policy π_ω is trained to maximize the accumulated rewards along the low-level optimization process. To ensure the reward signal accurately reflects the effects of π_ω, the objective values predicted by F_θ should at least keep the relative order consistency. This is exactly the motivation of our proposed ROA loss.

We have to highlight here that integrating surrogate learning into MetaBBO's learning system marks a significant step forward for this domain. Intuitively, existing MetaBBO approaches often train their meta-policy for hundreds of epochs and over dozens of training problem instances, which implies massive function evaluations. In surr-RLDE, it only consumes a small number of function evaluations to construct the training dataset D_f for training the surrogate model, which greatly reduces evaluation cost. In subsequent sections, we elaborate some specific designs in PLS of Surr-RLDE.

3.2.1 State Design. In MetaBBO approaches that leverage RL as the techniques, the state representation s^t should informatively reflect the dynamic optimization status of the low-level optimization process, so as to ensure the learning effectiveness of the meta-level policy. In BBO, a common practice for featuring optimization information is Exploratory Landscape Analysis (ELA) [28], which includes six feature groups of 55 single features, each profiling a particular aspect of the optimization problem such as local-optimum properties, fitness correlation distances, convexity, etc. However, computing some of these features for each low-level optimization step in MetaBBO is impractical due to the computational complexity and requirement for large scale sampling. To address this issue, existing MetaBBO approaches lean to select an easy-to-compute subset from ELA features [36]. A recent work SYMBOL [5], proposed a novel 9-dimensional state design that shows both effectiveness and efficiency for MetaBBO. It comprises three parts: a) $s_1 \sim s_3$ compute the distributional properties of the solution population. b) $s_4 \sim s_6$ are a subset of fitness distance correlation features in ELA, reflecting the optimization convergency. c) $s_7 \sim s_9$ are some time-stamp features that indicate the pre-mature and evolution progress. We borrow this idea from SYMBOL to serve as the feature extraction module in Surr-RLDE. For detailed definition of these features, refer to the original paper.

3.2.2 Action Space Design. The action space in PLS of Surr-RLDE is a series of mutation operator configurations which combines various mutation operators and various mutation strength values. For mutation operator, we select five well-known variants: DE/rand1,

---

Algorithm 2: Policy Learning in Surr-RLDE.

Input: Surrogate models $\{F_\theta^{(k)}\}_{k=1}^{|P_{train}|}$, Initialized prediction policy π_ω, Maximum learning steps $maxLS$, Maximum function evaluations $maxFEs$ per optimization run, Update period G_{up} of target policy, DE algorithm.

Output: Optimal meta-level policy π_{ω^*}.

1   Initialize replay buffer $RM \leftarrow \emptyset$;
2   Copy parameter weights in π_ω to taget policy π_{target};
3   Initialize learning step $LS \leftarrow 0$;
4   **while** $LS \leq maxLS$ **do**
5   **for** $k \leftarrow 1$ **to** $|P_{train}|$ **do**
6   initialize generation flag $t \leftarrow 0$;
7       $X^t \leftarrow DE.initialize()$, $Y^t = F_\theta^{(k)}(X^t)$;
8       $FEs \leftarrow DE.population\_size$;
9   Extract optimization state s^t following Section 3.2.1;
10   **while** $FEs \leq maxFEs$ **do**
11        $a^t \leftarrow \pi_\omega(s^t).epsilon\_greedy()$;
12   Set DE's mutation configuration as a^t;
13        $X^{t+1} \leftarrow DE.step(X^t, Y^t)$, $Y^{t+1} \leftarrow F_\theta^{(k)}(X^{t+1})$;
14   Compute reward r^t following Section 3.2.3;
15   Extract optimization state s^{t+1} following Section 3.2.1;
16        $RM.insert(\{s^t, a^t, r^t, s^{t+1}\})$;
17   Update π_ω following Eq. 11, $LS \leftarrow LS + 1$;
18   **if** $LS\%G_{up} = 0$ **then**
19   Update π_{target} by π_ω's parameter weights;
20   **end**
21        $FEs \leftarrow FEs + DE.population\_size$, $t \leftarrow t + 1$;
22   **end**
23   **end**
24   **end**
25   **return** *The trained prediction policy* π_ω

DE/best1, DE/current-to-rand, DE/current-to-pbest and DE/current-to-best, which show diverse exploration-exploitation behavior. For mutation strength F, we provide three options: 0.1, 0.5 and 0.9. The action space A hence comprises $5 \times 3 = 15$ optional actions, each featuring a mutation variant and corresponding mutation strength. By dynamically configuring the low-level DE algorithm with desired mutation configuration at each optimization step, Surr-RLDE could improve the overall optimization performance of DE with single mutation configuration. We note that the choice of the DE algorithm is based on its common use in existing MetaBBO approaches and empirical considerations.

3.2.3 Reward Design. We adopt a simple and classic reward function in PLS of Surr-RLDE:

$$r^t = \begin{cases} 0, & if \quad y^{*,t+1} \leq y^{*,t} \\ 1, & otherwise \end{cases} \tag{10}$$

where $y^{*,t}$ denotes the objective value of the best solution found so far, evaluated by the surrogate model F_θ. A positive reward signal is returned to the meta-level policy if a better solution is found. This reward function encourages the meta-level policy to dictate desired mutation configuration for the low-level DE algorithm, hence promoting the overall optimization progress.

3.2.4 Learning Paradigm. The overall learning paradigm of PLS in Surr-RLDE is presented in Algorithm 2. We maintain a Double-DQN [40] agent as the meta-policy, which includes two neural networks: prediction policy π_ω and target policy π_{target}. The prediction policy is used to sample mutation configurations (line 11) for the low-level DE algorithm, following an *epsilon_greedy* strategy, where either the action with maximum predicted Q-values is sampled or the action in randomly chosen, with a small probability. The target policy π_{target} serve as an invariant baseline, allowing for the removal of positive bias in regressing the Q-values of the prediction policy π_ω. During the low-level optimization (line 10 ∼ 22), we use surrogate models of the training problem set P_{train} as the objective functions to evaluate solution population (line 13). For each training step (line 17), we first sample a batch of transitions from a replay memory RM. Then for each transition $\{s^t, a^t, r^t, s^{t+1}\}$ the Bellman loss function used for updating the policy π_ω is formulated as below:

$$\mathcal{L}(\omega) = \frac{1}{2}\left[[r^t + \gamma \arg\max_{a \in A} \pi_{target}(s^{t+1}, a)] - \pi_\omega(s^t, a^t)\right]^2 \quad (11)$$

where we use Q-values of the target policy as the label to regress Q-values of π_ω. The training process lasts for *maxLS* learning steps and covers all training problem instances in P_{train}. The target policy is updated by the parameter weights of the up-to-date prediction policy every G_{up} learning steps (line 18 ∼ 20). Once training ends, the learned prediction policy π_ω could be directly applied for configuring DE algorithm on solving unseen optimization problems.

4 Experimental Results

Optimization Problem Dataset. In this paper, we adopt COCO-BBOB testsuites [12] as the basic optimization problem dataset, which contains 24 synthetic optimization problems with diverse properties including unimodal or multi-modal, separable or non-separable, adequate or weak global structure, etc. For train-test-split, we randomly select 16 problem instances for training and 8 for testing. Specifically, training set P_{train} includes following functions: *Sphere, Ellipsoidal, Rastrigin, BucheRastrigin, LinearSlope, AttractiveSector, StepEllipsoidal, Rosenbrockoriginal, Rosenbrockrotated, Ellipsoidalhighcond, Discus, BentCigar, SharpRidge, DifferentPowers, RastriginF15,Schwefel*; test set P_{test} includes following functions: *Weierstrass, Schaffers, Schaffershighcond, CompositeGrierosen, Gallagher101Peaks, Gallagher21Peaks, Katsuura, LunacekbiRastrigin*. Problem dimension is set to 10, searching range locates in $[-5, 5]$. For each instance,the *maxFEs* is set to 2×10^4 and the random shift/rotation option is turned off.

Surrogate learning stage. For this learning stage, we follow Algorithm 1 to train a separate KAN model for each problem instance in P_{train}. The KAN model is implemented following pykan[1], with the layer configuration as $[10, 10, 1]$, grid size $G = 5$ and B-spline order $k = 5$. For each $f \in P_{train}$, we leverage Latin Hypercube Sampling (LHS) to sample 5×10^4 points and accordingly construct training data D_f with these sample points and corresponding objective values. During training, batch size $N_{batch} = 100$, training

epochs $T_{mse} = 300, T_{roa} = 1000$, learning rate $\eta = 0.01$. After training, we obtain 16 models, each corresponds to the surrogate model of a problem instance in P_{train}.

Policy learning stage. We follow the algorithm template in MetaBox[2] to develop policy learning pipeline in Algorithm 2. We built up a MLP network as the prediction policy π_ω in Double-DQN, specifically with 9D inputs, 15D outputs, 3 hidden layers with $[32, 64, 32]$ hidden neurons and relu activations. During training, maximum learning steps *maxLS* is set to 1.5×10^6, update period G_{up} for π_{target} is set to 1000, discount factor $\gamma = 0.99$, learning rate is set to $1e - 4$. We utilize vanilla DE as the low-level optimizer. Within our implementation of DE, initialization strategy is set to uniformly sampling, crossover strategy is set to binary crossover with $Cr = 0.7$.

Running platform. All results presented are from a platform equipped with a RTX 2080Ti 11GB GPU, an Intel Xeon E5-2680 v4 @ 56x 3.3GHZ CPU and 128 GB RAM.

Baselines. We compare Surr-RLDE with 7 baselines, which can be divided into two categories: a) **BBO methods**: Random Search, DE [7], PSO [17], SAHLPSO [37]. b) **MetaBBO methods**: DE-DDQN [32], DEDQN [36] and GLEET [25]. DE-DDQN and DEDQN are two baselines using value-based RL techniques for dynamic mutation operator selection in DE. GLEET is an up-to-date baseline with strong performance using policy-gradient RL techniques for dynamic hyper-parameter tuning in PSO. In addition, we include two variants of Surr-RLDE in following experiments: a) **Surr-RLDE-O**, which skips the surrogate learning stage and uses the original training problems P_{train} for function evaluation in policy learning stage, similar with existing MetaBBO's paradigm. b) **Surr-RLDE-MSE**, which sorely uses MSE loss (Eq. 8) instead of our proposed ROA loss (Eq. 9) for the surrogate model training in SLS. All MetaBBO baselines follow their original settings in their papers, except that the training set and maximum learning steps coincide with the setting for Surr-RLDE, Surr-RLDE-O and Surr-RLDE-MSE, which is P_{train} and 1.5×10^6 respectively.

4.1 Performance Comparison Results

4.1.1 In-Distribution Generalization. After training our Surr-RLDE and Surr-RLDE-O, and other MetaBBO baselines DE-DDQN, DEDQN and GLEET on P_{train}, we test the learned meta-level policies in these baselines and the four BBO baselines on P_{test}, where the optimization problems hold the same dimension (10) as the problems in P_{train}. We hence refer to this testing as the in-distribution generalization test. We present the average best objective values and corresponding error bars over 51 independent runs of all baselines in Table 1. We additionally show the rank of each baseline on each tested problem instance and the averaged rank for intuitive performance comparison. From the results, we can observe that:

1) **Surr-RLDE v.s. Surr-RLDE-O.** Overall, Surr-RLDE surpasses optimization performance of Surr-RLDE-O, achieving second best average rank on the test set P_{test}. This clearly demonstrate that at least in this case study, substituting the original function evaluation in the lower level of a MetaBBO method by a surrogate model of the target optimization problem does no harm to the overall mata-learning effectiveness. This is an appealing observation since as we

---

[1]https://github.com/KindXiaoming/pykan

[2]https://github.com/GMC-DRL/MetaBox

Table 1: Generalization performance of the trained Surr-RLDE and baselines on test set P_{test}

	Weierstrass		Schaffers		Schaffers_high_cond		Composite_Grie_rosen		Gallagher_101Peaks		Gallagher_21Peaks		Katsuura		Lunacek_bi_Rastrigin		Avg Rank
	mean (std)	rank	mean (std)	rank	mean (std)	rank	mean (std)	rank	mean (std)	rank	mean (std)	rank	mean (std)	rank	mean (std)	rank	
Surr-RLDE	6.914E+00 (±1.603E+00)	5	4.330E-02 (±3.622E-02)	3	2.232E-01 (±1.007E-01)	3	1.769E+00 (±3.189E-01)	3	3.448E-07 (±1.071E-06)	1	1.972E-05 (±1.277E-04)	1	1.458E+00 (±3.026E-01)	4	4.245E+01 (±9.243E+00)	4	3
Surr-RLDE-O	2.173E+00 (±2.443E+00)	2	8.484E-02 (±5.639E-02)	4	4.270E-01 (±2.730E-01)	4	1.427E+00 (±4.405E-01)	2	1.719E+00 (±1.812E+00)	5	2.908E+00 (±3.250E+00)	3	1.417E+00 (±2.865E-01)	3	4.525E+01 (±7.022E+00)	5	3.5
DE_DDQN	4.173E+00 (±2.292E+00)	3	2.424E-04 (±3.196E-04)	1	8.645E-04 (±8.104E-04)	1	2.336E+00 (±4.484E-01)	5	7.973E-01 (±1.769E+00)	4	3.530E+00 (±2.863E+00)	4	1.479E+00 (±2.262E-01)	6	4.014E+01 (±6.514E+00)	3	3.375
DE_DQN	2.108E+01 (±4.650E+00)	9	7.181E+00 (±1.265E+00)	9	2.550E+01 (±4.107E+00)	9	1.226E+01 (±2.320E+00)	9	4.484E+01 (±1.103E+01)	9	5.847E+01 (±9.404E+00)	9	3.473E+00 (±8.317E-01)	9	1.586E+02 (±2.128E+01)	9	9
GLEET	4.944E-01 (±4.408E-01)	1	1.031E-01 (±7.516E-02)	5	4.965E-01 (±4.083E-01)	5	1.801E+00 (±5.846E-01)	4	2.330E-01 (±6.705E-01)	3	9.022E-01 (±2.035E+00)	2	1.216E+00 (±3.707E-01)	1	2.995E+01 (±9.255E+00)	1	2.75
DE	6.996E+00 (±1.921E+00)	6	2.438E-03 (±9.000E-03)	2	3.110E-02 (±4.503E-02)	2	1.109E+00 (±3.204E-01)	1	8.647E-02 (±3.902E-01)	2	5.600E+00 (±2.559E-01)	6	1.554E+00 (±2.255E-01)	8	3.915E+01 (±4.472E+00)	2	3.625
PSO	6.439E+00 (±1.817E+00)	4	1.533E+00 (±2.072E-01)	6	5.630E+00 (±9.901E-01)	6	3.104E+00 (±6.871E-01)	6	1.954E+00 (±2.664E+00)	6	4.513E+00 (±4.161E+00)	5	1.460E+00 (±2.711E-01)	5	5.948E+01 (±7.078E+00)	6	5.5
SAHLPSO	8.047E+00 (±5.463E+00)	7	2.281E+00 (±1.006E+00)	7	8.243E+00 (±3.467E+00)	7	5.267E+00 (±1.184E+00)	7	2.274E+00 (±4.457E+00)	7	6.643E+00 (±8.720E+00)	7	1.380E+00 (±5.432E-01)	2	8.842E+01 (±1.761E+01)	7	6.375
Random Search	1.005E+01 (±1.403E+00)	8	3.974E+00 (±5.344E-01)	8	1.406E+01 (±2.263E+00)	8	6.592E+00 (±9.522E-01)	8	1.330E+01 (±3.896E+00)	8	1.753E+01 (±5.886E+00)	8	1.482E+00 (±2.483E-01)	7	1.081E+02 (±9.746E+00)	8	7.875

(a) *Gallagher 21Peaks* 10D-S/R   (b) *Gallagher 101Peaks* 10D-S/R   (c) *Gallagher 21Peaks* 30D   (d) *Schaffers high cond* 30D
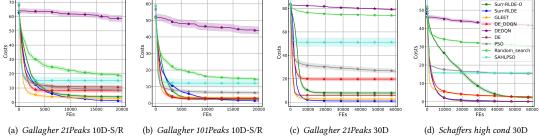
Figure 3: Generalization performance of the trained Surr-RLDE and baselines on out-of-distribution optimization problems.

elaborate in Section 3.2, incorporating MetaBBO method with surrogate model in its low-level optimization could reduce the massive number of function evaluations needed to sample trajectory and train the meta-level policy. A particular interesting observation we would like to mention here is: the first four problems in Table 1 are multimodal problems with clear global structure, where our Surr-RLDE is only comparable with Surr-RLDE-O. However, when we look at the results for the last four problems which are multimodal problem with unclear global structure, Surr-RLDE show superior optimization performance. A possible explanation is that although the surrogate model show certain prediction error for the landscape of the target problem, this might be beneficial for problems with complex landscape. In this case our Kan-based surrogate serves as a simplification of the original landscape and provides Surr-RLDE a shortcut to locate optima. Further studies on this point is needed and we mark it as an important future work.

2) **Surr-RLDE v.s. MetaBBO baselines.** GLEET as an up-to-date strong MetaBBO baseline achieves best performance against all baselines and our Surr-RLDE slightly underperforms it, with comparable even superior generalization performance on five of all eight tested problems. This further validate the effectiveness of the surrogate learning stage. By learning the landscape via the novel ROA loss we proposed, we ensure that the approximated landscape, especially the relative order information across different solutions is almost accurate and is enough to provide correct reward signal to assist the training of the meta-level policy. Our Surr-RLDE also outperforms DE-DDQN and DEDQN which employ similar value-based RL technique for dynamic algorithm configuration in

DE algorithm. In particular, we observe that DE-DDQN achieve best performance on *Schffers* and *Schffershighcond* which are multimodal problems with global structure. Surr-RLDE achives best performance on *Gallagher101Peaks* and *Gallagher21Peaks* which are multimodal problems with weak global structure. This might indicate that Surr-RLDE shows certain advantages on solving more complex problems since De-DDQN's configuration space (operator selection among several mutation operators) is much smaller than Surr-RLDE (mutation configurations including both operator selection and parameter tuning).

4.1.2 Out-Of-Distribution Generalization. For MetaBBO approach, a key performance metric to measure its usefulness is to measure the generalization of this approach towards those optimization problems which hold different optimization properties than the problem instances used for training [26]. This is so called out-of-distribution generalization aspect of a MetaBBO approach. In this experiment section, we compare such performance aspect of Surr-RLDE and other baselines. Specifically, we alternate properties of problems in the test set P_{test} by adding random shift/rotation on decision variables or expanding its dimension from 10 to 30. We denote these two cases as "10D-S/R" and "30D" respectively. For MetaBBO baselines including Surr-RLDE, we directly use their models trained on the training set P_{train} to the problems with modified properties. We only illustrate the optimization curves (with error bars over 51 independent runs) of all baselines on four of the tested problems in Figure 3 due to the space limitation. All results show that Surr-RLDE presents robust generalization performance on more complex
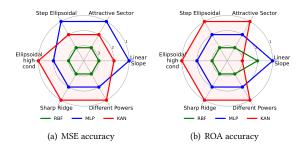
(a) MSE accuracy

(b) ROA accuracy

Figure 4: Performance comparison of three ANN models.

and unseen problems. This further underscore the effectiveness of integrating well-trained surrogate model into low-level function evaluation process MetaBBO.

4.2 Ablation Studies

4.2.1 Surrogate Model Choices. We have to note that the selection of KAN as the surrogate model is non-trivial. In literature related with surrogate learning, we found that the dominating model choices stay in-between MLP and RBF networks, each of which excels in some applications. However, since MetaBBO is a novel task and KAN is an emerging network architecture, existing literature barely discuss these three models' performance differences and technical (dis-) advantages. Hence we conduct a simple and intuitive comparison for these three models, where we train KAN, MLP and RBF as surrogate models for each problem instance in P_{train} and illustrate the MSE loss rank and ROA loss rank of them on several problem instances in Figure 4. The results based on 200 independent repetitions show that: a) RBF network as a kernel-based (e.g., Gaussian kernel) regression model could not achieve good prediction performance on the global landscape structure of complex optimization problem. b) In terms of MSE accuracy, MLP and KAN show similar prediction performance. c) However, KAN holds high-level ROA accuracy, which indicates that the spline-based univariate regression in KAN might be helpful to maintain local landscape, especially the relative order.

4.2.2 Relative-Order-Aware Loss. Now we have determined the most suitable surrogate model design for Surr-RLDE: KAN architecture. We further conduct two studies on the effectiveness of the proposed ROA loss compared to naive MSE regression loss. We first visualize some qualitative evidence in Figure 5, where we present two toy problems: *Rosenbrock original* 2D and *Schwefel* 2D as examples to show the landscapes predicted by the MSE-trained KAN model and the ROA-trained KAN model. The sample size for training KAN model on these two 2D problems is 10000. We observe that the model trained by ROA loss could ensure the smoothness on the valley of two toy problems, which the model trained by MSE could not. This indicates that our proposed ROA loss, which adds a order correction term to the ordinary MSE loss, could improve the prediction accuracy by maintaining the local landscape structures. In contrast, MSE loss might only contribute to the global landscape fitting of the target problem. We further meta-train our Surr-RLDE and Surr-RLDE-MSE on P_{train} and test the trained policies on the eight problems in P_{test}. The optimization results over 51 runs are
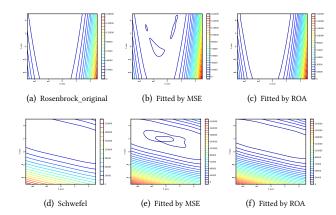

(a) Rosenbrock_original

(b) Fitted by MSE

(c) Fitted by ROA

(d) Schwefel

(e) Fitted by MSE

(f) Fitted by ROA

Figure 5: Landscape accuracy of MSE and ROA loss.

Table 2: Optimization performance comparison results of Surr-RLDE and Surr-RLDE-MSE on P_{test}

	Weierstrass	Schaffers	Schaffers high cond	Composite Grie rosen
	mean std	mean std	mean std	mean std
Surr-RLDE	6.914E+00 (±1.603E+00)	**4.330E-02** (±3.622E-02)	**2.232E-01** (±1.007E-01)	**1.769E+00** (±3.189E-01)
Surr-RLDE-MSE	**3.161E+00** (±2.187E+00)	2.936E+00 (±9.212E-01)	8.653E+00 (±3.756E+00)	1.791E+00 (±9.466E-01)
	Gallagher 101Peaks	Gallagher 21Peaks	Katsuura	Lunacek bi Rastrigin
Surr-RLDE	**3.448E-07** (±1.071E-06)	**1.972E-05** (±1.277E-04)	1.458E+00 (±3.026E-01)	**4.245E+01** (±9.243E+00)
Surr-RLDE-MSE	5.234E+00 (±3.316E+00)	1.217E+01 (±1.148E+01)	**5.576E-01** (±2.321E-01)	5.725E+01 (±1.611E+01)

presented in Table 2 and demonstrate that using surrogate models trained by ROA loss could provide more accurate reward signal than those trained by ordinary MSE loss.

5 Conclusion

In this paper, we conduct a preliminary study on feasibility of combining surrogate model learning with MetaBBO to achieve both effectiveness and efficiency. Specifically, we first identify the core limitation of ordinary MSE regression loss within surrogate learning: inability for maintaining accurate local landscape structure. To address this limitation, on the one hand, we adopt a novel neural network architecture KAN as the surrogate model which improves the surrogate's accuracy through spline-based univariate function composition. On the other hand, we design a relative-order-aware loss which forces the KAN model to learn the local landscape details. To verify if the proposed surrogate learning can assist MetaBBO, we design a simple MetaBBO task where the meta-level RL policy dynamically configures the low-level DE algorithm. Through comprehensive benchmarking, the feasibility of integrating the surrogate model into MetaBBO is sufficiently validated, including the contribution of both model selection and loss design. Surr-RLDE denotes a pivotal step towards MetaBBO with high efficiency, especially tailored for expensive, hard-to-evaluate BBO problems.

Acknowledgments

References

[1] Stéphane Alarie, Charles Audet, Aïmen E Gheribi, Michael Kokkolaras, and Sébastien Le Digabel. 2021. Two decades of blackbox optimization applications. *EURO Journal on Computational Optimization* (2021).

[2] Reza Alizadeh, Janet K Allen, and Farrokh Mistree. 2020. Managing computational complexity using surrogate models: a critical review. *Research in Engineering Design* 31, 3 (2020), 275–298.

[3] Claudio Angione, Eric Silverman, and Elisabeth Yaneske. 2022. Using machine learning as a surrogate model for agent-based simulations. *Plos one* 17, 2 (2022), e0263150.

[4] Tanmoy Chatterjee, Souvik Chakraborty, and Rajib Chowdhury. 2019. A critical review of surrogate assisted robust design optimization. *Archives of Computational Methods in Engineering* 26 (2019), 245–274.

[5] Jiacheng Chen, Zeyuan Ma, Hongshu Guo, Yining Ma, Jie Zhang, and Yue-Jiao Gong. 2024. SYMBOL: Generating Flexible Black-Box Optimizers through Symbolic Equation Learning. In *The Twelfth International Conference on Learning Representations.* https://openreview.net/forum?id=vLJcd43U7a

[6] Ziwen Chen, Gundavarapu, and WU DI. 2024. Vision-KAN: Exploring the Possibility of KAN Replacing MLP in Vision Transformer. https://github.com/chenziwenhaoshuai/Vision-KAN.git.

[7] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. 2010. Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation* 15, 1 (2010), 4–31.

[8] Jan Drgona, Aaron Tuor, James Koch, Madelyn Shapiro, Bruno Jacob, and Draguna Vrabie. 2023. NeuroMANCER: Neural Modules with Adaptive Nonlinear Constraints and Efficient Regularizations. (2023). https://github.com/pnnl/neuromancer

[9] Hongshu Guo, Yining Ma, Zeyuan Ma, Jiacheng Chen, Xinglin Zhang, Zhiguang Cao, Jun Zhang, and Yue-Jiao Gong. 2024. Deep Reinforcement Learning for Dynamic Algorithm Selection: A Proof-of-Principle Study on Differential Evolution. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2024).

[10] Hongshu Guo, Zeyuan Ma, Jiacheng Chen, Yining Ma, Zhiguang Cao, Xinglin Zhang, and Yue-Jiao Gong. 2024. ConfigX: Modular Configuration for Evolutionary Algorithms via Multitask Reinforcement Learning. *arXiv preprint arXiv:2412.07507* (2024).

[11] Nikolaus Hansen. 2016. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772* (2016).

[12] Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. 2010. *Real-parameter black-box optimization benchmarking 2010: Experimental setup.* Ph. D. Dissertation. INRIA.

[13] Hao Hao, Xiaoqun Zhang, Bingdong Li, and Aimin Zhou. 2024. A First Look at Kolmogorov-Arnold Networks in Surrogate-assisted Evolutionary Algorithms. *arXiv preprint arXiv:2405.16494* (2024).

[14] Hao-Gan Huang and Yue-Jiao Gong. 2022. Contrastive learning: An alternative surrogate for offline data-driven evolutionary computation. *IEEE Transactions on Evolutionary Computation* 27, 2 (2022), 370–384.

[15] Yaochu Jin, Handing Wang, Tinkle Chugh, Dan Guo, and Kaisa Miettinen. 2018. Data-driven evolutionary optimization: An overview and case studies. *IEEE Transactions on Evolutionary Computation* 23, 3 (2018), 442–458.

[16] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.

[17] James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, Vol. 4. ieee, 1942–1948.

[18] Sun Hye Kim and Fani Boukouvala. 2020. Machine learning-based surrogate modeling for data-driven optimization: a comparison of subset selection for regression techniques. *Optimization Letters* 14, 4 (2020), 989–1010.

[19] Slawomir Koziel, David Echeverría Ciaurri, and Leifur Leifsson. 2011. Surrogate-based methods. *Computational optimization, methods and algorithms* (2011), 33–59.

[20] Robert Lange, Tom Schaul, Yutian Chen, Chris Lu, Tom Zahavy, Valentin Dalibard, and Sebastian Flennerhag. 2023. Discovering attention-based genetic algorithms via meta-black-box optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference.* 929–937.

[21] Robert Lange, Tom Schaul, Yutian Chen, Tom Zahavy, Valentin Dalibard, Chris Lu, Satinder Singh, and Sebastian Flennerhag. 2023. Discovering evolution strategies via meta-black-box optimization. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation.* 29–30.

[22] Xiaobin Li, Kai Wu, Yujian Betterest Li, Xiaoyu Zhang, Handing Wang, and Jing Liu. 2024. Pretrained Optimization Model for Zero-Shot Black Box Optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems.*

[23] Xiaobin Li, Kai Wu, Xiaoyu Zhang, Handing Wang, and Jing Liu. 2023. B2Opt: Learning to Optimize Black-box Optimization with Little Budget. *arXiv preprint arXiv:2304.11787* (2023).

[24] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, and Max Tegmark. 2025. KAN: Kolmogorov–Arnold Networks. In *The Thirteenth International Conference on Learning Representations.* https://openreview.net/forum?id=Ozo7qJ5vZi

[25] Zeyuan Ma, Jiacheng Chen, Hongshu Guo, Yining Ma, and Yue-Jiao Gong. 2024. Auto-configuring exploration-exploitation tradeoff in evolutionary computation via deep reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference.* 1497–1505.

[26] Zeyuan Ma, Hongshu Guo, Jiacheng Chen, Zhenrui Li, Guojun Peng, Yue-Jiao Gong, Yining Ma, and Zhiguang Cao. 2024. MetaBox: a benchmark platform for meta-black-box optimization with reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2024).

[27] Zeyuan Ma, Hongshu Guo, Yue-Jiao Gong, Jun Zhang, and Kay Chen Tan. 2024. Toward Automated Algorithm Design: A Survey and Practical Guide to Meta-Black-Box-Optimization. *arXiv preprint arXiv:2411.00625* (2024).

[28] 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.* 829–836.

[29] Ali Wagdy Mohamed, Karam M Sallam, Prachi Agrawal, Anas A Hadi, and Ali Khater Mohamed. 2023. Evaluating the performance of meta-heuristic algorithms on CEC 2021 benchmark problems. *Neural Computing and Applications* 35, 2 (2023), 1493–1517.

[30] Yanhong Peng, Yuxin Wang, Fangchao Hu, Miao He, Zebing Mao, Xia Huang, and Jun Ding. 2024. Predictive modeling of flexible EHD pumps using Kolmogorov–Arnold Networks. *Biomimetic Intelligence and Robotics* (2024), 100184. doi:10.1016/j.birob.2024.100184

[31] Moein E Samadi, Younes Müller, and Andreas Schuppert. 2024. Smooth Kolmogorov Arnold networks enabling structural knowledge representation. *arXiv preprint arXiv:2405.11318* (2024).

[32] Mudita Sharma, Alexandros Komninos, Manuel López-Ibáñez, and Dimitar Kazakov. 2019. Deep reinforcement learning based parameter control in differential evolution. In *Proceedings of the genetic and evolutionary computation conference.* 709–717.

[33] Jianyong Sun, Xin Liu, Thomas Bäck, and Zongben Xu. 2021. Learning adaptive differential evolution algorithm from optimization experiences by policy gradient. *IEEE Transactions on Evolutionary Computation* 25, 4 (2021), 666–680.

[34] Richard S Sutton. 2018. Reinforcement learning: An introduction. *A Bradford Book* (2018).

[35] Rong-Xi Tan, Ke Xue, Shen-Huan Lyu, Haopu Shang, yaowang, Yaoyuan Wang, Fu Sheng, and Chao Qian. 2025. Offline Model-Based Optimization by Learning to Rank. In *The Thirteenth International Conference on Learning Representations.* https://openreview.net/forum?id=sb1HgVDLjN

[36] Zhiping Tan and Kangshun Li. 2021. Differential evolution with mixed mutation strategy based on deep reinforcement learning. *Applied Soft Computing* 111 (2021), 107678.

[37] Xinmin Tao, Xiangke Li, Wei Chen, Tian Liang, Yetong Li, Jie Guo, and Lin Qi. 2021. Self-adaptive two roles hybrid learning strategies-based particle swarm optimization. *Information Sciences* 578 (2021), 457–481.

[38] Virginia Torczon and Michael Trosset. 1998. Using approximations to accelerate engineering design optimization. In *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* 4800.

[39] Rohit K Tripathy and Ilias Bilionis. 2018. Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of computational physics* 375 (2018), 565–588.

[40] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[41] Christopher Williams and Carl Rasmussen. 1995. Gaussian processes for regression. *Advances in neural information processing systems* 8 (1995).

[42] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. 2016. Two-stage transfer surrogate model for automatic hyperparameter optimization. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16.* Springer, 199–214.

[43] Xu Yang, Rui Wang, Kaiwen Li, and Hisao Ishibuchi. 2025. Meta-Black-Box optimization for evolutionary algorithms: Review and perspective. *Swarm and Evolutionary Computation* 93 (2025), 101838.

[44] Runpeng Yu, Weihao Yu, and Xinchao Wang. 2024. Kan or mlp: A fairer comparison. *arXiv preprint arXiv:2407.16674* (2024).