Degradation-Resistant Offline Optimization via Accumulative Risk Control

Huakang Lu[a], Hong Qian[a;*], Yupeng Wu[a], Ziqi Liu[c], Ya-Lin Zhang[c], Aimin Zhou[a] and Yang Yu[b]

[a]Shanghai Institute of AI for Education and School of Computer Science and Technology, East China Normal University, Shanghai 200062, China
[b]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[c]Ant Group, Hangzhou 310023, China

Abstract. Offline optimization aims to elaborately construct a solution that optimizes a black-box function with only access to the offline dataset. A typical manner of constructing the solution is to train a surrogate model of the black-box function on the offline dataset and optimize the solution guided by the surrogate model. However, this manner often encounters a fundamental challenge that the surrogate model could erroneously estimate out-of-distribution (OOD) solutions. Therefore, the optimizer would be misled to produce inferior solutions for online applications, i.e., degradation of performance. To this end, this paper formalizes the risk of degradation for OOD solutions and proposes an **a**ccumulative **r**isk **c**ontrolled **o**ffline **o**ptimization (ARCOO) method. Specifically, ARCOO learns a surrogate model in conjunction with an energy model. The energy model characterizes the risk of degradation by learning on high-risk solutions and low-risk ones contrastively. In the optimization procedure, the behavior of the optimizer in each step is controlled by a risk suppression factor calculated via the energy model, which leads to the controllable accumulative risk. Theoretically, we justify the efficacy of energy for accumulative risk control. Extensive experiments on offline optimization tasks show that ARCOO surpasses state-of-the-art methods in both degradation-resistance and optimality of the output solution.

1 Introduction

Black-box optimization has been developed and applied in a wide range of disciplines, yet the widespread adoption of it in real-world domains has been hampered by the demand for frequent evaluations. In many tasks, the objective function evaluation is perilous, expensive, or even infeasible, such as the design of trauma system [29], blast furnace [38] and materials [20]. Despite the inaccessibility to actively evaluating new solutions for an optimization algorithm, the historical data in the form of solution and the corresponding function value pair is available. Offline optimization, without actively querying the solutions online, aims to make full use of the offline data and finally recommend a potential high-quality solution to be applied online.

Based on the static offline dataset, a typical way of offline optimization is to learn a predictive model to surrogate the black-box function and guide the optimizer to search for the best possible solutions for online applications. Unfortunately, the quality of solutions found in the optimization procedure has a risk of degrading dramatically because of the overestimation error in the surrogate model [18]. A well-trained surrogate model provides accurate predictions for the solutions in the vicinity of offline data, but it can make erroneous predictions for OOD solutions [28, 3, 23]. Due to the exploration nature of global optimization, optimizers tend to try unseen OOD solutions for potentially better performance (i.e., pursue much better solutions than the best one in the offline dataset). As the optimization procedure proceeds, the risk of performance degradation increases, and the quality of the found solutions could continue to decline. Without active evaluation feedback, it is difficult to determine when to stop the optimization procedure and return the final solution. Eventually, the optimizers could be easily driven towards solutions with high estimated quality but turn out to be poor in the online application. In a nutshell, the issue of performance degradation becomes particularly critical.

Related Work. To address the issue of performance degradation, recent studies propose various approaches. Bayesian optimization (BO) [31, 25] under the offline scenario explicitly assesses uncertainty by the Gaussian process surrogate model and leverages it in the acquisition function to alleviate the performance degradation in optimization. Although the latest BO studies have made significant progress in addressing the curse of dimensionality under certain conditions [26, 36, 4], they still have difficulties in addressing large-scale offline data. Some existing methods propose to sample solutions from learned generative models and introduce regularization in the sampling process. MIN [18] learns an inverse mapping using generative adversarial network [11] from output values to input solutions and generates new solutions by this mapping according to a reliable score. CbAS [2] and Autofocus [7] use variational auto-encoder (VAE) [17] to model a distribution over the solution space and adapt the VAE-based generative model to the optimal solutions within a trust region. As a result, output solutions are sampled within an acceptable extent of uncertainty. The regularization in generative model methods intends to evade sampling OOD solutions and mitigate the performance degradation issue. However, the applications of generative model based methods often require elaborate tuning across different tasks.

Benefiting from the powerful learning ability, predictive models such as deep neural networks, are served as surrogate models in recent work. DDEA-SE [30] applies selective ensemble methods to perceive uncertainty from multiple surrogate predictions and prevent the evolutionary optimizer [14, 15] from falling into overestimated OOD solutions. BDI [3] develops a data distillation [21] technique that backward maps the searched solutions to offline data and forces alignment between them. Besides, several studies propose to address

* Corresponding Author. E-mail: hqian@cs.ecnu.edu.cn.

the performance degradation issue by imposing different priors on the surrogate model. In this way, the surrogate model can provide predictions for OOD solutions with a penalizing bias and lead the optimizer to elude highly overestimated solutions. COMs [28] proposes a conservative model training method to depress the prediction of adversarial solutions. RoMA [35] uses the local smoothness prior to overcome the non-smooth nature of the surrogate deep neural network models in order to realize conservatism. NEMO [8] leverages the normalized maximum likelihood estimation to be aware of the uncertainty in the conservative surrogate model. By formalizing offline optimization as domain adaptation [37], IOM [23] enforces invariant representation in the surrogate model and makes mediocre predictions for OOD solutions. By involving conservatism in surrogate models, the conservative methods intentionally lower the predictions of overestimated OOD solutions that are likely to provoke performance degradation. However, surrogate models could occasionally be too conservative, resulting in little improvement of the output solution compared with the best one in the offline dataset.

Motivation and Contribution. Although unseen OOD solutions inevitably raise the risk of performance degradation in optimization, they are desired for their potentially better performance than the historical offline data. One way to realize such a tradeoff is that we need to optimize for solutions under a certain extent of risk control. Different from previous methods that control the risk of degradation in the optimization procedure implicitly via conservative models, a flexible method that restricts the risk by directly controlling it in the optimizer is appealing. Through this method, it is expected that, if the accumulative risk of the optimization procedure is bounded, the optimizer could be degradation-resistant and the searched solutions could always be safe for online applications.

Driven by this motivation, this paper proposes an **a**ccumulative **r**isk **c**ontrolled **o**ffline **o**ptimization (ARCOO) method based on the energy model to explicitly characterize the risk of solutions and control the accumulative risk in the optimization procedure. We first define the risk of solutions incurring performance degradation. Based on that, in ARCOO, a dual-head model is developed to not only learn the surrogate prediction but also explicitly characterize risk via energy. Specifically, we theoretically analyze the training behavior and demonstrate that the learned energy is an effective indicator of risk. The output energy of the developed dual-head model is transformed into a risk suppression factor in each step, which is proved to be upper bounded. The risk suppression factor is used to optimize the solution along with the prediction in each step of the optimization procedure. The accumulative risk in the whole optimization procedure is controlled under a certain extent induced by the risk suppression factor. The behavior of ARCOO is flexibly and adaptively controlled according to the risk. The efficacy of ARCOO is verified on offline optimization tasks such as drug discovery, material invention, and robotic design. Extensive experiment results show that ARCOO surpasses the state-of-the-art (SOTA) methods with respect to both degradation resistance and optimality of the output solution. Notably, ARCOO achieves an average 1.52 times improvement to the best solution in the offline dataset across all tasks, which is the highest.

The subsequent sections respectively present the problem formalization, introduce the proposed ARCOO method, show the empirical results, and finally conclude the paper.

2 Problem Formalization

Offline Optimization. Given a black-box objective function $f(\boldsymbol{x})$ and an offline dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ with $y_i = f(\boldsymbol{x}_i)$, the goal of offline optimization is to find a solution $\boldsymbol{x}_{\text{app}}$ in the sololution space $\mathcal{X}$ to approximate $\arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$ for the online application. Since $f(\boldsymbol{x})$ is not accessible in this scenario, a practical way to enable offline optimization is to learn a parameterized surrogate model $\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ of the objective function $f(\boldsymbol{x})$ based on the offline dataset $\mathcal{D}$, i.e., $\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta})$, where $\boldsymbol{\theta} \in \Theta$ is the surrogate model parameter and $\mathcal{L}(\cdot)$ is the loss function (e.g., mean squared error). The trained surrogate model $\widehat{f}_{\boldsymbol{\theta}^\star}(\boldsymbol{x})$ (hereinafter referred to as $\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$) is then used for providing surrogate evaluations and driving the optimizer to find solutions with superior performance. The optimizer iteratively updates the solutions (e.g., gradient ascent) and sets a certain time step T as the terminating condition to output the solution $\boldsymbol{x}_T$ as $\boldsymbol{x}_{\text{app}}$ for the online application. That is to say,

$$\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} + \nabla_{\boldsymbol{x}} \widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})|\boldsymbol{x} = \boldsymbol{x}_{t-1},\ t = 1, ..., T;\ \boldsymbol{x}_T = \boldsymbol{x}_{\text{app}}. \quad (1)$$

Challege. The terminating condition time step T is nontrivial to set. If reliable predictions could be provided for all possible input solutions, an ideal T should be set to a large enough number. However, since the surrogate model is learned from $\mathcal{D}$, the generalization ability is limited to the coverage of the dataset in the solutions space. Considering that the surrogate model is prone to produce inaccurate predictions for OOD solutions, the optimizer would be continuously misled to search for overestimated solutions and bring performance degradation to the optimization procedure. In this respect, a large T inevitably suffers from performance degradation, yet a small T leads to insufficient optimization. This paper proposes that a degradation-resistant optimization procedure allows large T and risk control is an effective way to realize it.

Definition of Risk. The *risk* of a solution is defined as the possibility to provoke performance degradation in the optimization procedure. As discussed above, performance degradation often emerges when the surrogate model makes erroneous overestimations on OOD solutions. Therefore, the OOD solutions that provoke large prediction errors (i.e., $|\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - f(\boldsymbol{x})|$) are high risk, and the solutions $\boldsymbol{x} \in \mathcal{D}$ are low risk since $\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is trained on $\mathcal{D}$ and make relatively accurate predictions on these solutions. Consider $\mathcal{P}$ as an empirical distribution over the offline dataset with probability density function $p(\boldsymbol{x}) = \Sigma_{i=1}^N \delta_{\boldsymbol{x}=\boldsymbol{x}_i}, \boldsymbol{x}_i \in \mathcal{D}$, where δ is a Dirac delta function that smooths the discrete offline dataset into a continuous offline data distribution [28], and let $S(\mathcal{P}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[f(\boldsymbol{x})]$, $\widehat{S}_{\boldsymbol{\theta}}(\mathcal{P}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})]$.

Based on the discussion above, for a well-trained model parameter $\boldsymbol{\theta}$, it is reasonable to assume that there exists $\varepsilon_{\boldsymbol{\theta}}$ such that $|\widehat{S}_{\boldsymbol{\theta}}(\mathcal{P}) - S(\mathcal{P})| \leq \varepsilon_{\boldsymbol{\theta}}$. It implies that the surrogate model makes limited prediction error on $\mathcal{P}$, and $\mathcal{P}$ is a low-risk distribution. This inspires us to define $\mathcal{Q}$ as a distribution over OOD solutions satisfying $|\widehat{S}_{\boldsymbol{\theta}}(\mathcal{Q}) - S(\mathcal{Q})| > \varepsilon_{\boldsymbol{\theta}}$. Herein $\mathcal{Q}$ is a high-risk distribution due to the large prediction error produced by the surrogate model. Let $q(\boldsymbol{x})$ denote the probability density function of $\mathcal{Q}$. A solution $\boldsymbol{x}$ with high density $q(\boldsymbol{x})$ is also high risk.

3 The Proposed Method

This section presents ARCOO, which is an energy model based risk-control offline optimization method. An illustration of ARCOO is shown in Figure 1. Typical offline optimization methods utilize supervised learning approaches to build a surrogate model and produce predictions of input solutions. In order to explicitly model the risk of solutions, ARCOO additionally applies a self-supervised energy model and regards the learned energy as an estimation of risk. Instead of training separate models, ARCOO applies a dual-head neural

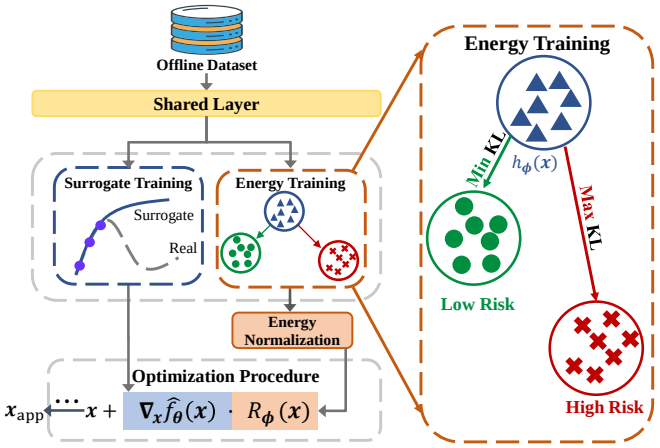


Figure 1. An overview of ARCOO. The dual-head model first learns the surrogate prediction and risk indicator from the offline dataset. For the surrogate head, supervised learning is applied to fit the unknown objective function. For the energy head, a self-supervised energy model drives the modeled distribution to approximate low-risk solutions distribution and drift away from high-risk solutions distribution sampled by Langevin dynamics. In the optimization procedure, the output energy is normalized into a risk suppression factor to guide the behavior of an optimizer together with the surrogate prediction under risk control. ARCOO finally outputs a high-quality solution for the online application.

network and desires heterogeneous output of two different heads, terms surrogate head $\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and energy head $E_{\boldsymbol{\phi}}(\boldsymbol{x})$ respectively. For $\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$, similar to most offline optimization methods, a supervised learning procedure equipped with mean squared error (MSE) loss is used to fit the actual unknown objective function, i.e., minimizing $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i)^2$ where $(\boldsymbol{x}_i, y_i) \in \mathcal{D}$. In the optimization procedure, a risk suppression factor is first calculated from energy to indicate the risk of the input solution. Then, a gradient ascent optimizer is driven by not only the surrogate prediction but also the risk suppression factor to search for solutions with preferable surrogate values while preventing the optimization from performance degradation. The procedure of ARCOO is shown in Algorithm 1.

In the rest of this section, we first perform the contrastive training strategy of energy based model and show that energy is an efficient risk indicator, then demonstrate how the information from the surrogate and energy model is used to guide the optimizer.

3.1 *Modeling Risk with Energy*

Based on the definition of risk in Section 2, a straightforward way to assess the risk of OOD solutions is learning to model a distribution $h_{\boldsymbol{\phi}}(\boldsymbol{x})$ to approximate $p(\boldsymbol{x})$ of the low-risk distribution. We employ the energy based model (EBM) [19, 5] to characterize the risk explicitly. EBM represents the likelihood of a probability distribution for solution $\boldsymbol{x}$ as

$$h_{\boldsymbol{\phi}}(\boldsymbol{x}) = \frac{\exp(-E_{\boldsymbol{\phi}}(\boldsymbol{x}))}{Z(\boldsymbol{\phi})}, \tag{2}$$

where $Z(\boldsymbol{\phi}) = \int_{\mathcal{X}} \exp(-E_{\boldsymbol{\phi}}(\boldsymbol{x}))\mathrm{d}\boldsymbol{x}$ is the partition function, $E_{\boldsymbol{\phi}}(\boldsymbol{x})$ denotes the energy function, and $\boldsymbol{\phi} \in \Phi$ is the energy model parameters. As the main building block in EBM, the energy function is a mapping from input to a scalar, i.e., $E_{\boldsymbol{\phi}}\colon \mathbb{R}^N \to \mathbb{R}$, and thus it can be represented by a neural network that takes solution $\boldsymbol{x} \in \mathbb{R}^N$ as input and outputs energy $E_{\boldsymbol{\phi}}(\boldsymbol{x}) \in \mathbb{R}$. The ability to identify low-risk and high-risk solutions is desired in energy model training. To realize that, we apply Contrastive Divergence (CD) [13] to train EBM,

$$\mathcal{L}_{\mathrm{CD}}(\boldsymbol{\phi}) = \mathrm{KL}(p(\boldsymbol{x})\|h_{\boldsymbol{\phi}}(\boldsymbol{x})) - \mathrm{KL}(q(\boldsymbol{x})\|h_{\boldsymbol{\phi}}(\boldsymbol{x})), \tag{3}$$

where $KL(\cdot||\cdot)$ denotes the Kullback-Leibler (KL) divergence of two given distributions. As shown in Figure 1, on the one hand, $\mathcal{L}_{\mathrm{CD}}(\boldsymbol{\phi})$ minimizes the divergence between $p(\boldsymbol{x})$ and $h_{\boldsymbol{\phi}}(\boldsymbol{x})$, driving $h_{\boldsymbol{\phi}}(\boldsymbol{x})$ to approximate $p(\boldsymbol{x})$. On the other hand, the divergence between $q(\boldsymbol{x})$ and $h_{\boldsymbol{\phi}}(\boldsymbol{x})$ is maximized in $\mathcal{L}_{\mathrm{CD}}(\boldsymbol{\phi})$, pushing $h_{\boldsymbol{\phi}}(\boldsymbol{x})$ away from $q(\boldsymbol{x})$. For any given input $\boldsymbol{x}$, a high probability density $h_{\boldsymbol{\phi}}(\boldsymbol{x})$ indicates that $p(\boldsymbol{x})$ is also high, meanings that $\boldsymbol{x}$ is likely to be a low-risk solution, and vice versa.

Although $h_{\boldsymbol{\phi}}(\boldsymbol{x})$ can evaluate the risk of solutions, it cannot be reliably computed since the $Z(\boldsymbol{\phi})$ part involves integration over the entire input space. Alternatively, we show that the energy $E_{\boldsymbol{\phi}}(\boldsymbol{x})$ can also indicate the risk of input solution $\boldsymbol{x}$ effectively, and it is easy to be utilized.

Algorithm 1 Accumulative Risk Controlled Offline Optimization (ARCOO)

Input: Offline dataset $\mathcal{D}$, learning rate η, maximum Langevin dynamics step K, Langevin dynamics stepsize λ, and initial momentum m.

1: Initialize dual-head model that consists of surrogate head $\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and energy head $E_{\boldsymbol{\phi}}(\boldsymbol{x})$.
2: **for** each training epoch **do**
3: Update $\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ using MSE loss: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta})$.
4: Sample high-risk distribution $q(\boldsymbol{x})$ by Langevin dynamics: $q(\boldsymbol{x}) = LD_{\boldsymbol{\theta}}(p(\boldsymbol{x}); K)$, i.e., $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} + \lambda\nabla_{\boldsymbol{x}}\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{k-1}) + \boldsymbol{\omega}_k$, $k = 1, \ldots, K$, where $\omega_k^i \sim \mathcal{N}(0, \lambda)$, and $\boldsymbol{x}_0 \sim p(\boldsymbol{x})$. Sampling starts from the low-risk empirical distribution $p(\boldsymbol{x})$ over the offline dataset.
5: Update $E_{\boldsymbol{\phi}}(\boldsymbol{x})$ using contrastive divergence loss: $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \eta\nabla_{\boldsymbol{\phi}}[\mathrm{KL}(p(\boldsymbol{x})\|h_{\boldsymbol{\phi}}(\boldsymbol{x})) - \mathrm{KL}(q(\boldsymbol{x})\|h_{\boldsymbol{\phi}}(\boldsymbol{x}))]$, where $h_{\boldsymbol{\phi}}$ is derived from $E_{\boldsymbol{\phi}}(\boldsymbol{x})$.
6: **end for**
7: Let $\widetilde{\mathcal{P}}$ be an empirical distribution over a batch of the high-quality solutions in $\mathcal{D}$, and $\widetilde{\mathcal{Q}} = LD_{\boldsymbol{\theta}}(\widetilde{\mathcal{P}}; K)$.
8: Calculate the risk suppression factor: $R_{\boldsymbol{\phi}}(\boldsymbol{x}) = m(E_{\widetilde{\mathcal{Q}}} - E_{\boldsymbol{\phi}}(\boldsymbol{x}))(E_{\widetilde{\mathcal{Q}}} - E_{\widetilde{\mathcal{P}}})^{-1}$.
9: **for** $t = 1$ **to** T **do**
10: $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} + R_{\boldsymbol{\phi}}(\boldsymbol{x}_{t-1})\nabla_{\boldsymbol{x}}\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1})$.
11: **end for**
12: **Return** Final solution $\boldsymbol{x}_{\text{app}} = \boldsymbol{x}_T$ for online application.

Theorem 1. *(Equivalent form of $\mathcal{L}_{\mathrm{CD}}(\boldsymbol{\phi})$) Given a low-risk distribution $\mathcal{P}$ and a high-risk distribution $\mathcal{Q}$, $\mathcal{L}_{\mathrm{CD}}(\boldsymbol{\phi})$ in Equation (3) has an equivalent form of $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[E_{\boldsymbol{\phi}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}}[E_{\boldsymbol{\phi}}(\boldsymbol{x})]$.*

The proof of Theorem 1 is in Appendix A.1. The equivalent form of $\mathcal{L}_{\mathrm{CD}}(\boldsymbol{\phi})$ implies that training the modeled distribution $h_{\boldsymbol{\phi}}(\boldsymbol{x})$ is corresponding to directly training on the output of $E_{\boldsymbol{\phi}}(\boldsymbol{x})$. Guided by the alternative training strategy, the training process lowers the energy of solutions from the low-risk distribution $\mathcal{P}$ and raises the energy of solutions from the high-risk distribution $\mathcal{Q}$. In this way, the energy model can extract information from both $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ to estimate the risk of any given solution. Therefore, a low-risk solution is expected to have low energy and vice versa. Namely, the output energy could be seen as a representation of the risk for input solutions.

By now we have shown how the EBM is trained to model the risk of solutions. However, the training process of EBM involves a high-risk distribution $\mathcal{Q}$ that remains unfulfilled. Considering $\mathcal{Q}$ is identified as a distribution over OOD solutions that are easily overestimated, Markov Chain Monte Carlo (MCMC) [10, 32] methods can be applied to sample such OOD solutions and construct $\mathcal{Q}$. To achieve this, we utilize Langevin dynamics $LD_{\boldsymbol{\theta}}$ [22, 6] as the kernel of MCMC sampling. Let $\mathcal{Q} = LD_{\boldsymbol{\theta}}(\mathcal{P}; K)$, $\boldsymbol{x}_0 \sim \mathcal{P}$, $\boldsymbol{x}_k \sim \mathcal{Q}^k$, and $\mathcal{Q}^k$ is the distribution sampled in the k-th step of $LD_{\boldsymbol{\theta}}$, then

$$\boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} + \lambda \nabla_{\boldsymbol{x}} \widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{k-1}) + \boldsymbol{\omega}_k, k = 1, \ldots, K\,, \tag{4}$$

where ω_k^i denotes the i-th element of $\boldsymbol{\omega}_k$ and $\omega_k^i \sim \mathcal{N}(0, \lambda)$, and K denotes the maximum Langevin dynamics time step. $LD_{\boldsymbol{\theta}}(\mathcal{P}; K)$ starting from the low-risk distribution $\mathcal{P}$ and applying K steps of noisy gradient ascent on it to pursue a distribution over overestimated OOD solutions and return as high-risk distribution $\mathcal{Q}$. Intuitively, distribution with high risk is expected to be sampled with a large K. We now show that the risk of $\mathcal{Q}^k$ sampled in each step of Langevin dynamics is ascending, and ultimately, the return $\mathcal{Q}^K$ is high risk.

Let $\|g\|_\infty = \sup\{|g(\boldsymbol{x})| \colon \boldsymbol{x} \in \mathcal{X}\}$. Denote the total variation (which is a distance measure) of two probability distributions $\mathcal{U}$ and $\mathcal{V}$ as $TV(\mathcal{U}; \mathcal{V}) = \int_{\mathcal{X}} |u(\boldsymbol{x}) - v(\boldsymbol{x})| \mathrm{d}\boldsymbol{x}$, where $u(\boldsymbol{x})$ and $v(\boldsymbol{x})$ are the probability density functions of distributions $\mathcal{U}$ and $\mathcal{V}$.

Theorem 2. *(Upper Bound of the Prediction Error in Langevin Sampling) For any distribution $\mathcal{Q}^k$ sampled in the Langevin dynamics procedure, if $\|f\|_\infty$ and $\|\widehat{f}_{\boldsymbol{\theta}}\|_\infty$ exist, the prediction error of $\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ on $\mathcal{Q}^k$ is upper bounded*

$$\widehat{S}_{\boldsymbol{\theta}}(\mathcal{Q}^k) - S(\mathcal{Q}^k) \leq \widehat{S}_{\boldsymbol{\theta}}(\mathcal{P}) - S(\mathcal{P}) + C_{f;f_{\boldsymbol{\theta}}} \cdot TV(\mathcal{Q}^k; \mathcal{P}), \tag{5}$$

where $C_{f;f_{\boldsymbol{\theta}}} = 2(\|\widehat{f}_{\boldsymbol{\theta}}\|_\infty + \|f\|_\infty)$.

The proof of Theorem 2 is in Appendix A.2. Theorem 2 confirms the effectiveness of the self-supervised learning in ARCOO by showing the obtained $\mathcal{P}$ and $\mathcal{Q}^k$ are highly contrastive in terms of the risk level. It indicates that, as Langevin dynamics proceeds, the distance between the sampled OOD solutions distribution $\mathcal{Q}^k$ and low-risk distribution $\mathcal{P}$ is getting longer, so the total variance $TV(\mathcal{Q}^k; \mathcal{P})$ is getting larger. Therefore, the large predictive error is more likely to emerge on $\mathcal{Q}^k$. The final sampled distribution $\mathcal{Q}^K$ has the largest upper bound for $\widehat{S}_{\boldsymbol{\theta}}(\mathcal{Q}^k) - S(\mathcal{Q}^k)$ and is highly possible to make large prediction error, and thus it is a high-risk distribution over OOD solutions according to the corresponding definition in Section 2.

3.2 Risk-Control Optimization

ARCOO aims to explicitly apply risk in the optimizer and perform degradation-resistance in optimization. Although the energy produced by $E_{\boldsymbol{\phi}}(\boldsymbol{x})$ could be a representation of risk, independently using it is infeasible due to the contrastive energy learning making the output energy an absolute quantity. To this end, we introduce a normalization method to map the energy into a specific range and propose a *risk suppression factor* $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ as Equation (6). $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ suppresses the risk to a corresponding level in each step of the optimization procedure.

$$R_{\boldsymbol{\phi}}(\boldsymbol{x}) = \frac{m(E_{\widetilde{\mathcal{Q}}} - E_{\boldsymbol{\phi}}(\boldsymbol{x}))}{E_{\widetilde{\mathcal{Q}}} - E_{\widetilde{\mathcal{P}}}}\,, \tag{6}$$

where $E_{\widetilde{\mathcal{Q}}} = \mathbb{E}_{\boldsymbol{x}' \sim \widetilde{\mathcal{Q}}}[E_{\boldsymbol{\phi}}(\boldsymbol{x}')]$, $E_{\widetilde{\mathcal{P}}} = \mathbb{E}_{\boldsymbol{x}' \sim \widetilde{\mathcal{P}}}[E_{\boldsymbol{\phi}}(\boldsymbol{x}')]$, and m denotes the initial momentum. Similar to $\mathcal{P}$, $\widetilde{\mathcal{P}}$ is an empirical distribution over the high-quality batch of solutions in the offline dataset. Correspondingly, $\widetilde{\mathcal{Q}} = LD_{\boldsymbol{\theta}}(\widetilde{\mathcal{P}}; K)$ is a high-risk distribution sampled by K steps Langevin dynamics starting from $\widetilde{\mathcal{P}}$. The reason for using $\widetilde{\mathcal{P}}$ in $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ is that the optimization begins from the high-quality batch of solutions in offline data, and $\widetilde{\mathcal{P}}$ is set to be the low-risk distribution to keep align with this setting. Note that although we apply a step of K to keep consistent with the training process, one may set different K to alter the $\widetilde{\mathcal{Q}}$ in the risk suppression factor.

$R_{\boldsymbol{\phi}}(\boldsymbol{x})$ normalizes the energy of solutions found by the optimizer into the range of $m \to 0$ as $E_{\widetilde{\mathcal{P}}} \to E_{\widetilde{\mathcal{Q}}}$. And then $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ is applied along with surrogate prediction in gradient ascent optimizer to perform risk-control offline optimization,

$$\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} + R_{\boldsymbol{\phi}}(\boldsymbol{x}_{t-1}) \nabla_{\boldsymbol{x}} \widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1})\,. \tag{7}$$

Intuitively, $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ plays a similar role as the adaptive step size in online optimization methods (e.g., Adam [16]). Different from online methods that could tune the step size according to the gradient information of true evaluation feedback, offline optimizers are not allowed to query new evaluations. Instead, this paper proposes to attain risk information from the offline dataset and perform adaptive offline optimization using $R_{\boldsymbol{\phi}}(\boldsymbol{x})$. To be specific, the initial momentum m decides how fast the optimizer can behave at the beginning, and $E_{\widetilde{\mathcal{Q}}}$ decides how soon the $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ drops to zero. Hence, the risk-control optimizer can adapt the behavior according to the current risk situation. If the risk is large, $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ can suppress the risk of degradation to a low level. We further show that $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ has an upper bound correlate to the distance away from the high-risk distribution $\widetilde{Q}$.

Theorem 3. *(Upper Bound of the Risk Suppression Factor) Given a solution $\boldsymbol{x}$, consider a Gaussian distribution $\mathcal{N}(\boldsymbol{x}; \sigma)$ where σ is the standard deviation, if $\|E\|_\infty$ exists, then $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ in Equation* (6) *is upper bounded*

$$R_{\boldsymbol{\phi}}(\boldsymbol{x}) \leq m \frac{TV(\widetilde{\mathcal{Q}}; \mathcal{N}(\boldsymbol{x}; \sigma))}{TV(\widetilde{\mathcal{Q}}; \widetilde{\mathcal{P}})}\,. \tag{8}$$

The discussion of Theorem 3 is in Appendix A.3. Theorem 3 discloses that the risk suppression factor on each step of the risk-control optimization is constrained by the distance from the high-risk distribution $\widetilde{Q}$. Once the current found solution is close to high-risk distribution $\widetilde{Q}$, the bound will shrinkage and suppress the risk to induce performance degradation.

As discussed in Section 2, the risk control of the entire optimization procedure is essential for offline optimization to avoid performance degradation. One principal insight of ARCOO is to constrain the optimizer with respect to the accumulative risk. The behavior of ARCOO is flexibly and adaptively controlled under a certain level of risk. During the optimization, the risk suppression factor $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ converges to 0 as the energy $E_{\boldsymbol{\phi}}(\boldsymbol{x})$ approaches $E_{\widetilde{\mathcal{Q}}}$, which means for any time step T in Equation (1), the accumulation of $R_{\boldsymbol{\phi}}(\boldsymbol{x})$, called *accumulative risk*, can be upper bounded. That is, no matter how many time steps the optimizer goes, the overall risk is controlled under a certain level. However, since accumulating $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ in each step is mathematically intractable, we instead turn the accumulation into the integration of $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ as the $E_{\boldsymbol{\phi}}(\boldsymbol{x})$ increases from $E_{\widetilde{\mathcal{P}}}$ to $E_{\widetilde{\mathcal{Q}}}$, i.e.,

$$\int_{E_{\widetilde{\mathcal{P}}}}^{E_{\widetilde{\mathcal{Q}}}} R_{\boldsymbol{\phi}}(\boldsymbol{x}) \mathrm{d}E_{\boldsymbol{\phi}}(\boldsymbol{x}) \leq m \cdot \|E\|_\infty \cdot TV(\widetilde{\mathcal{Q}}; \widetilde{\mathcal{P}})\,. \tag{9}$$

The proof of Inequality (9) is in Appendix A.4. As shown above, in ARCOO, the accumulative risk is upper bounded jointly by initial momentum m and the distance between $\widetilde{\mathcal{P}}$ and $\widetilde{\mathcal{Q}}$. Conceptually, it

means a larger accumulative risk bound is related to larger initial momentum and a high-risk $\widetilde{\mathcal{Q}}$ further from $\widetilde{\mathcal{P}}$, and vice versa. Moreover, it is implied that as long as the accumulative risk of optimizers is upper bounded to a comparable extent, the behavior of each step in T can be trade-offed by m and $\widetilde{\mathcal{Q}}$, which could be seen as the maximum tolerance of risk. The empirical analysis in the following section supports the conclusions.

We would like to point out that, due to the absence of online feedback and information about optimal solution $\boldsymbol{x}^*$, in the offline optimization scenario, it is difficult to develop a bound of the distance between $\boldsymbol{x}_T$ and $\boldsymbol{x}^*$ or simple regret $f(\boldsymbol{x}^*) - f(\boldsymbol{x}_T)$ which is usually desired in online optimization. Instead, one of the main theoretical results of this paper is that ARCOO can control the accumulative risk, and thus it produces $\boldsymbol{x}_T$ with more reliability, which is also an important quality in offline optimization in addition to optimality.

4 Experiments

In this section, we first describe our experimental tasks, comparison methods, and implementation details. Then, elaborate comparative experiments are conducted to verify the effectiveness of ARCOO, and further comparison with conservative methods in terms of degradation resistance is performed. Next, an ablation study is presented to examine the capability of key components of ARCOO. Finally, we analyze the accumulative risk control via a pair of key hyperparameters in the risk suppression factor, and another crucial hyperparameter is studied. The implementation of ARCOO and the appendix of this paper can be found from https://github.com/luhuakang/ARCOO.

4.1 Experimental Setup

Offline Optimization Tasks. The experiments are conducted on three discrete offline optimization tasks and three continuous ones from Design-bench [27], which is an offline optimization benchmark that is widely used in recent work [8, 35, 3]. The employed tasks are based on the problems of DNA optimization, drug discovery, material invention, and robotic design. For some offline optimization tasks, the ground-truth function, i.e., black-box objective function, for new solution evaluation is intractable, and various oracles need to be designed according to the properties of specific tasks. We briefly depict the goal and dataset of each task as follows, and more details about the employed tasks and oracles are described in Appendix B.

(1) `TF Bind 8` task aims to design a length-8 DNA sequence for human transcription factor SIX6_REF_R1 with maximum binding activity. The dataset consists of 32898 data, and each datum is a length-8 sequence of categorical variables that take one of 4 nucleotides.

(2) `ChEMBL` task designs a molecule for assay CHEMBL3885882 with maximum MCHC value, which is derived from a large-scale drug property database [9]. The dataset consists of 1093 data, and each datum is a length-31 SMILES string of categorical variables that take one of 591 elements.

(3) `UTR` task aims to design a length-50 DNA sequence with maximum expression level, which is derived from work [24]. The dataset consists of 140000 data, and each datum is a length-50 sequence of categorical variables that take one 4 nucleotides.

(4) `Superconductor` task aims to optimize the chemical formula for a superconducting material with maximum critical temperature. The dataset consists of 21263 data, and each datum is an 86-dimensional vector of continuous variables that represents the number of atoms in the superconductor chemical formula.

(5) `Dkitty` task aims to optimize Dkitty robot morphologies for navigation with maximum efficiency. The dataset consists of 25009 data, and each datum is a 56-dimensional vector of robot parameters such as size and orientation.

(6) `Hopper` task aims to optimize the weights of a neural network policy with maximum Hopper-v2 Gym [1] return. The dataset consists of 3200 data, and each datum is a 5126-dimensional parameter of neural network weights.

Baselines and SOTA. We compare ARCOO with various offline optimization baseline methods. We consider Bayesian optimization equipped with quasi expected improvement acquisition function (BO-qEI) [34], CMA-ES [12], REINFORCE [33], and gradient ascent. Note that the above methods are unable to solve offline optimization problems directly since none of them is proposed for the offline setting. Therefore, an ensemble of 5 surrogate models trained with bootstrapping is applied to provide surrogate prediction and guide the optimization methods. The SOTA methods are also involved: DDEA-SE [30], CbAS [2], MIN [18], IOM [23], COMs [28] and NEMO [8].

Implementation Details and Hyperparameters. The dual-head surrogate and energy model in ARCOO has the same architecture across all the experiments, which consists of one shared hidden layer, one hidden layer surrogate head, and one hidden layer energy head. The size is set to 2048 and the activation function is ReLU for all hidden layers. Adam optimizers [16] with the same learning rate $\eta = 0.001$ (Line 3 and 5, Algorithm 1) are used to train each model.

For the hyperparameters, they are implemented universally for all tasks unless otherwise specified: The Langevin dynamics steps K is set to 64 according to the results in Section 4.4 showing it is enough to sample high-risk distribution for energy model training. We set initial momentum $m = 0.02$ for all continuous tasks and $m = 2$ for all discrete tasks with respect to the complexity of different kinds of tasks. The Langevin dynamics stepsize λ (Line 4, Algorithm 1) is set to be the same number of m as a reasonable value for MCMC sampling in the input space. As analyzed in Section 3.2, for any optimization steps $T > 0$ (Line 9, Algorithm 1), the accumulative risk of the whole optimization procedure is controlled and upper bounded. Thus, we set the optimization steps as a sufficiently large number, i.e., $T = 200$.

The evaluation setup follows recent studies [28, 8, 2, 18]. Instead of starting from only the best solution in the offline dataset, we maintain and optimize a batch of 128 candidate solutions (Line 7, Algorithm 1) and output the best among them as the final solution. The experimental results are averaged over 8 trials and the standard deviation is reported. All the experiments are processed with dual NVIDIA RTX 3090 GPUs and dual Intel Xeon Gold 6354 CPU @3.00GHz CPUs. It should be noted that any experimental trial exceeding a duration of 48 hours is deemed to be unsuccessful.

4.2 Comparative Experiment

Optimality of Output Solution for Online Application. The results of comparative experiments are provided in Table 1. In addition to the output solutions scores on each task, the average of improvements to the best solution in offline dataset $\boldsymbol{x}^\star_{\text{OFF}} = \arg\max_{\boldsymbol{x}_i} y_i$, where $(\boldsymbol{x}_i, y_i) \in \mathcal{D}$, on all 6 tasks are also reported, i.e., score/$\boldsymbol{x}^\star_{\text{OFF}}$.

In the Hopper task, which is relatively hard to optimize with a high searching dimension of 5126, all the conservative methods (i.e., NEMO, COMs, and IOM) and ARCOO show significantly greater improvements to offline dataset (1.84$\times$ in average) than other methods (1.08$\times$ at most). These suggest that restricting the exploration to OOD solutions is an effective way to deal with the performance degradation issue. Among these methods, although IOM outperforms

Table 1. Comparisons on the quality of output solutions for online application with the baseline and SOTA methods on each task. $\boldsymbol{x}^\star_{\text{OFF}}$ indicates the best solution in each offline dataset. The mean and standard deviation of performance are reported. The symbol "-" means that the algorithm cannot complete the corresponding task within 48 hours. The last column reports the average performance improvement to $\boldsymbol{x}^\star_{\text{OFF}}$. The best mean scores on each task are marked to be bold.

	Discrete Tasks			**Continuous Tasks**			
	TF Bind 8	**ChEMBL**	**UTR**	**Superconductor**	**Dkitty**	**Hoppper**	**Avg.**
$\boldsymbol{x}^\star_{\text{OFF}}$	0.44	383700.00	7.12	73.90	199.40	1361.60	1.00×
BO-qEI	0.86±0.06	401910.23±3719.47	8.04±0.13	94.89±1.90	213.80±0.00	763.77±136.89	1.18×
CMA-ES	0.93±0.04	386032.58±3448.00	9.04±0.20	89.16±4.22	5.11±1.88	938.68±240.50	1.05×
REINFORCE	0.93±0.04	**402077.50**±3132.94	8.17±0.07	89.34±3.65	-142.71±227.30	29.13±40.68	0.81×
Gradient Ascent	**0.97**±0.01	397463.75±3125.75	8.11±0.41	92.94±3.07	186.18±21.99	1465.71±930.18	1.28×
CbAS	0.90±0.05	397757.52±3984.75	8.25±0.10	83.57±6.77	214.33±9.13	262.78±13.38	1.11×
MINs	0.86±0.07	396902.52±2440.44	8.25±0.08	85.34±6.98	269.01±7.41	294.79±160.36	1.15×
DDEA-SE	0.91±0.03	383736.25±3197.65	9.01±0.18	87.99±3.82	5.34±1.77	686.33±265.55	1.01×
IOM	0.94±0.04	389105.00±1659.29	–	86.51±12.41	262.26±8.73	**2764.13**±516.22	1.44×
NEMO	0.90±0.06	393908.75±4644.34	8.25±0.13	92.55±3.66	267.90±8.90	2225.52±209.68	1.41×
COMs	0.93±0.04	397572.50±2657.84	8.29±0.25	81.36±6.95	268.05±5.96	2573.25±537.03	1.44×
ARCOO	**0.97**±0.01	399826.25±4256.49	**9.48**±0.19	**95.56**±3.62	**277.23**±8.38	2485.62±304.78	**1.52**×

in some tasks, it fails to complete the UTR task within the time limitation (i.e., 48 hours). Due to the relatively large-scale property of UTR, adding complicated conservative regularization to the surrogate model may dramatically increase the computational complexity. ARCOO is the only method that achieves the best score on four out of six tasks and within a standard deviation margin of the best score in the rest tasks. These results are promising in that the adaptation ability of ARCOO with the risk-control optimizer is substantially greater than conservative model based methods across a variety of tasks with respect to the optimality of output solutions for online applications.

Degradation-Resistance in Optimization Procedure. The results above indicate that NEMO, COMs, and ARCOO are promising methods to deal with offline optimization problems. While all of them benefit from constraining the optimizer from searching for OOD solutions, the main difference among them is the way to perform this constraint. As NEMO and COMs impose conservatism prior to the surrogate models, ARCOO directly suppresses the risk in the optimization procedure and performs accumulative risk control. To further illustrate this discrepancy, we conduct an experiment on the Hopper task with a large number of optimization steps (i.e., $T = 200$) to show the behavior of optimizers in different methods.

The visualization results are shown in Figure 2. In the early stage of optimization, all three methods search for better solutions stably under low-risk situations. As all the methods reach the highest online evaluation score of their own optimization procedure, it comes to a point where both the surrogate prediction and the risk are large, because it is already far from the offline dataset. For NEMO and COMs, since the goal of the optimizers is to pursue better solutions, it has to continue to perform gradient ascent with a constant learning rate even if the risk is relatively high at this point. Consequently, the optimizers break the implicit conservative constraint in the surrogate model and lead to degradation of performance at the later stage of optimization. Conversely, ARCOO acts as fast as COMs at the beginning of the optimization procedure but searches slower as the procedure goes on. That is because when the optimizer gets further outside the observed area, the energy rises up and $R_\phi(\boldsymbol{x})$ tends to suppress the risk of degradation to a lower level since the surrogate model would probably produce overestimated results. Since the accumulative risk is controlled to a certain extent. the $R_\phi(\boldsymbol{x})$ would eventually approach 0, and the optimizer would not go further to avoid performance degradation. With the ability of degradation resistance, setting the termination condition T as a large number becomes innocuous.

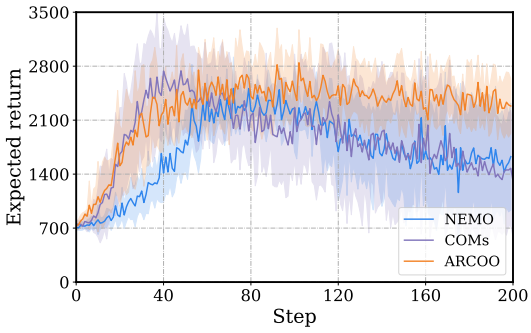


Figure 2. Comparison among conservative methods and ARCOO in the ability of performance degradation resistance on the Hopper task. The vertical axis signifies the true evaluation score of solutions, which are only shown for visualization purposes and are inaccessible for offline optimization.

4.3 Ablation Study

The results of previous experiments suggest that ARCOO can outperform the SOTA offline optimization methods and carry out risk control in the offline optimizer effectively. We further examine how the key components of ARCOO work by conducting an ablation study on the Hopper task. To be specific, we compare our original ARCOO with (1) SE-GA that uses gradient ascent (GA) as optimizer but still involves the whole training process, and (2) S-GA that only applies training on $\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and is equipped with GA optimizer.

The results are shown in Figure 3. Compared with ARCOO, SE-GA fails on performance degradation which is caused by high-risk overestimated OOD solutions at the later stage of optimization. It should be noted that GA can be regarded as a risk-ignore version of ARCOO, and this confirms the significance of the risk-control optimizer in offline optimization. Intriguingly, it is also observed that SE-GA is more stable than S-GA in terms of mitigating performance degradation, despite both of them employing GA as optimizers. Since SE-GA additionally trains $E_\phi(\boldsymbol{x})$, the predictions made by $\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$

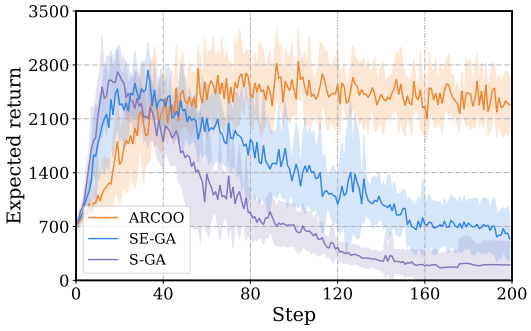


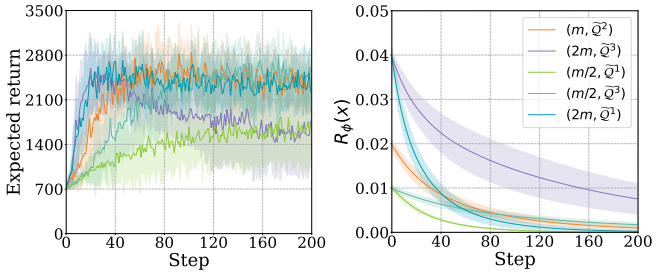
Figure 3. Ablation of key components in ARCOO on the Hopper task. The vertical axis signifies the true evaluation score of solutions.

could benefit from the shared layer, to which $E_{\boldsymbol{\phi}}(\boldsymbol{x})$ embeds risk information in the training time. This reveals that ARCOO not only utilizes risk in the optimizer explicitly but also in $\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ implicitly.

4.4 *Hyperparameter Analysis*

Analysis of the Accumulative Risk Control. As in Section 3.2, the accumulative risk is controlled under a certain level by the upper bound, which can be seen as the level of total risk tolerance. To verify it, we empirically study the accumulative risk control mechanism through the hyperparameters that contribute to this upper bound (i.e., m and $\widetilde{\mathcal{Q}}$). Since $\widetilde{\mathcal{Q}}$ is sampled by $LD_{\boldsymbol{\theta}}(\widetilde{\mathcal{P}}; K)$, different K are used to derive different high-risk distributions. Specifically, five (m, $\widetilde{\mathcal{Q}}$) settings for ARCOO are investigated on the Hopper task. We set $m = 0.02$ and $\widetilde{\mathcal{Q}}^2 = LD_{\boldsymbol{\theta}}(\widetilde{\mathcal{P}}; K)$ with $K = 64$. Then, we have $\widetilde{\mathcal{Q}}^1 = LD_{\boldsymbol{\theta}}(\widetilde{\mathcal{P}}; K/2)$ and $\widetilde{\mathcal{Q}}^3 = LD_{\boldsymbol{\theta}}(\widetilde{\mathcal{P}}; 2K)$, and notice that $TV(\widetilde{\mathcal{P}}; \widetilde{\mathcal{Q}}^1) < TV(\widetilde{\mathcal{P}}; \widetilde{\mathcal{Q}}^2) < TV(\widetilde{\mathcal{P}}; \widetilde{\mathcal{Q}}^3)$. Finally, five hyperparameter settings are chosen as $s1\colon (m/2, \widetilde{\mathcal{Q}}^1)$, $s2\colon (m/2, \widetilde{\mathcal{Q}}^3)$, $s3\colon (m, \widetilde{\mathcal{Q}}^2)$, $s4\colon (2m, \widetilde{\mathcal{Q}}^1)$, and $s5\colon (2m, \widetilde{\mathcal{Q}}^3)$.

From Figure 4 (a), we observe that ARCOO$_{s1}$ shows the strongest risk control as it searches for a sub-optima solution with too low risk in the final output. On the contrary, ARCOO$_{s5}$ shows the weakest risk control that it is misled by the overestimated surrogate prediction and falls into performance degradation. The other ARCOO settings show comparative performance as they are constrained with a proper accumulative risk control upper bound. These results fit the analyses in Section 3.2 that m and $\widetilde{\mathcal{Q}}$ can flexibly control the accumulative risk



(a) Expected return in the optimization procedure

(b) $R_\phi(\boldsymbol{x})$ in the optimization procedure

Figure 4. Accumulative risk control hyperparameters analysis on different combinations of m and $\widetilde{\mathcal{Q}}$. Figure (a) shares the same legend with (b). We draw the mean and standard deviation of true evaluation score and $R_\phi(\boldsymbol{x})$.

to desired level. We further consider the tradeoff between m and $\widetilde{\mathcal{Q}}$ by comparing ARCOO$_{s2}$, ARCOO$_{s3}$, and ARCOO$_{s4}$. Although all three settings lead to similar results, the optimizers behave differently during the procedure.

As shown in Figure 4 (b), ARCOO$_{s4}$ has the largest $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ at the very beginning and behaves radically to reach the best solution firstly, while ARCOO$_{s4}$ starts with a small $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ then slowly search for the optimal solution and ARCOO$_{s3}$ is in the middle of them. Accordingly, the tradeoff between m and $\widetilde{\mathcal{Q}}$ is a way to control the behavior of every step in the optimization procedure under a certain level of accumulative risk control across the whole optimization procedure.

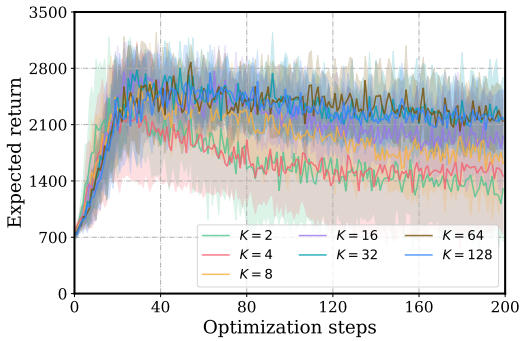


Figure 5. Comparison of ARCOO optimization performance under different Langevin dynamics steps K values on the Hopper task. The vertical axis signifies the true evaluation score of solutions.

Choice of Langevin Dynamics Steps K. Intuitively, the Langevin dynamics steps K (Line 4, Algorithm 1) determine how far the high-risk solutions are from the observed solutions. A proper number of K allows the dual-head surrogate model to assign high energy to solutions with high risk in the energy training phase. In order to study the sensitivity of ARCOO to K, we compare the optimization performance (the true evaluation score) of ARCOO under five K values, i.e., $K \in \{2, 4, 8, 16, 32, 64, 128\}$ on the Hopper task. As can be observed from Figure 5, the robustness of ARCOO to overestimated surrogate prediction improves as K increases from 2 to 32. However, when K continues to increase from 32 to 128, all three curves turn out to be substantially similar, in terms of avoiding performance degradation at the later stage of optimization. The results demonstrate that Langevin dynamics is capable of finding the high-risk solutions for energy training with any sufficiently large K. Thus, we empirically set $K = 64$ for ARCOO in all experiments.

5 Conclusion

This paper addresses the performance degradation in offline optimization. We formalize the risk of degradation for OOD solutions and propose an accumulative risk controlled offline optimization (ARCOO) method based on the energy model, in which the accumulative risk in the optimization procedure is controlled by a risk suppression factor. Theoretical and empirical results show that ARCOO possesses the merit of risk tolerance, and the behavior of ARCOO can be flexibly and adaptively controlled. The future work includes extending ARCOO to the online Bayesian-like optimization scenario, developing more powerful theoretical analysis tools to reveal the simple regret bound, and exploring more real-world applications.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive comments. The authors also would like to thank Prof. Ke Tang for the helpful discussion. This work is supported by the National Natural Science Foundation of China (No. 62106076), the CCF-AFSG Research Fund (No. CCF-AFSG RF20220205), and the Natural Science Foundation of Shanghai (No. 21ZR1420300).

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, 'Openai gym', *CoRR*, **abs/1606.01540**, (2016).

[2] David H. Brookes, Hahnbeom Park, and Jennifer Listgarten, 'Conditioning by adaptive sampling for robust design', in *Proceedings of the 36th International Conference on Machine Learning*, pp. 773–782, Long Beach, California, (2019).

[3] Can Chen, Yingxue Zhang, Jie Fu, Xue Liu, and Mark Coates, 'Bidirectional learning for offline infinite-width model-based optimization', *CoRR*, **abs/2209.07507**, (2022).

[4] Ian A. Delbridge, David Bindel, and Andrew Gordon Wilson, 'Randomly projected additive gaussian processes for regression', in *Proceedings of the 37th International Conference on Machine Learning*, pp. 2453–2463, Vienna, Austria, (2020).

[5] Yilun Du, Shuang Li, Joshua B. Tenenbaum, and Igor Mordatch, 'Improved contrastive divergence training of energy-based models', in *Proceedings of the 38th International Conference on Machine Learning*, pp. 2837–2848, Virtual, (2021).

[6] Yilun Du and Igor Mordatch, 'Implicit generation and modeling with energy based models', in *Advances in Neural Information Processing Systems 32*, pp. 3603–3613, Vancouver, Canada, (2019).

[7] Clara Fannjiang and Jennifer Listgarten, 'Autofocused oracles for model-based design', in *Advances in Neural Information Processing Systems 33*, pp. 12945–12956, Virtual, (2020).

[8] Justin Fu and Sergey Levine, 'Offline model-based optimization via normalized maximum likelihood estimation', in *Proceedings of the 9th International Conference on Learning Representations*, Virtual, (2021).

[9] Anna Gaulton, Louisa J. Bellis, A. Patrícia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, and John P. Overington, 'ChEMBL: A large-scale bioactivity database for drug discovery', *Nucleic Acids Research*, **40**, 1100–1107, (2012).

[10] Charles J Geyer, 'Practical Markov chain Monte Carlo', *Statistical Science*, **7**(4), 473–483, (1992).

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio, 'Generative adversarial nets', in *Advances in Neural Information Processing Systems 27*, pp. 2672–2680, Montreal, Canada, (2014).

[12] Nikolaus Hansen, 'The CMA evolution strategy: A comparing review', in *Towards a New Evolutionary Computation - Advances in the Estimation of Distribution Algorithms*, volume 192, 75–102, Springer, (2006).

[13] Geoffrey E. Hinton, 'Training products of experts by minimizing contrastive divergence', *Neural Computation*, **14**(8), 1771–1800, (2002).

[14] Yaochu Jin, 'Surrogate-assisted evolutionary computation: Recent advances and future challenges', *Swarm and Evolutionary Computation*, **1**(2), 61–70, (2011).

[15] Yaochu Jin, Handing Wang, Tinkle Chugh, Dan Guo, and Kaisa Miettinen, 'Data-driven evolutionary optimization: An overview and case studies', *IEEE Transactions on Evolutionary Computation*, **23**(3), 442–458, (2019).

[16] Diederik P. Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', in *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, (2015).

[17] Diederik P Kingma and Max Welling, 'Auto-encoding variational bayes', *arXiv preprint arXiv:1312.6114*, (2013).

[18] Aviral Kumar and Sergey Levine, 'Model inversion networks for model-based optimization', in *Advances in Neural Information Processing Systems 33*, pp. 5126–5137, Virtual, (2020).

[19] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang, 'A tutorial on energy-based learning', *Predicting Structured Data*, **1**(0), (2006).

[20] Aria Mansouri Tehrani, Anton O Oliynyk, Marcus Parry, Zeshan Rizvi, Samantha Couper, Feng Lin, Lowell Miyagi, Taylor D Sparks, and Jakoah Brgoch, 'Machine learning directed search for ultraincompressible, superhard materials', *Journal of the American Chemical Society*, **140**(31), 9844–9853, (2018).

[21] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee, 'Dataset meta-learning from kernel ridge-regression', in *Proceedings of the 9th International Conference on Learning Representations*, Virtual, (2021).

[22] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu, 'Learning non-convergent non-persistent short-run MCMC toward energy-based model', in *Advances in Neural Information Processing Systems 32*, pp. 5233–5243, Vancouver, Canada, (2019).

[23] Han Qi, Yi Su, Aviral Kumar, and Sergey Levine, 'Data-driven model-based optimization via invariant representation learning', in *Advances in Neural Information Processing Systems 35*, New Orleans, Louisiana, (2022).

[24] Paul J Sample, Ban Wang, David W Reid, Vlad Presnyak, Iain J McFadyen, David R Morris, and Georg Seelig, 'Human 5 utr design and variant effect prediction from a massively parallel translation assay', *Nature Biotechnology*, **37**(7), 803–809, (2019).

[25] 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*, **104**(1), 148–175, (2016).

[26] Lei Song, Ke Xue, Xiaobin Huang, and Chao Qian, 'Monte Carlo tree search based variable selection for high-dimensional Bayesian optimization', in *Advances in Neural Information Processing Systems 35*, New Orleans, LA, (2022).

[27] Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine, 'Design-bench: Benchmarks for data-driven offline model-based optimization', in *Proceedings of the 39th International Conference on Machine Learning*, pp. 17–23, Baltimore, Maryland, (2022).

[28] Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine, 'Conservative objective models for effective offline model-based optimization', in *Proceedings of the 38th International Conference on Machine Learning*, pp. 10358–10368, Virtual, (2021).

[29] Handing Wang, Yaochu Jin, and Jan O. Jansen, 'Data-driven surrogate-assisted multiobjective evolutionary optimization of a trauma system', *IEEE Transactions on Evolutionary Computation*, **20**(6), 939–952, (2016).

[30] Handing Wang, Yaochu Jin, Chaoli Sun, and John Doherty, 'Offline data-driven evolutionary optimization using selective surrogate ensembles', *IEEE Transactions on Evolutionary Computation*, **23**(2), 203–216, (2019).

[31] Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer, 'Recent advances in bayesian optimization', *CoRR*, **abs/2206.03301**, (2022).

[32] Max Welling and Yee Whye Teh, 'Bayesian learning via stochastic gradient langevin dynamics', in *Proceedings of the 28th International Conference on Machine Learning*, pp. 681–688, Bellevue, Washington, (2011).

[33] Ronald J. Williams, 'Simple statistical gradient-following algorithms for connectionist reinforcement learning', *Machine Learning*, **8**, 229–256, (1992).

[34] James T. Wilson, Riccardo Moriconi, Frank Hutter, and Marc Peter Deisenroth, 'The reparameterization trick for acquisition functions', *CoRR*, **abs/1712.00424**, 75–102, (2017).

[35] Sihyun Yu, Sungsoo Ahn, Le Song, and Jinwoo Shin, 'RoMA: Robust model adaptation for offline model-based optimization', in *Advances in Neural Information Processing Systems 34*, pp. 4619–4631, Virtual, (2021).

[36] Yangwenhui Zhang, Hong Qian, Xiang Shu, and Aimin Zhou, 'High-dimensional dueling optimization with preference embedding', in *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, pp. 11280–11288, Washington, DC, (2023).

[37] Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J. Gordon, 'On learning invariant representations for domain adaptation', in *Proceedings of the 36th International Conference on Machine Learning*, pp. 7523–7532, Long Beach, California, (2019).

[38] Ping Zhou, Youbin Lv, Hong Wang, and Tianyou Chai, 'Data-driven robust RVFLNs modeling of a blast furnace iron-making process using cauchy distribution weighted m-estimation', *IEEE Transactions on Industrial Electronics*, **64**(9), 7141–7151, (2017).