Offline Model-Based Optimization: Comprehensive Review

Anonymous authors
Paper under double-blind review

Abstract

Offline optimization is a fundamental challenge in science and engineering, where the goal is to optimize black-box functions using only offline datasets. This setting is particularly relevant when querying the objective function is prohibitively expensive or infeasible, with applications spanning protein engineering, material discovery, neural architecture search, and beyond. The main difficulty lies in accurately estimating the objective landscape beyond the available data, where extrapolations are fraught with significant epistemic uncertainty. This uncertainty can lead to *objective hacking* (*reward hacking*)—exploiting model inaccuracies in unseen regions—or other spurious optimizations that yield misleadingly high performance estimates outside the training distribution. Recent advances in *model-based optimization* (MBO) have harnessed the generalization capabilities of deep neural networks to develop offline-specific surrogate and generative models. Trained with carefully designed strategies, these models are more robust against out-of-distribution issues, facilitating the discovery of improved designs. Despite its growing impact in accelerating scientific discovery, the field lacks a comprehensive review. To bridge this gap, we present the first thorough review of offline MBO. We begin by formalizing the problem for both *single-objective* and *multi-objective* settings and by reviewing recent benchmarks and evaluation metrics. We then categorize existing approaches into two key areas: *surrogate modeling*, which emphasizes accurate function approximation in out-of-distribution regions, and *generative modeling*, which explores high-dimensional design spaces to identify high-performing designs. Finally, we examine the key challenges and propose promising directions for advancement in this rapidly evolving field including safe control of superintelligent systems.

1 Introduction

Offline optimization is a fundamental challenge in science and engineering, where the objective is to optimize a black-box function using only a fixed dataset (Trabucco et al., 2022). This setting has broad applications, including protein engineering (Sarkisyan et al., 2016b), material discovery (Hamidieh, 2018), and neural architecture search (Lu et al., 2023). For instance, in neural architecture search, the goal is to identify high-performing architectures solely from existing architecture-performance pairs, without training any new models which can be expensive. Unlike *online optimization*, which allows direct interaction with the objective function, offline optimization is particularly relevant when querying the function is costly, time-consuming, or infeasible (Angermüller et al., 2020; Barrera et al., 2016b; Sample et al., 2019b).

Offline optimization is challenging because it requires accurately estimating the landscape of the black-box function beyond the available offline data (Trabucco et al., 2021). Extrapolating into these unseen regions suffers from significant epistemic uncertainty in two problematic scenarios. On the one hand, it may trigger *reward hacking* (Skalse et al., 2022) (which we refer to as *objective hacking* in our text), wherein the model exploits inaccuracies within the objective estimation in regions beyond training data. On the other hand, it can give rise to other forms of spurious optimization – especially in guided generative modeling – that yield misleadingly high performance estimates outside the training distribution (Brookes et al., 2019).

Recent advances in offline *model-based optimization* (MBO) have harnessed the generalization power of deep neural networks to develop offline-specific surrogate and generative models. This progress has spurred two complementary lines of research. One line focuses on building *surrogate models* that extrapolate beyond

the offline dataset, enabling robust function approximation and reliable gradient-based optimization to improve existing designs (Trabucco et al., 2021; Fu & Levine, 2021). The other line explores the use of *generative models* to navigate high-dimensional design spaces more effectively, facilitating the discovery of high-performing designs underrepresented in the offline data (Kumar & Levine, 2020; Kim et al., 2024b). Importantly, these two lines are not mutually exclusive – surrogate and generative models often complement each other to enhance overall performance (Fannjiang & Listgarten, 2020; Chen et al., 2024).

Despite the rapid progress in offline MBO, both newcomers and seasoned researchers find it challenging to stay abreast of its evolving methodologies. Furthermore, the diversity of approaches and objectives has led to a fragmented landscape, making it difficult to discern overarching trends. To address these challenges, we present the first comprehensive review on offline MBO, synthesizing recent advances, categorizing key areas, and highlighting emerging directions. This review serves as both an accessible introduction for newcomers and a structured synthesis for experts looking to navigate the evolving frontiers of offline optimization.

In this work, we formalize the problem settings for both offline *single-* and *multi-objective optimization* (Section 2). We also introduce a generative modeling perspective that frames offline optimization as *conditional generation*, compare single- and multi-objective settings, and discuss the connections between online and offline optimization. Additionally, we review recent benchmarks and propose a taxonomy that categorizes them into four application areas (Section 3): (1) *synthetic function*, (2) *real-world system*, (3) *scientific design*, and (4) *machine learning (ML) model*. Evaluation costs tend to increase – and our understanding of the underlying mechanisms tends to decrease – from categories (1) through (3). We address category (4) separately, given its growing prominence in the ML community. For each category, we detail the associated tasks, including the number of objectives and the oracle evaluators. We also provide an overview of the commonly used evaluation metrics, including *usefulness*, *novelty*, and *diversity*.

Next, we categorize existing approaches into two key research lines—as we have discussed above: *surrogate modeling* (Section 4) and *generative modeling* (Section 5). Importantly, these approaches are not mutually exclusive; we explore in detail how they interact and complement one another. Finally, we conclude our paper (Section 6) by outlining promising future directions in this rapidly evolving field. In particular, we highlight several key areas for further exploration: (1) *robust and realistic benchmarking*, (2) *uncertainty estimation of surrogate models*, (3) *causal graphical surrogate models*, (4) *advanced generative modeling*, (5) and *application to LLM alignment and AI Safety*. An outline of our paper organization is depicted in Figure 1.

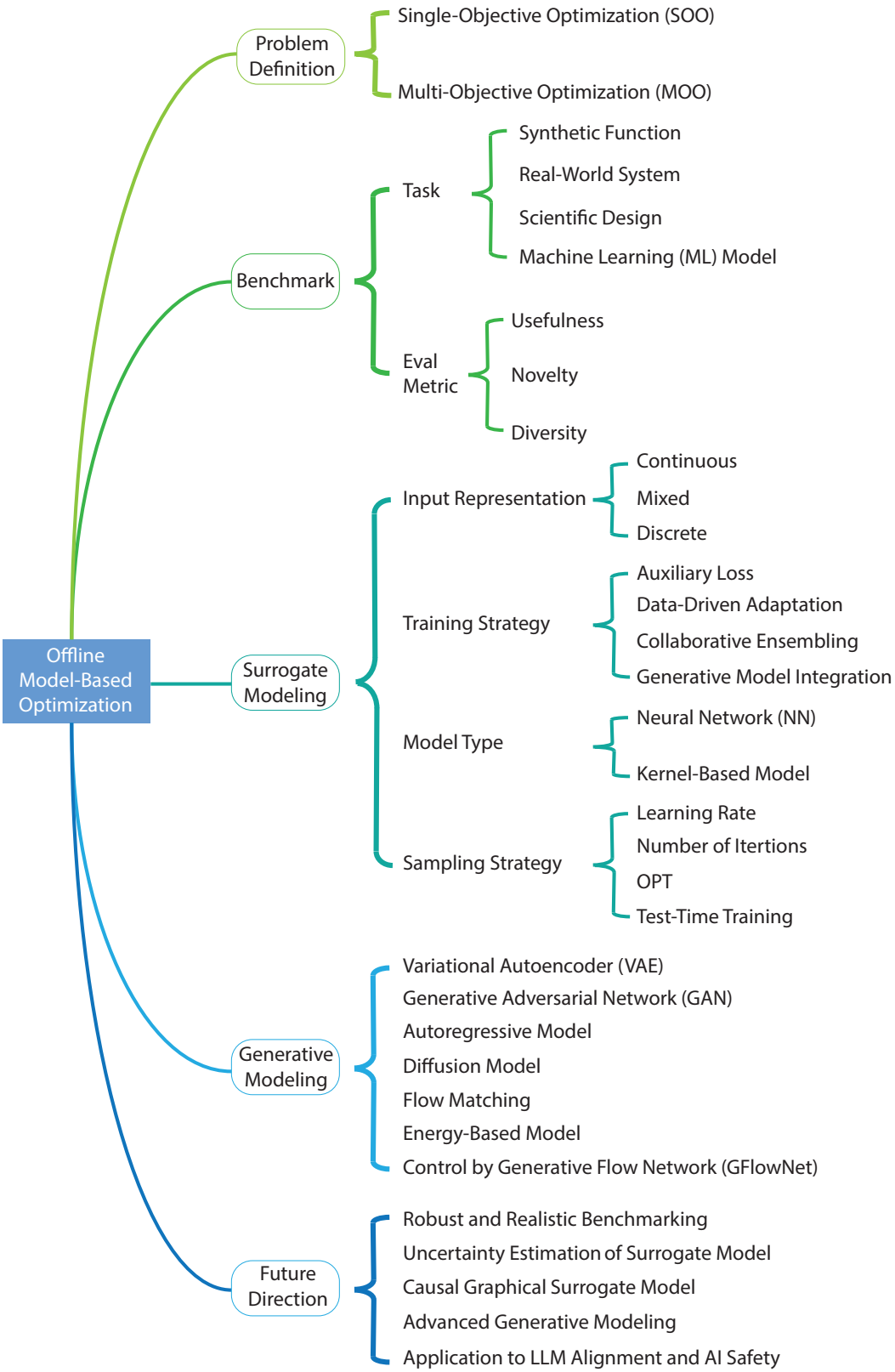

Figure 1: Paper overview.

2 Problem Definition

In offline optimization, the goal is to discover a new design, denoted by $\boldsymbol{x}^*$, that maximizes the objective(s) $\boldsymbol{f}(\boldsymbol{x})$. This is achieved using an offline dataset $\mathcal{D}$, which consists of N designs paired with their property labels. In particular, the dataset is given by

$$\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N} \,. \tag{1}$$

where each design vector $\boldsymbol{x}_i$ belongs to a design space $\mathcal{X} \subseteq \mathbb{R}^d$, and each property label $\boldsymbol{y}_i \in \mathbb{R}^m$ contains the corresponding m objective values for that design. The function $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^m$ maps a design to its m-dimensional objective value vector.

Single-Objective Optimization In offline *single-objective optimization* (SOO), only one objective is considered (i.e., $m = 1$), leading to the formulation:

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} f(\boldsymbol{x}) \,.$$

For instance, the design $\boldsymbol{x}$ might represent a neural network architecture, with $f(\boldsymbol{x})$ denoting the network's accuracy on a given dataset (Zoph & Le, 2017).

A prevalent method for addressing this problem involves training a deep neural network (DNN) surrogate model, $f_{\boldsymbol{\phi}}(\cdot)$, with parameters $\boldsymbol{\phi}$ on an offline dataset using supervised learning. The model parameters are optimized by minimizing the mean squared error between the model's predictions and the true labels:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \frac{1}{N} \sum_{i=1}^{N} \Big(f_{\boldsymbol{\phi}}(\boldsymbol{x}_i) - y_i \Big)^2 \,. \tag{2}$$

After training, the surrogate $f_{\boldsymbol{\phi}^*}(\cdot)$ is employed as a stand-in for the true objective function, and design optimization proceeds via gradient ascent updates:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \eta \nabla_{\boldsymbol{x}} f_{\boldsymbol{\phi}^*}(\boldsymbol{x}) \Big|_{\boldsymbol{x}=\boldsymbol{x}_t} \,, \quad \text{for } t \in [0, T-1] \,, \tag{3}$$

where η is the learning rate and T is the total number of iterations. The final design, $\boldsymbol{x}_T$, is then taken as the candidate solution.

A critical challenge of this method is the accurate estimation of the objective landscape beyond the region covered by the training data. In these extrapolated regions, the surrogate model's predictions are subject to significant epistemic uncertainty, potentially leading to *objective (or reward) hacking*. In other words, the model might exploit inaccuracies in regions lacking training data. We discuss strategies for building robust surrogates in Section 4.

Multi-Objective Optimization Offline *multi-objective optimization* (MOO) extends the framework to simultaneously address multiple objectives using the dataset $\mathcal{D}$. In this setting, the goal is to find solutions that balance competing objectives effectively. For instance, when designing a neural architecture, one might seek to achieve both high accuracy and high efficiency (Lu et al., 2023). Formally, the multi-objective optimization problem is defined as:

$$\text{Find } \boldsymbol{x}^* \in \mathcal{X} \text{ such that there is no } \boldsymbol{x} \in \mathcal{X} \text{ with } \boldsymbol{f}(\boldsymbol{x}) \succ \boldsymbol{f}(\boldsymbol{x}^*) \,, \tag{4}$$

where $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^m$ is the vector of m objective functions and the symbol $\succ$ indicates *Pareto dominance*. Specifically, a solution $\boldsymbol{x}$ is said to *Pareto dominate* another solution $\boldsymbol{x}^*$ (denoted $\boldsymbol{f}(\boldsymbol{x}) \succ \boldsymbol{f}(\boldsymbol{x}^*)$) if

$$\forall i \in \{1, \ldots, m\}, \quad f_i(\boldsymbol{x}) \geq f_i(\boldsymbol{x}^*) \quad \text{and} \quad \exists j \in \{1, \ldots, m\} \text{ such that } f_j(\boldsymbol{x}) > f_j(\boldsymbol{x}^*) \,. \tag{5}$$

In simpler terms, $\boldsymbol{x}$ is no worse than $\boldsymbol{x}^*$ in every objective and is strictly better in at least one. A design is considered *Pareto optimal* if no other design in $\mathcal{X}$ Pareto dominates it. The collection of all such Pareto optimal designs forms the *Pareto set* (PS), and the corresponding set of objective vectors,

$$\{\boldsymbol{f}(\boldsymbol{x}) \mid \boldsymbol{x} \in \text{PS}\} \,, \tag{6}$$

is known as the *Pareto front* (PF). The overarching aim in MOO is to obtain a diverse set of solutions that closely approximates the PF, thereby capturing the best possible trade-offs among the objectives. Analogous to the single-objective case, a naive approach involves modeling each of the m objectives with separate surrogate models and combining their predictions through a weighted sum to compute the gradient (Ma et al., 2020). However, this approach fails to account for conflicts among objectives and is also susceptible to *objective hacking*. We discuss approaches for addressing these issues in Section 4.

Generative Modeling In addition to the surrogate modeling discussed above, generative modeling is another key ingredient in offline optimization. In fact, offline optimization methods can be viewed through the lens of *conditional generation*, where the objective is to model the distribution $p(\boldsymbol{x} \mid \boldsymbol{y}_c)$ with $\boldsymbol{y}_c$ representing the desired conditions. By applying Bayes' rule, this distribution can be decomposed as

$$p(\boldsymbol{x} \mid \boldsymbol{y}_c) \propto p(\boldsymbol{x})\, p(\boldsymbol{y}_c \mid \boldsymbol{x})\,. \tag{7}$$

From this perspective, offline optimization methods generally fall into two categories: *inverse* and *forward*. Inverse methods directly train a conditional generative model for $p(\boldsymbol{x} \mid \boldsymbol{y}_c)$. For example, MIN (Brookes et al., 2019) employs a GAN-based inverse mapping from $\boldsymbol{y}$ to $\boldsymbol{x}$ to generate designs that meet the desired specifications. Forward methods, on the other hand, leverage a surrogate model for $p(\boldsymbol{y}_c \mid \boldsymbol{x})$ to guide an unconditional generative model $p(\boldsymbol{x})$. For instance, ROMA (Yu et al., 2021) computes gradients in the latent space of a VAE to iteratively refine generated designs. Similarly, gradient-based approaches such as COMs (Trabucco et al., 2021) and ICT (Yuan et al., 2023) fall into this category, although many of these methods do not explicitly model the generative component $p(\boldsymbol{x})$.

Since the literature on offline optimization often integrates surrogate and generative modeling, we separate their discussion in this review to provide clearer insights into each component: *surrogate modeling* is discussed in Section 4, while *generative modeling* is covered in Section 5.

Comparison between SOO and MOO Both single-objective optimization (SOO) and multi-objective optimization (MOO) strive to optimize objectives using only an offline dataset, which leads to some inherent similarities. In both settings, surrogate models are built to approximate the objective function(s), and generative models are employed to explore the design space efficiently.

However, significant methodological differences emerge due to their distinct optimization goals. One major distinction lies in the training of surrogate models. In SOO, the surrogate is typically trained to minimize the prediction error for a single objective while incorporating relevant priors, thereby facilitating direct gradient-based optimization. In contrast, MOO must capture the interdependencies among multiple objectives, often by leveraging their relationships to enhance surrogate modeling. This challenge usually necessitates the use of multi-task learning strategies (Chen et al., 2018; Yu et al., 2020).

Another fundamental difference is the sampling strategy. In SOO, the focus is on a single property, which simplifies conditional generation to either computing the gradient of that property (Chen et al., 2023a) or building an inverse mapping from the property to the design (Kumar & Levine, 2020). These approaches, however, are not directly applicable to MOO, where improvements in one objective might lead to the deterioration of another. Consequently, MOO relies on *Pareto-aware* sampling strategies to navigate the trade-offs among conflicting objectives. For example, Paretoflow (Yuan et al., 2024a) assigns uniform weight vectors to different objectives to guide flow matching towards the Pareto front.

In summary, while SOO and MOO share foundational principles and both contend with out-of-distribution issues, MOO introduces additional challenges related to balancing competing objectives, thereby requiring specialized modeling and optimization strategies.

Relation with Online Optimization (Active Learning). *Online optimization* (often referred to as *active learning* in this context) involves iteratively querying an expensive ground-truth objective function for new design points and then using these new data points to update a surrogate model (Jain et al., 2022; Gruver et al., 2023; Frey et al., 2025). In high-cost domains such as drug discovery—where each experiment (e.g., a clinical trial) can be prohibitively expensive (Angermüller et al., 2020)—these queries must be made

sparingly under a limited budget. The crux of online optimization is to strategically select new points that are both likely to yield high rewards and carry substantial uncertainty, thereby maximizing information gain and accelerating the search.

Bayesian models, such as Gaussian Processes (GPs) (Williams & Rasmussen, 1995; Rasmussen & Williams, 2006), are commonly used for online optimization because they provide principled *uncertainty quantification*. However, GPs have cubic complexity in the number of data points, which becomes infeasible for large datasets. As a result, deep neural networks equipped with approximate uncertainty estimation techniques (e.g., Monte Carlo dropout or ensembles) are often employed, though their uncertainty estimates can be less reliable in practice (Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017).

In offline optimization, there is no ability to query the objective after an initial dataset has been collected—effectively a *zero-round* version of active learning. Since no further data can be acquired, offline methods often avoid high-uncertainty regions where the surrogate model might be highly inaccurate or risky. Consequently, offline optimization algorithms often adopt more conservative strategies, prioritizing regions of the design space that are both high reward and well-understood according to the existing dataset.

Despite these differing strategies—online seeking to reduce uncertainty versus offline avoiding it—both paradigms rely on accurate and scalable uncertainty quantification. Scalable methods for uncertainty estimation can thus benefit both fields. We highlight two candidate methods: Neural Processes, which circumvent the cubic complexity of GPs (Garnelo et al., 2018), and GFlowNets, which can offer amortized Bayesian posterior inference over the parameter space to estimate model uncertainty (Bengio et al., 2023).

Futhermore online and offline approaches can synergize naturally. In the online phase, the goal is to collect a high-quality dataset by judiciously querying promising and uncertain regions. Once the budget for queries is exhausted and the dataset becomes static, offline optimization can then leverage the collected data for a final decision. In this way, online optimization focuses on acquiring the most informative dataset possible, while offline optimization focuses on extracting the best solution from the available data.

3 Benchmark

In this section, we present a systematic overview of benchmarks in offline MBO by first categorizing the available tasks in Section 3.1 and then discussing the evaluation metrics in Section 3.2.

3.1 Task

We begin by grouping tasks into four main categories: (1) *synthetic function*, which leverages closed-form mathematical functions to provide efficient, scalable, and analytically transparent benchmarks (see Section 3.1.1); (2) *real-world system*, which addresses practical engineering challenges in domains such as robotics (see Section 3.1.2); (3) *scientific design*, encompassing applications in biology, chemistry, and material science (see Section 3.1.3); and (4) *machine learning model*, which includes problems like neural architecture design (see Section 3.1.4). Evaluation costs tend to rise—and our understanding of the underlying mechanisms tends to diminish—as we move from category (1) to category (3). We address category (4) separately, reflecting its increasing significance within the ML community.

It is important to note that these categories are not mutually exclusive; some benchmarks may belong to more than one category. For example, Ehrlich (Stanton et al., 2024) can be viewed as a *synthetic function* task for biological sequence optimization, yet we mainly discuss it under *scientific design* to better reflect its application context. Each subsubsection first discusses the intrinsic advantages and limitations of its task category and then introduces the commonly used tasks within that category. For each task, we report its *size* (i.e., the offline dataset), *space* (design space), *dimension* (design dimension), *# Obj* (number of objectives), and *oracle* (evaluation oracle type), as summarized in Tables 1, 2, 3, and 4. This organization offers a balanced view of both the theoretical appeal and practical applicability of each task category.

3.1.1 Synthetic Function

Synthetic function tasks use closed-form mathematical functions to generate offline datasets for offline MBO methods. During evaluation, these same functions serve as oracles to benchmark these methods. These functions can, in principle, also be applied in an online context; however, in this discussion they serve as benchmarks in the offline setting, where algorithms are limited to using only the offline dataset.

Advantages (1) *Computational efficiency*: With closed-form expressions, these functions are quick to compute, allowing for extensive experimentation. (2) *Scalability*: Their ability to be defined for arbitrary input dimensions and numbers of objectives makes them highly adaptable for evaluating algorithms on large-scale and many-objective problems. (3) *Analytical transparency*: The known analytical forms enable exact computation of the Pareto front along with other key properties like gradients and Hessians, which helps in thoroughly understanding both the problem landscape and the behavior of optimization algorithms.

Limitations (1) *Lack of realism*: The inherent simplicity and often smooth nature of synthetic functions can fail to capture the complexities of real-world problems, including discontinuous or highly constrained landscapes. For example, in protein sequence design, a single amino acid change can cause a dramatic, discontinuous shift in properties, and the design space itself is highly constrained by biological functionality (Angermüller et al., 2020; Jain et al., 2022). Consequently, performance on synthetic tasks may not directly translate to practical applications. (2) *Limited adaptability for deep learning*: Deep neural networks have achieved significant success in areas such as image (Krizhevsky et al., 2012), language (Brown et al., 2020), and molecule (Jumper et al., 2021), and recent work has extended these models to offline MBO (Chen et al., 2023b; Watson et al., 2023). However, synthetic data generated by mathematical functions often lack the rich, complex patterns found in real-world data (e.g., molecular structures), which prevents deep learning methods from fully demonstrating their potential in these benchmarks.

Below, we introduce specific synthetic function tasks for both single-objective and multi-objective optimization, covering most tasks from commonly used benchmarks such as the BayesO benchmark (Kim, 2023; Surjanovic & Bingham, 2013), holo-bench (Stanton et al., 2024), pymoo (Blank & Deb, 2020), and Off-MOO-Bench (Xue et al., 2024). We describe each synthetic function class in Table 1. Due to the nature of synthetic function, the evaluation oracle is *analytical*, meaning that the mathematical form is known. Furthermore, the offline dataset size can be arbitrarily large (*Any*); however, researchers typically use defaults such as 50 000 (Chen et al., 2024) for SOO or 60 000 for MOO (Xue et al., 2024; Yuan et al., 2024a) (as indicated in parentheses). Similarly, many synthetic functions offer the flexibility to operate over *any* number of dimensions or objectives, and conventional practice often employs specific values (also indicated in parentheses). Finally, the design space in synthetic functions is generally *continuous*.

Single-Objective Optimization The SOO benchmarks are designed to assess algorithm performance across various landscapes with *many local minima*, *bowl-shaped*, *plate-shaped*, *valley-shaped*, *steep ridges/drops*, and more, each posing unique challenges to optimization (Surjanovic & Bingham, 2013).

- **Many local minima** Functions with many local minima are designed to challenge an algorithm's exploration capabilities by presenting a rugged landscape filled with numerous suboptimal traps. Common examples in this category include Ackley (Bäck, 1996), Bukin 6 (Jamil & Yang, 2013), Cosines (Kim & Choi, 2023), Drop-Wave (Jamil & Yang, 2013), Eggholder (Jamil & Yang, 2013), Gramacy & Lee, Griewank (Gramacy & Lee, 2010), Holder Table (Jamil & Yang, 2013), Kim1 (Kim & Choi, 2023), Levy (Jamil & Yang, 2013), Rastrigin (Jamil & Yang, 2013), and Shubert (Jamil & Yang, 2013).
- **Bowl-shaped** Bowl-shaped functions with a convex landscape offer a comparatively smooth and unimodal environment. Their gradual curvature makes them ideal for assessing the convergence speed and stability of an algorithm under benign conditions. Representative functions in this class include Bohachevsky (Jamil & Yang, 2013), Kim2 (Kim & Choi, 2023), Kim3 (Kim & Choi, 2023), and Sphere (Picheny et al., 2013).
- **Plate-shaped** Plate-shaped functions are characterized by expansive flat regions or plateaus where gradients are nearly zero over large areas of the search space. Although dedicated plate-shaped functions

Table 1: Overview of the *synthetic function* tasks.

Task Name	**Size**	**Space**	**Dimension**	**# Obj**	**Oracle**
Many local minima	Any (50000)	Continuous	1 - Any (60)	1	Analytical
Bowl-shaped	Any (50000)	Continuous	2 - Any (60)	1	Analytical
Plate-shaped	Any (50000)	Continuous	Any (60)	1	Analytical
Valley-shaped	Any (50000)	Continuous	2 - Any (60)	1	Analytical
Steep ridges/drops	Any (50000)	Continuous	2	1	Analytical
Other SOO tasks	Any (50000)	Continuous	2 - 6	1	Analytical
DTLZ	Any (60000)	Continuous	Any (7 - 10)	Any (3)	Analytical
Omnitest	Any (60000)	Continuous	2	2	Analytical
VLMOP	Any (60000)	Continuous	1 - 6	2 - 3	Analytical
ZDT	Any (60000)	Continuous	10 - 30	2	Analytical

are less prevalent in standard benchmarks, some tasks like Zakharov (Jamil & Yang, 2013) exhibit plateau-like regions.

- **Valley-shaped** Valley-shaped functions possess a narrow, elongated channel leading to the global optimum. This class—exemplified by the Three-Hump Camel (Jamil & Yang, 2013), Six-Hump Camel (Jamil & Yang, 2013), and Rosenbrock (Picheny et al., 2013)—tests an algorithm's precision and its ability to efficiently traverse an ill-conditioned, elongated basin without overshooting the target.
- **Steep ridges/drops** Functions characterized by steep ridges or abrupt drops exhibit sudden, dramatic changes in the objective landscape. Representative examples include De Jong 5 (Jamil & Yang, 2013), Easom (Jamil & Yang, 2013), and Michalewicz (Jamil & Yang, 2013).
- **Others** There are some synthetic benchmark functions that do not neatly fit into the previously discussed classes but are nonetheless widely used in single-objective optimization. Examples include Beale (Jamil & Yang, 2013), Branin (Picheny et al., 2013), Colville (Jamil & Yang, 2013), Goldstein-Price (Picheny et al., 2013), Hartmann 3D (Jamil & Yang, 2013), and Hartmann 6D (Jamil & Yang, 2013). Each of them exhibits unique landscape features such as nonlinearity, moderate modality, or intricate curvature.

Multi-Objective Optimization In addition to SOO tasks, many synthetic functions have been developed for the MOO setting, where algorithms often optimize multiple conflicting objectives simultaneously. We discuss four families of synthetic functions for MOO—*DTLZ*, *Omnitest*, *VLMOP*, and *ZDT*—each characterized by distinct Pareto front properties and varying levels of complexity.

- **DTLZ** The DTLZ family (Deb et al., 2005) is designed to be scalable with respect to both the number of objectives and the decision variables, making them versatile for testing algorithms under various conditions. For example, DTLZ1 features a linear Pareto front, while DTLZ2 through DTLZ7 introduce different degrees of nonlinearity, multimodality, and deceptive features.
- **Omnitest** Omnitest typically involves two objectives and presents a challenging landscape marked by convexity and multiple local optima (Deb & Tiwari, 2008).
- **VLMOP** These functions (van Veldhuizen & Lamont, 1999) are characterized by intricate, concave or disconnected Pareto fronts, which challenge algorithms to capture a diverse set of trade-off solutions.
- **ZDT** Functions in the ZDT family exhibit diverse Pareto front geometries, from convex to concave and even disconnected fronts (Zitzler et al., 2000). For example, ZDT1 and ZDT4 feature convex Pareto fronts, whereas ZDT2 and ZDT6 exhibit concave ones. Unlike other functions in the ZDT family, ZDT3 has a disconnected Pareto front.

3.1.2 Real-World System

Real-world system tasks encapsulate the inherent complexity of practical engineering design. While many of these tasks still utilize analytical functions for construction, unlike the purely synthetic functions, they are derived from models of real-world systems (Tanabe & Ishibuchi, 2020). Alternatively, some tasks employ simulations (Brockman et al., 2016) or surrogate models (Tanabe & Ishibuchi, 2020) as oracles. Overall, compared to the *synthetic function* category, these tasks incur higher evaluation costs and provide less transparency into the underlying mechanisms.

Advantages (1) *Enhanced realism*: These tasks are constructed from real engineering systems, and they incorporate real-world constraints such as physical feasibility, manufacturability, energy consumption, and cost, thereby faithfully representing practical design challenges. (2) *Heterogeneous input representation*: The design spaces often involve a mix of continuous, categorical, and permutation-based variables, mirroring the multifaceted nature of real-world problems. This variety can more effectively test offline MBO algorithms.

Limitations (1) *Reduced analytical transparency*: While analytical functions offer closed-form expressions, real-world system tasks often rely on simulations or surrogate models that obscure the underlying mechanisms. (2) *High evaluation cost*: For some simulation tasks, computational expenses tend to be high, which may limit the number of feasible experiments (Brockman et al., 2016). (3) *Limited adaptability for deep learning*: Although these tasks are more realistic than pure synthetic functions, their design is often quite basic (Tanabe & Ishibuchi, 2020) and may still lack the rich, intricate patterns necessary for deep learning methods to fully demonstrate their potential, similar to the *synthetic function* tasks.

Below we introduce *real-world system* tasks. Due to the wide range of tasks, classifying them by application area proves challenging. Therefore, we propose to categorize these tasks based on their oracle type—*analytical*, *simulation*, and *surrogate*—which we believe better reflects the inherent characteristics of each task.

Analytical For some real-world system tasks, the underlying mechanisms are well understood and can be readily represented by mathematical functions. These functions serve as analytical oracles. For example, the RE2-4-1 task in the *RE suite* (Tanabe & Ishibuchi, 2020) involves minimizing both the structural volume and the joint displacement of a four-bar truss. This problem is formulated based on Newtonian physics, allowing the oracle function to be derived analytically. Other analytical tasks from the RE suite—such as various truss designs (Cheng & Li, 1999; Coello Coello & Pulido, 2005; Qian et al., 2025), reinforced concrete beams (Amir & Hasegawa, 1989), pressure vessels (Kannan & Kramer, 1994), hatch covers (Amir & Hasegawa, 1989), coil compression springs (Lampinen & Zelinka, 2000), welded beams (Ray & Liew, 2002), disc brakes (Ray & Liew, 2002), speed reducers (Farhang-Mehr & Azarm, 2002), gear trains (Deb & Srinivasan, 2006), conceptual marine design (Parsons & Scott, 2004), and water resource planning (Ray et al., 2001)—fall into this category.

The *industrial suite* benchmarks (Qian et al., 2025) further enrich this category, which are formulated as constrained optimization problems. For instance, the optimal operation task targets improvements in chemical processing quality, where the objective is to enhance the alkylating product subject to 14 constraints designed to limit onboard fuel and launcher performance. The process flow sheeting problem and process synthesis problem focus on efficient process design under practical constraints. Both objective functions and constraints are expressed mathematically for these tasks. Besides, some classic combinatorial optimization problems in industrial applications—such as the multi-objective traveling salesman problem (Lust & Teghem, 2010), the multi-objective capacitated vehicle routing problem (Zajac & Huber, 2021a), and the multi-objective knapsack problem (Ishibuchi et al., 2015)—have analytical formulations that allow for rapid computation. However, these problems are not typically considered traditional design problems.

Simulation Sometimes, even when the underlying mechanisms of a task are well understood, their complexity prevents us from expressing the oracle in a closed analytical form. In such cases, we rely on numerical simulations to obtain the oracle, despite the high computational cost often involved. A typical example is *robot morphology design*. Robot morphology design tasks focus on optimizing the structural parameters of simulated robots to maximize performance on specific tasks, as exemplified by the *Ant* and *D'Kitty* Morphology tasks (Trabucco et al., 2022). In the Ant Morphology task, the objective is to optimize 60

continuous parameters—encoding limb size, orientation, and location—for a quadruped robot from OpenAI Gym (Brockman et al., 2016) so that it runs as fast as possible. In the D'Kitty Morphology task, the goal is to adjust 56 continuous parameters defining the D'Kitty robot from ROBEL (Ahn et al., 2020) to enable a pre-trained, morphology-conditioned neural network controller, optimized using Soft Actor Critic (Haarnoja et al., 2018), to navigate the robot to a fixed location. Both tasks utilize the MuJoCo simulation (Todorov et al., 2012) to run simulations for 100 time steps and average the results over 16 independent trials, thereby providing reliable yet computationally efficient performance estimates.

This category also includes tasks for *electronics design*. These tasks focus on optimizing hardware configurations and accelerator architectures to meet specific performance and efficiency objectives, typically involving selecting parameters for processing elements, memory hierarchies, and dataflow patterns, directly influencing factors such as latency (Kumar et al., 2022). The simulator oracle is used to evaluate the performance and feasibility of accelerator designs by providing latency, power, and other metrics. Other simulation tasks include radar waveform design (Hughes, 2007) for signal processing, heat exchanger (Daniels et al., 2018) and car structure design (Kohira et al., 2018) for engineering optimization, TopTrumps (Volz et al., 2019) problem for game-based optimization, and MarioGAN (Volz et al., 2019) for procedural content generation. These tasks provide valuable and diverse benchmarks for offline MBO.

Table 2: Overview of the *real-world system* tasks.

Task Name	**Size**	**Space**	**Dimension**	**# Obj**	**Oracle**
RE suite	Any (60000)	Mixed	3 - 7	2 - 9	Analytical / Surrogate
Industrial suite	Any (2000 - 7000)	Continuous	2 - 7	1	Analytical
Ant / D'Kitty	25009	Continuous	60 / 56	1	Simulation
Electronics design	8000	Categorical	10	1	Simulation

Surrogate Some real-world system tasks utilize surrogate models to approximate the oracle based on simulation data, thereby reducing the cost of simulations or actual experiments. For instance, several tasks within the RE suite (Tanabe & Ishibuchi, 2020)—including vehicle crashworthiness (Liao et al., 2008), car side impact design (Jain & Deb, 2014), rocket injectors (Vaidyanathan et al., 2003), and car cab design (Deb & Jain, 2014)—employ surrogate oracles. The surrogate parameters are determined using the response surface method on data sampled from simulations. As a result, these problems differ from their original counterparts, and the inherent approximations may introduce bias that could affect the reliability of performance assessments.

3.1.3 Scientific Design

In contrast to the engineering problems discussed earlier, *scientific design* tasks focus on addressing fundamental scientific questions. In the context of offline MBO, these tasks span domains such as biology, chemistry, and materials science, with the aim of discovering novel designs with desired properties. The offline setting is particularly relevant here, as conducting online experiments for these designs is often prohibitively expensive.

Advantages (1) *Realistic challenge*: Scientific design tasks emulate real-world discovery processes—such as protein sequence design, molecular optimization, and material discovery—that are sufficiently complex to differentiate between various offline MBO methods. Successful optimization in these domains can significantly accelerate scientific discovery and benefit humankind. (2) *Deep learning benchmarking*: Scientific design problems, including those in biological (Jumper et al., 2021) and chemical (Kuenneth & Ramprasad, 2023), exhibit rich and meaningful patterns. These characteristics provide a robust platform to reveal the potential of deep learning-based offline MBO methods.

Limitations (1) *Inaccurate oracles*: Unlike synthetic functions with well-defined analytical forms, scientific design tasks often rely on approximate oracles—such as surrogate models or physics-based simulations—resulting in less convincing evaluations. (2) *Limited scalability*: The size of offline datasets in these tasks is typically quite small, making it difficult to assess the scalability of some methods. Additionally, the

predetermined nature of the problem settings (e.g., fixed search spaces and objective counts) limits flexibility in comprehensively benchmarking offline MBO methods.

Below, we present specific scientific design tasks across various domains—namely *biology*, *chemistry*, and *materials science*. As summarized in Table 3, these benchmarks offer diverse and challenging scenarios for offline MBO, reflecting the complexity of real-world scientific problems. Given the inherent nature of scientific design, the design space is typically discrete, and the evaluation oracles are implemented via *surrogate*, *simulation*, *analytical*, or *lookup table*.

Biology The *biology* category encompasses a diverse set of tasks aimed at optimizing biological sequences to enhance their performance. The primary targets in this category are *protein*, *DNA*, and *RNA* designs. Together, these tasks provide a rich testbed for evaluating the efficacy and robustness of offline MBO methods in the life sciences. Below, we describe each design in detail.

- **Protein** Proteins are sequences composed of 20 standard amino acids, and optimizing these sequences can lead to improved properties with applications in antibody development (Luo et al., 2022; Chen et al., 2025) and enzyme engineering (Hua et al., 2024). While many tasks have been proposed for protein design (see, e.g., YE et al. (2025)), here we focus specifically on those that address sequence design within the framework of offline MBO.

  (1) *Green Fluorescent Protein* (GFP): Optimizes a 237-length protein sequence to increase fluorescence, using a dataset of 56, 086 variants (Sarkisyan et al., 2016a). (2) *5' Untranslated Region* (UTR) : Designs a 50-nucleotide sequence to maximize gene expression, predicting ribosome load based on 280, 000 sequences (Sample et al., 2019a). (3) *Adeno-Associated Virus* (AAV): Optimizes a 28-amino acid segment of the VP1 protein using 284, 000 variants, targeting improved viral viability (Ogden et al., 2019). (4) *E4B Ubiquitination Factor* (E4B): Enhances ubiquitination activity by optimizing 100, 000 mutations (Starita et al., 2013). (5) *TEM-1 β-Lactamase* (TEM): Aims to improve thermodynamic stability, leveraging 17, 857 sequences (Ren et al., 2022). (6) *Aliphatic Amide Hydrolase* (AMIE): Focuses on boosting enzyme activity, using 6, 629 variants (Wrenbeck et al., 2017). (7) *Levoglucosan Kinase* (LGK): Targets enzyme performance improvements, based on 7, 891 mutants (Klesmith et al., 2015). (8) *Poly(A)-Binding Protein 1* (Pab1): Optimizes RNA binding efficiency with over 36, 000 sequences (Melamed et al., 2013). (9) *SUMO E2 Conjugase* (UBE2I): Focuses on protein function optimization, based on 2, 000 variants (Weile et al., 2017). (10) *Regex*: Modifies sequences to maximize bigram counts, simulating sequence editing operations (Stanton et al., 2022). (11) *Affinity maturation* (Affinity): Mutates antibody to maximize binding affinity with the antigen (Chen et al., 2025). (12) *Red Fluorescent Protein* (RFP): A multi-objective optimization problem balancing stability and solvent-accessible surface area (Stanton et al., 2022).

  In general, most protein properties are evaluated using *surrogate* oracles trained on large supervised datasets due to the high cost of direct evaluation. For some properties—such as binding affinity in (11) and stability in (12)—*simulation* oracles (e.g., Rosetta (Alford et al., 2017) or FoldX (Schymkowitz et al., 2005)) can be employed, though these simulations often lack sufficient accuracy. Similarly, properties like bigram counts can be computed using *analytical* oracles; however, such tasks tend to be relatively trivial.

- **DNA** DNA design involves the optimization of sequences comprising four nucleotides (A, C, G, and T). (1) *Transcription Factor Binding 8* (TFB8) : Optimizes transcription factor binding activity in an 8-length sequence space (Barrera et al., 2016a). (2) *Transcription Factor Binding 10* (TFB10): Extends the optimization to a 10-length sequence space, aiming for higher binding affinity (Barrera et al., 2016a). Trabucco et al. (2022) further process the data to ensure training set given to offline MBO methods is restricted to the bottom 50%. Oracle evaluation is performed via a *lookup table* oracle, as the properties of all possible sequences have been pre-measured.

- **RNA** RNA design involves the optimization of sequences comprising four nucleobases (A, U, C, and G). The *RNA-Binding* task is a typical example. The objective is to optimize a 14-length RNA sequence to maximize binding activity with a target transcription factor. Kim et al. (2023) adopt three target transcriptions, termed RNA-Binding-A (for L14 RNA1), RNA-Binding-B (for L14 RNA2), and RNA-Binding-C (for L14 RNA3). These properties are evaluated using the ViennaRNA package as a *simulation* oracle (Lorenz et al., 2011).

In addition to the above tasks, Stanton et al. (2024) proposes *Ehrlich* functions, a new class of closed-form test functions for biophysical sequence optimization. These functions are formulated to encapsulate key geometric properties—such as non-additivity, epistasis, and discrete feasibility constraints—that are characteristic of real-world sequence design problems. Their provable solvability, low evaluation cost, and capacity to mimic realistic biophysical interactions make Ehrlich functions a valuable benchmark in offline MBO and a promising direction for future research.

Table 3: Overview of the *scientific design* tasks.

Task Name	**Size**	**Space**	**Dimension**	**# Obj**	**Oracle**
GFP	56086	Categorical	237	1	Surrogate
UTR	280000	Categorical	50	1	Surrogate
AAV	284000	Categorical	28	1	Surrogate
E4B	100000	Categorical	102	1	Surrogate
TEM	17857	Categorical	286	1	Surrogate
AMIE	6629	Categorical	341	1	Surrogate
LGK	7891	Categorical	439	1	Surrogate
Pab1	36000	Categorical	75	1	Surrogate
UBE2I	2000	Categorical	159	1	Surrogate
Regex	42048	Categorical	73	3	Analytical
Affinity	4158	Categorical	N/A	1	Simulation
RFP	4937	Categorical	489	2	Simulation
TFB8	32898	Categorical	8	1	Lookup Table
TFB10	50000	Categorical	10	1	Lookup Table
RNA-Binding	5000	Categorical	14	1	Simulation
Ehrlich	Any (50000)	Continuous	7	1	Analytical
ChEMBL	1093	Categorical	31	1	Surrogate
ZINC	48000	Categorical	257	2	Analytical
Molecule	49001	Continuous	32	2	Surrogate
Superconductor	21263	Continuous	86	1	Surrogate

Chemistry The *chemistry* category comprises tasks focused on small-molecule design, in contrast to the large-molecule design tasks typical in the biology category. Numerous benchmarks for small-molecule design exist (Brown et al., 2019; Huang et al., 2021); in this paper, we restrict our discussion to those commonly used in offline MBO. Some typical examples include: (1) ChEMBL: Derived from a large-scale drug property database (Gaulton et al., 2011), this task aims to maximize molecular activity by optimizing the MCHC value associated with assay CHEMBL3885882. It employs a training set of $1,093$ molecules and a discrete design space of 31-length sequences over 591 categorical values. This benchmark has been incorporated in Trabucco et al. (2022), where a random forest surrogate is used as the oracle. (2) ZINC: This benchmark provides a multi-objective molecule optimization problem involving a small molecule of roughly 128 tokens. The goal is to improve *analytical* druglikeness properties such as logP and QED (Stanton et al., 2022). (3) Molecule: As described in Zhao et al. (2022), this task tackles a two-objective molecular generation problem, aiming to optimize activities against the biological targets GSK3β and JNK3 in a 32-dimensional continuous latent space. Candidate solutions are decoded into molecular strings using a pre-trained decoder.

Material Science Unlike biological tasks focusing on large molecules such as proteins, and chemical tasks centering on small molecules, the *material science* category is dedicated to designing and optimizing materials based on complex compositional and structural performance metrics. A typical offline MBO benchmark in this domain is *Superconductor*. Derived from a dataset of 21,263 superconductors with annotated critical

temperatures (Hamidieh, 2018), it has also been incorporated into design-bench (Trabucco et al., 2022). The goal is to maximize the critical temperature by optimizing an 86-dimensional continuous representation of material composition, with a random forest surrogate serving as the evaluation oracle.

Some less common tasks include:

- **Crystal structure prediction**, which aims to identify stable crystal configurations for a given chemical composition (Qi et al., 2023). These configurations are represented through features such as lattice parameters and fractional coordinates, and a *simulation* serves as the oracle to estimate formation energies.
- **Battery materials design**, where research efforts focus on discovering better active materials for lithium-ion batteries (Valladares & Others, 2021).

3.1.4 Machine Learning Model

We have discussed three groups of tasks: *synthetic function*, *real-world system*, and *scientific design*. In addition, a widely adopted task in the ML community is the design of *machine learning models*. This process typically involves optimizing architectures, model parameters, and hyperparameters. The evaluation is generally based on the final performance metrics obtained on a test set.

Advantages (1) *Practical relevance*: Enhancements in model architectures, parameters, or hyperparameters can yield significant performance improvements, with applications across deep learning, reinforcement learning, and more. (2) *Rich offline datasets*: In contrast to scientific design tasks, machine learning models often have access to extensive pre-collected datasets, enabling more robust benchmarking. (3) *Oracle evaluation*: In ML tasks, oracle evaluation is performed either via *lookup table*, *surrogate*, or through *real experiment*, offering more accurate assessments compared to the approximate evaluations typical in scientific tasks.

Limitations (1) *Limited search space*: Although many neural architecture benchmarks provide lookup tables, the available search spaces are often constrained and overly simplistic. In contrast, real-world architectures tend to be much larger and more complex, leading to higher evaluation costs. (2) *Oracle variability*: The inherent stochasticity in training deep learning models can introduce significant noise in performance metrics, which may obscure true improvements and complicate benchmarking.

Table 4 summarizes key characteristics of these tasks, covering common benchmarks used in offline MBO research. We categorize these tasks into three major subcategories—*model architecture*, *model parameter*, and *model hyperparameter*—which are described in detail below.

Model Architecture *Model architecture* design, often referred to as *neural architecture search*, seeks to automatically discover high-performing network architectures by exploring vast, discrete search spaces.

- Early work (Zoph & Le, 2017) focuses on single-objective optimization and evaluate the discovered architectures via *real experiment*. The goal is to identify a 32-layer convolutional neural network with residual connections that maximizes test accuracy on CIFAR-10. The search space is defined over architectural hyperparameters such as kernel sizes, selected from $\{2, 3, 4, 5, 6\}$, and activation functions including ReLU, ELU, leaky ReLU, SELU, and SiLU, resulting in a 64-dimensional discrete space with 5 categories per dimension.
- Multi-objective NAS (MO-NAS) extends the search paradigm by simultaneously optimizing multiple performance metrics of neural architectures (Lu et al., 2023). In many MO-NAS frameworks, architectures are pre-trained and their performances recorded in *lookup tables*, enabling rapid evaluation. Typical objectives include prediction error, model complexity (e.g., number of parameters), and hardware efficiency metrics (e.g., GPU latency or FLOPs). By jointly optimizing these criteria, MO-NAS aims to identify architectures that achieve an optimal balance among accuracy, computational cost, and hardware efficiency. Representative benchmarks include *NAS-Bench-201-Test* (Krizhevsky, 2009) as well as the *C-10/MOP* and *IN-1K/MOP* suites (Lu et al., 2023).

Table 4: Overview of the *machine learning model* tasks.

Task Name	Size	Space	Dimension	# Obj	Oracle
NAS	1771	Categorical	64	1	Real Experiment
NAS-Bench-201-Test	9375	Categorical	6	3	Lookup Table
C-10/MOP	9375 - 60000	Categorical	5 - 32	2 - 8	Lookup Table
IN-1K/MOP	60000	Categorical	21 - 34	2 - 4	Lookup Table
Hopper	3200	Continuous	5126	1	Simulation
MO-Swimmer	8571	Continuous	9734	2	Simulation
MO-Hopper	4500	Continuous	10184	2	Simulation
HPOBench	10000	Mixed	2-27	1	Lookup Table / Surrogate

Model Parameter *Model parameters* are typically optimized via loss functions and gradient-based methods. In the context of offline MBO, these parameters are treated as high-dimensional design variables, with a particular focus on agent policy parameters. This category of tasks centers on fine-tuning the neural network weights to enhance performance.

- The *Hopper* task (Trabucco et al., 2022) aims to design a feed-forward neural network controller for a 2D hopping robot in MuJoCo (Todorov et al., 2012) to maximize the expected discounted return. The design space encompasses thousands of continuous weight parameters. Crucially, in the offline setting, the algorithm only has access to a dataset of (policy parameters, return) pairs collected from prior RL experiments, rather than interacting directly with the environment. Consequently, the policy optimization becomes a purely data-driven, model-based optimization challenge rather than a traditional learning task.
- Extending this idea to multi-objective scenarios, Xue et al. (2024) introduce *MO-Swimmer* and *MO-Hopper*, where each environment presents two conflicting objectives. In MO-Swimmer, the trade-off is between forward velocity and energy efficiency, while in MO-Hopper, it is between forward velocity and jumping height. In both cases, the full set of neural network policy weights constitutes the search space, with data collected by Xu et al. (2020). The multi-objective policy search thus aims to identify new policies that can simultaneously balance these competing criteria—without any additional simulation calls.

Model Hyperparameter *Model hyperparameters*, such as learning rate, weight decay, and batch size, play a crucial role in controlling the training process. Numerous benchmarks including *HPOBench* (Eggensperger et al., 2021) exist for this task, and popular methods like Bayesian optimization (Frazier, 2018) have been widely applied. The objective is to identify the optimal hyperparameter configuration that maximizes a performance metric, typically modeled as a black-box function. To the best of our knowledge, few offline MBO methods have been developed for hyperparameter optimization. This may be because hyperparameter configurations typically involve simple scalar values, and data-driven offline MBO approaches—often relying on neural networks to capture complex design patterns—may not be ideally suited for such tasks.

3.2 Evaluation Metric

In this subsection, we present the *evaluation metrics* that underpin our benchmarking of offline MBO methods. Although we have described oracle evaluations when discussing the tasks, it is important to elaborate on the evaluation metrics within this context. The oracle evaluation provides a per-sample usefulness score; however, a complete assessment requires evaluating a set of samples to measure novelty and diversity. In the following, we discuss the metrics for *usefulness*, *novelty*, and *diversity*. We assume that a batch $\mathcal{B}$ of K candidates (e.g., $K = 128$) is used to evaluate these metrics.

3.2.1 Usefulness

The primary criterion is that the candidate set should contain high-performing solutions. In offline SOO, where only one property is considered, this *usefulness* criterion is straightforward. In contrast, in offline MOO,

usefulness is inherently coupled with *diversity*, as the quality of the solution set is evaluated based on both its proximity to the Pareto front and its distribution along that front. We discuss these two settings separately.

Single-Objective Optimization For SOO tasks, *usefulness* is evaluated by considering the normalized ground-truth scores of the candidates. Specifically, we examine both the score at the 100th percentile (i.e., the best design) and the 50th percentile (i.e., the median) as suggested in Trabucco et al. (2022). The normalized score is computed as:

$$y_n = \frac{y - y_{\min}}{y_{\max} - y_{\min}},$$

where y denotes the oracle evaluation score of a design, and $y_{\min}$ and $y_{\max}$ are the minimum and maximum scores in the offline dataset, respectively. Additionally, following Jain et al. (2022), the overall performance of a candidate set $\mathcal{B}$ is also quantified by its mean score:

$$\text{Mean}(\mathcal{B}) = \frac{\sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{B}} y_i}{|\mathcal{B}|},$$

where y_i is the oracle evaluation for $\boldsymbol{x}_i$. This metric provides an aggregate measure of the average quality of the candidate designs.

Multi-Objective Optimization In the MOO setting, *usefulness* is assessed by evaluating both the proximity of the candidate set $\mathcal{B}$ to the true Pareto front and its distribution along that front, a metric that naturally also captures *diversity*. Two widely used metrics are the *hypervolume* (HV) (Zitzler & Thiele, 1998) and the *inverted generational distance* (IGD) (Bosman & Thierens, 2003).

The HV metric quantifies the size of the objective space that is dominated by the candidate set $\mathcal{B}$ and bounded by a reference point $\boldsymbol{r} = (r^1, r^2, \ldots, r^m)$. The reference point is typically chosen to be worse than any observed objective value (i.e., a nadir point). Mathematically, the HV is defined as:

$$HV(\mathcal{B}) = \text{vol}\left(\bigcup_{\boldsymbol{y} \in \mathcal{B}} \prod_{i=1}^{m} [y^i, r^i] \right),$$

where $\prod_{i=1}^{m} [y^i, r^i]$ represents an m-dimensional hyperrectangle (or box) spanning from the coordinates of $\boldsymbol{y}$ to the reference point $\boldsymbol{r}$ along each objective, and $\text{vol}(\cdot)$ denotes the Lebesgue measure (i.e., volume) of the union of these hyperrectangles. In simple terms, a larger hypervolume indicates that the solution set is both close to the Pareto front and well-distributed across the objective space.

The IGD metric measures the average distance from points on the true Pareto front (denoted as PF) to the nearest solution in the candidate set $\mathcal{B}$:

$$IGD(\mathcal{B}, PF) = \frac{1}{|PF|} \sum_{\boldsymbol{y}_{pf} \in PF} \min_{\boldsymbol{y} \in \mathcal{B}} \|\boldsymbol{y}_{pf} - \boldsymbol{y}\|.$$

Since the true Pareto front is generally unknown in real-world tasks, most existing studies primarily rely on HV to evaluate MOO performance (Tanabe & Ishibuchi, 2020; Xue et al., 2024; Yuan et al., 2024a).

3.2.2 Diversity

Diversity quantifies the spread or variability within the set of generated candidate designs (Jain et al., 2022; Kim et al., 2023; Kirjner et al., 2024). In offline MBO, it is crucial not only to identify high-performing designs but also to explore diverse regions of the design space, thereby capturing multiple modes of the black-box function. A common metric for measuring diversity is the average pairwise distance between all distinct candidates in the set $\mathcal{B}$, defined as:

$$\text{Diversity}(\mathcal{B}) = \frac{1}{|\mathcal{B}|(|\mathcal{B}| - 1)} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{B}} \sum_{\substack{(\boldsymbol{x}_j, y_j) \in \mathcal{B} \\ (\boldsymbol{x}_j, y_j) \neq (\boldsymbol{x}_i, y_i)}} \delta(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

where $\delta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ denotes a distance measure (e.g., the Euclidean distance for continuous designs or the Levenshtein edit distance (Haldar & Mukhopadhyay, 2011) for discrete designs). Alternatively, the median of the pairwise distances may be used instead of the mean (Kirjner et al., 2024). A higher diversity score indicates that, on average, the generated designs are more varied within the batch $\mathcal{B}$.

Another related metric is *coverage*, as adopted in Yao et al. (2025), which evaluates how well the batch of candidate designs collectively spans the search space. It is computed as:

$$\mathrm{L1C}(\mathcal{B}) = \frac{1}{d} \sum_{k=1}^{d} \max_{i \neq j} \Big| x_{ik} - x_{jk} \Big|,$$

where d is the number of design dimensions and x_{ik} denotes the k-th component of the i-th design $\boldsymbol{x}_i$. Note that this formulation assumes a continuous representation of the design; discrete designs must first be embedded into a continuous latent space. Intuitively, a higher coverage value indicates that the designs in $\mathcal{B}$ are more widely spread across each dimension, suggesting improved diversity.

3.2.3 Novelty

Novelty measures the degree to which the newly generated candidate set $\mathcal{B}$ differs from the offline dataset $\mathcal{D}$. In offline MBO, it is important that the proposed candidates not only perform well but also explore regions of the design space that are distinct from known designs. Novelty is typically quantified by computing the mean of the minimum distances between each candidate in $\mathcal{B}$ and the closest design in $\mathcal{D}$ (Jain et al., 2022; Kim et al., 2023). Mathematically, the novelty score is defined as:

$$\mathrm{Novelty}(\mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{B}} \min_{\boldsymbol{x} \in \mathcal{D}} \delta(\boldsymbol{x}_i, \boldsymbol{x}),$$

where $\delta(\boldsymbol{x}_i, \boldsymbol{x})$ denotes the distance metric between design $\boldsymbol{x}_i$ from $\mathcal{B}$ and a design $\boldsymbol{x}$ from $\mathcal{D}$. As before, the Euclidean distance is commonly used for continuous designs, while the Levenshtein edit distance (Haldar & Mukhopadhyay, 2011) is appropriate for discrete designs. Alternatively, one may use the median of the minimum distances instead of the mean (Kirjner et al., 2024). A higher novelty score indicates that, on average, the generated designs are more dissimilar from those in $\mathcal{D}$, thereby fostering the discovery of innovative and unexplored designs.

Overall Summary Most offline MBO methods (Trabucco et al., 2021; Chen et al., 2023a) primarily focus on the *usefulness* metric, as it is the fundamental measure of candidate quality. However, *diversity* and *novelty* are also important for a comprehensive evaluation. We conjecture that this focus is partly due to the relative ease of demonstrating improvements in usefulness, whereas proving enhancements across all three metrics for a proposed method is more challenging. We encourage future research to consider all metrics.

Besides, some recent work has proposed alternative metrics beyond the three discussed above. For instance, Qian et al. (2025) introduces a *stability* metric that measures an algorithm's ability to consistently surpass the performance of the offline dataset during the optimization process. Specifically, this metric evaluates not only the final design but also all intermediate samples along the optimization trajectory, thereby assessing whether these intermediate solutions can outperform the best design in the offline dataset—a critical consideration given the challenge of determining an appropriate stopping point in offline MBO.

4 Surrogate Modeling

As shown in Eq. (2), we typically employ the mean squared error loss to fit the surrogate model—a loss that can also be interpreted as a maximum likelihood loss when accounting for uncertainty (Chen et al., 2024). In this work, we decompose surrogate modeling into four key components: *input representation*, *model type*, *training strategy*, and *sampling strategy*.

4.1 Input Representation

The *input representation* refers to the type of design $\boldsymbol{x}$, which we categorize into *continuous*, *discrete*, and *mixed* representations. Since we focus on offline MBO, the ability to compute gradients of $\boldsymbol{x}$ is a vital property of the surrogate model. For discrete and mixed representations, a common strategy is to transform them into a continuous space to facilitate gradient computation.

Continuous *Continuous* representations are the most common and are often used directly in their native space due to their inherent continuity. For example, in the superconductor task, the search space is an 86-dimensional continuous vector representing the elemental composition of superconductors (Hamidieh, 2018). Other examples include Ant Morphology and D'Kitty Morphology, where the task is to optimize the morphology of quadrupedal robots using 60 and 56 continuous parameters respectively to enhance locomotion, and Hopper Controller, which involves optimizing a neural network policy with 5126 continuous weights to maximize return (Brockman et al., 2016). The continuous nature of these representations naturally facilitates gradient computation. An interesting case is image optimization; for instance, DeepDream optimizes continuous images using surrogate gradients to enhance specific features (Mordvintsev et al., 2015).

Even when using continuous inputs, it is common to map them into the latent space of a generative model for optimization and manipulation, as this latent space can better capture the semantic meaning of the design. For example, classifier (or surrogate) guidance optimizes the continuous image latent space, steering samples toward a specific category in diffusion (Dhariwal & Nichol, 2021) and flow matching models (Dao et al., 2023).

Discrete In many real-world applications, the design $\boldsymbol{x}$ is *discrete*. The most common representation is *categorical encoding*, which is suitable for unordered discrete variables. Typical applications include neural architecture search, biological sequence design, and molecule design. When computing surrogate gradients on categorically encoded inputs, three strategies are typically employed. First, one may remain in the raw discrete space and use discrete gradient estimators to approximate gradients (Bengio et al., 2013; Jang et al., 2017; Chen et al., 2023b). Second, the input can be mapped into a latent space where gradients are more directly accessible (Luo et al., 2018; Gómez-Bombarelli et al., 2018). Third, the design may be transformed into a continuous representation through design-specific modeling. For instance, Liu et al. (2019); Fu et al. (2021) propose continuous relaxations of discrete representations for neural architectures and chemical structures, respectively, to facilitate gradient computation.

A less common discrete representation is *permutation encoding*, which is used when the relative order of elements is crucial. This encoding is typical in problems such as the traveling salesman problem (Lust & Teghem, 2010) and the capacitated vehicle routing problem (Zajac & Huber, 2021b). However, these problems generally do not rely on surrogate models since their final solutions can be evaluated efficiently.

Mixed Some optimization problems involve a combination of continuous and discrete variables, leading to hybrid input representations that require domain-specific surrogate modeling. For instance, in protein property prediction, a protein is characterized by both continuous atom coordinates and discrete residue types. A practical surrogate model is to input the discrete residue types into a pre-trained language model to obtain residue embeddings, and then feed both the residue embeddings and atom coordinates into a graph neural network for the final prediction (Wang et al., 2022; Zhang et al., 2023; Chen et al., 2023c). In this scenario, gradient computation leverages strategies developed for both continuous and discrete representations.

4.2 Model Type

Surrogate models can be broadly classified based on their parameterization into *parametric* and *non-parametric* models. Parametric models have a fixed number of parameters determined by their architecture, whereas non-parametric models adapt their complexity based on the available data (Hastie et al., 2009). In the context of offline MBO, *neural networks* are the predominant parametric models, whereas *kernel-based models* are the typical non-parametric choices.

Neural Networks (NN) *Neural networks* are a class of parametric models where the architecture (e.g., the number of layers and neurons per layer) is predetermined prior to training (Salakhutdinov, 2014). These models approximate the target function by optimizing a fixed set of parameters, making them well-suited for high-dimensional and complex tasks. Owing to their generalization capabilities and scalability, NN-based surrogate models have become increasingly popular in offline MBO (Trabucco et al., 2022).

Some approaches use data-agnostic neural networks—such as multi-layer perceptrons (MLPs)—as surrogate models (Trabucco et al., 2021; Yu et al., 2021; Yuan et al., 2023; Chen et al., 2023a). While these methods demonstrate the overall effectiveness of the optimization framework, simple MLPs may struggle to accurately capture the intricacies of complex black-box functions, especially when the design itself contains rich semantic information. To better exploit domain-specific information, recent works have designed specialized architectures tailored to the data. For example, Lee et al. (2023) employs a graph neural network (GNN) designed for modeling molecular properties and guiding molecule generation, while Chen et al. (2025) uses a protein language model to extract residue embeddings that feed into a GNN for predicting antibody binding affinity, thereby steering antibody structures towards a more stable conformation.

It is worth noting that besides neural networks, other parametric models—such as linear regression, logistic regression, polynomial regression, and support vector machines (SVM)—are also useful (Hastie et al., 2009). However, in offline MBO, these alternatives are used less frequently, as the current surge in offline MBO has largely been driven by advances in deep NNs due to their superior generalization ability and scalability.

Kernel-Based Model In offline MBO, the offline dataset is often limited, making non-parametric *kernel-based models* particularly attractive. Their effective complexity increases with the number of data points, enabling them to capture intricate function behaviors while providing principled uncertainty estimates (Rasmussen & Williams, 2006). Although offline MBO does not allow further queries of the black-box function, uncertainty quantification remains crucial: it helps identify regions in the design space where the surrogate's predictions are less reliable. This information can be used to guide conservative optimization strategies that avoid over-optimistic predictions in poorly sampled areas.

Chen et al. (2022b) demonstrates that the neural tangent kernel—associated with infinite-width neural networks—can be more effective than standard kernels like the RBF kernel. Furthermore, Chen et al. (2023b) introduces a kernel parameterized by pre-trained biological language models for biological sequence design. In cases where pre-trained models are unavailable, deep kernel learning can be employed to learn the kernel directly from the data, with scalability achieved via inducing points (Wilson et al., 2016).

Besides kernel-based models, other non-parametric methods—such as k-nearest neighbors (kNN) and random forests—can also serve as surrogate models. However, these approaches are typically non-differentiable and do not naturally provide robust uncertainty estimates, which makes them less suitable for gradient-based optimization in offline MBO (Hastie et al., 2009).

4.3 Training Strategy

To enhance generalization and robustness, surrogate models often incorporate specialized *training strategies*. We categorize these into four groups: *auxiliary loss*, *data-driven adaptation*, *collaborative ensembling*, and *generative model integration*. Note that a single method can span multiple categories due to its inherent complexity. In such cases, the method may be discussed in more than one group, with emphasis placed on the aspect most relevant to that group.

Auxiliary Loss *Auxiliary losses* are incorporated into surrogate models to refine the training process by targeting specific model behaviors, particularly regularization. For instance, Trabucco et al. (2021) trains the surrogate to systematically underestimate the true objective on out-of-distribution inputs by identifying potential adversarial examples via gradient ascent and penalizing the surrogate's predictions at these points, thereby enforcing a conservative estimate. Similarly, Yu et al. (2021) smooths the training data with a Gaussian filter, finds the weight perturbation that maximizes the loss, and then adjusts the model parameters to maintain local smoothness with respect to both the inputs and the weights. In the same vein, Dao et al. (2024a;b) incorporate measures of model sharpness and sensitivity, respectively, as additional regularizers.

Further, Chen et al. (2022b; 2023b) propose a bidirectional learning framework that integrates knowledge from an offline dataset into a high-performing design using both forward and backward loss mappings. Here, a kernel-based model provides a closed-form solution while the backward mapping serves as a regularizer to mitigate the impact of out-of-distribution inputs. Strictly speaking, this framework does not regularize the surrogate itself but rather the behavior of the optimized designs.

In Qi et al. (2022a), offline MBO is reframed as a domain adaptation problem by treating the offline dataset as the source domain and the optimized designs as the target domain. A distance loss between these distributions is employed to train the surrogate, ensuring it produces mediocre predictions when the optimized designs deviate significantly from the training data. Furthermore, Hoang et al. (2024) shows that the quality of optimization correlates with how well the surrogate aligns with the latent gradient field underlying the offline data, and it proposes a loss function to enforce this gradient matching. Moreover, ranking losses have been shown to offer greater robustness compared to the traditional pointwise mean squared error loss: Chen et al. (2023a) employs a pairwise ranking loss, whereas Tan et al. (2025) utilizes a listwise ranking loss. Finally, auxiliary losses naturally extend to multi-objective settings. Multi-task learning techniques such as GradNorm (Chen et al., 2018) and PcGrad (Yu et al., 2020) can be used so that learning one property via a surrogate task aids in predicting another—a valuable capability in domains like biology where labeled data is often limited (Xu et al., 2022).

Data-Driven Adaptation *Data-driven adaptation* generally falls into three categories:

- **Sample Reweighting** This method assigns higher weights to samples deemed more relevant. For instance, Yuan et al. (2023) leverages bi-level optimization (Chen et al., 2022a) to learn weights for generated samples, thereby mitigating noise and enhancing surrogate. Similarly, AutoFocus calculates offline sample weights as the ratio of the probability under the search model to the initial probability, effectively refining the surrogate in the most relevant design regions (Fannjiang & Listgarten, 2020).
- **Synthetic Data Generation** Synthetic data is extensively used in offline MBO. For example, Trabucco et al. (2021) employs gradient ascent to generate adversarial designs, penalizing the surrogate on these points. Moreover, Chen et al. (2023a) and Yuan et al. (2023) apply pseudo-labeling for nearby points based on surrogate predictions, filtering out noisy samples to further improve the surrogate.
- **Domain Knowledge Injection** Incorporating domain-specific knowledge can enrich the surrogate model's understanding and enhance its extrapolation capabilities. For instance, Chen et al. (2023b) leverages a pre-trained biological language model—trained on millions of biological sequences—as a feature extractor, yielding superior performance compared to models without pre-training. Furthermore, Kuba et al. (2024b) introduces Functional Graphical Models that build a data-specific graph capturing functional independence properties, thereby imposing a structural bias that benefits black-box optimization and mitigates distribution shifts.

These data-driven techniques are applicable not only to surrogate modeling but also to generative modeling, as discussed in Section 5. In addition to *auxiliary losses* and *data-driven adaptations*, training strategies also benefit from insights drawn from *peer models* and *generative models*, as described next.

Collaborative Ensembling Ensemble learning techniques combine predictions from multiple models to achieve improved performance and generalization compared to individual base learners (Hansen & Salamon, 1990; Dietterich, 2000). In the context of offline MBO, recent studies have focused on developing ensemble-based surrogate models tailored to these settings. For example, Chen et al. (2023a) and Yuan et al. (2023) employ a mean ensemble of surrogates, wherein multiple models exchange valuable sample information during optimization to enhance learning—contrasting with traditional ensembles that generally interact only during aggregation. Additionally, Fu & Levine (2021) introduces an innovative approximation of the normalized maximum-likelihood (NML) distribution to construct an uncertainty-aware forward model. For each optimization point, the approach assigns multiple labels and trains separate models on each point-label pair, with the resulting ensemble estimating the conditional NML distribution to provide robust surrogate predictions that guide the design optimization process. Furthermore, Kolli (2023) tackles gradient conflicts

among ensemble members by employing multiple gradient descent steps and conflict-averse gradient descent, thereby striking a balance between conservatism and optimality.

Generative Model Integration Surrogate models often guide the sampling process of generative models, and in turn, several studies leverage insights from generative models to further enhance surrogate modeling. For instance, Fannjiang & Listgarten (2020) employs a variational autoencoder to model both the offline data distribution and the search model distribution, using the ratio of these probabilities as an importance weight to retrain the surrogate. Similarly, Qi et al. (2022a) trains a GAN discriminator to differentiate between the offline distribution and the desired design distribution, subsequently fine-tuning the surrogate to yield mediocre predictions when designs deviate significantly from the offline data. Moreover, Chen et al. (2024) derives a conditional distribution from a diffusion model, which is used to regularize the surrogate by minimizing the KL divergence between the surrogate's output and the derived distribution.

4.4 Sampling Strategy

Once the surrogate model is trained, gradients are computed to guide the sampling process. The typical procedure is outlined in Equation (3), with the general form given by:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \eta \cdot \mathrm{OPT}\Big(\nabla_{\boldsymbol{x}} f_{\boldsymbol{\phi}}(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{x}_t}\Big), \quad \text{for } t \in [0, T-1]\,. \tag{8}$$

This section discusses key aspects of the sampling process: the *learning rate* η, the *number of iterations* T, the optimizer *OPT*, and *test-time training*.

Learning Rate Selecting an optimal *learning rate* η poses a significant challenge in offline MBO due to the absence of a dedicated validation set. While Beckham et al. (2024) suggests the introduction of a validation set in offline MBO, this method may require sacrificing some high-performance data. On the other hand, Trabucco et al. (2022) recommends a learning rate of $2\sqrt{d}$ for discrete tasks and $0.05\sqrt{d}$ for continuous tasks, where d represents the dimension of the design space. The higher learning rate in discrete spaces accommodates their unique properties, and scaling by $\sqrt{d}$ compensates for the increase in the overall gradient norm as dimensionality increases. Additionally, Chen et al. (2023b) proposes training an auxiliary model to provide weak supervision signals for optimizing the learning rate, thus enhancing sampling effectiveness. Similarly, Chemingui et al. (2024) formulates offline MBO as an offline reinforcement learning problem, where a learned policy takes the current design as input and outputs the optimal learning rate; however, this approach may still be vulnerable to reward hacking due to out-of-distribution issues in the surrogate model.

Number of Iterations The number of optimization steps, denoted as T, is another vital hyperparameter. Determining the appropriate T is challenging due to the absence of a ground-truth function, which raises concerns about overfitting. This hyperparameter also correlates with the learning rate: a higher learning rate might necessitate a smaller T to avoid deviating from the distribution. The strategy suggested by Trabucco et al. (2021) involves using 50 steps to generate adversarial samples and regularizing the surrogate model based on these samples. Subsequently, a similar 50-step approach is employed during the sampling phase. Meanwhile, Yu et al. (2021); Fu & Levine (2021) report that their methods remain robust even as T increases. Furthermore, Damani et al. (2023) proposes training a binary classifier to distinguish offline data from design data, observing that the degree of distribution shift depends on T. The classifier logits serve as a proxy for distribution shift, allowing users to constrain T to regions where the surrogate predictions remain reliable.

OPT The term *OPT* denotes the optimizer, which can be an algorithm such as SGD, Adam, etc. (Ruder, 2016). These optimizers are typically applied to offline SOO. In offline MOO, however, multiple gradients must be managed simultaneously. A naive approach of computing a weighted sum of these gradients often results in conflicts that hamper effective optimization. Recent methods address these conflicts: the multiple gradient descent algorithm (Désidéri, 2012) finds a common descent direction by assigning nonnegative weights that minimize the norm of the combined gradient, while PCGrad (Yu et al., 2020) resolves conflicts by projecting one gradient onto the orthogonal space of another when their inner product is negative, thereby enhancing robustness in multi-objective settings.

Test-Time Training A common sampling strategy involves adapting the surrogate model at the current optimization point. While this concept overlaps with the training strategies discussed in Section 4.3, the focus here is on fine-tuning the surrogate locally. Given the impracticality of optimizing the surrogate globally, it is more feasible to refine its performance near the current point. For example, Fu & Levine (2021) estimates the conditional normalized maximum likelihood by incorporating the current point into the surrogate modeling process, and (Yu et al., 2021) adjusts the surrogate to increase local smoothness. Moreover, Chen et al. (2023a); Yuan et al. (2023) generate pseudo pairwise and pointwise labels in the neighborhood to further refine the surrogate's local behavior.

5 Generative Modeling

In addition to *surrogate modeling* described in Section 4, *generative modeling* plays a pivotal role in offline MBO. The high-dimensionality of design spaces renders exploration challenging, and generative models offer an effective means to navigate these spaces. As detailed in Eq. (7), offline optimization can be framed as a conditional generation problem, where the objective is to model the distribution $p(\boldsymbol{x} \mid \boldsymbol{y}_c)$ with $\boldsymbol{y}_c$ representing the target conditions. By Bayes' rule, this distribution is proportional to the product of the prior $p(\boldsymbol{x})$ and the likelihood $p(\boldsymbol{y}_c \mid \boldsymbol{x})$. Broadly, two categories of conditional generation emerge in this context:

- **Inverse** This approach directly trains a conditional generative model to learn the mapping from target conditions $\boldsymbol{y}_c$ to designs $\boldsymbol{x}$, thereby capturing $p(\boldsymbol{x} \mid \boldsymbol{y}_c)$ and enabling conditional sampling.
- **Forward** This approach leverages a surrogate model for $p(\boldsymbol{y}_c \mid \boldsymbol{x})$ to steer an unconditional generative model $p(\boldsymbol{x})$ toward desirable designs. A notable special case is the use of direct gradient ascent, which bypasses the need to explicitly model the generative component $p(\boldsymbol{x})$.

In the remainder of this section, we first outline the *general principles* of these generative models—including *variational autoencoder* (VAE) (Kingma & Welling, 2014), *generative adversarial network* (GAN) (Goodfellow et al., 2014), *autoregressive model* (Vaswani et al., 2017), *diffusion model* (Ho et al., 2020), *flow matching* (Lipman et al., 2023) and *energy-based model* (EBM) (LeCun et al., 2006)—followed by a discussion on how they achieve *conditional generation* within offline MBO. Finally, we introduce *Generative Flow Network* (GFlowNet) (Bengio et al., 2023), a versatile control strategy applicable to a wide range of generative models.

5.1 Variational Autoencoder (VAE)

General Principle *Variational Autoencoders* (VAEs) integrate ideas from variational inference and autoencoders to learn a probabilistic latent representation $\boldsymbol{z}$ for the data $\boldsymbol{x}$ (Kingma & Welling, 2014). The model expresses the data likelihood as

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{z})\, p(\boldsymbol{z})\, d\boldsymbol{z},$$

and introduces an approximate posterior $q_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{x})$ to facilitate efficient inference.

In particular, the VAE adopts an encoder–decoder architecture: the encoder approximates the posterior $q_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{x})$, while the decoder models the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$. The model is trained by maximizing the evidence lower bound (ELBO):

$$\mathcal{L}_{\text{VAE}}(\boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] - \text{KL}\Big(q_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{x}) \,\|\, p(\boldsymbol{z})\Big).$$

Both the likelihood term $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ and the prior $p(\boldsymbol{z})$ (typically chosen as $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$) are usually modeled as Gaussian distributions. The KL divergence acts as a regularizer that promotes stable training, though balancing reconstruction accuracy with latent regularization—and addressing issues like *posterior collapse*—often requires careful tuning (Higgins et al., 2017). Sampling from a VAE is performed by drawing a latent code $\boldsymbol{z}$ from the prior $p(\boldsymbol{z})$ and then generating $\boldsymbol{x}$ using the decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$.

Conditional Generation In the context of VAEs, conditional generation can be achieved through two categories of methods: inverse and forward methods. The inverse method directly trains a label-conditioned VAE to learn the mapping from target conditions to designs. For example, Brookes et al. (2019) adaptively trains the VAE on high-performing designs (such as protein sequences), enabling the direct sampling of promising candidates without the need for an external surrogate model.

In contrast, forward methods are more commonly employed, leveraging the continuous latent space of the VAE. Here, an unconditional VAE is first trained to embed designs into a continuous latent space, after which a surrogate model supplies gradient information to steer the latent codes toward regions corresponding to improved designs. This approach is particularly beneficial for discrete design optimization, as demonstrated by works on general discrete designs (Yu et al., 2021) and discrete molecules (Gómez-Bombarelli et al., 2018), which map these designs into a latent space amenable to gradient-based optimization. Together, these methods illustrate how VAEs can effectively navigate high-dimensional design spaces, either by directly conditioning on target attributes via the inverse method or by leveraging gradient-driven manipulations within the latent space via the forward method.

We also briefly compare VAEs and normalizing flows (Kobyzev et al., 2020), as both models map designs to latent spaces and back. While VAEs rely on a learned encoder–decoder architecture, normalizing flows use carefully designed invertible operators to establish a one-to-one correspondence between the latent and input spaces. Lee et al. (2025) observes that this invertibility effectively mitigates the reconstruction gap often seen in VAEs—which can cause property discrepancies between original and reconstructed designs. Consequently, they propose a normalizing flow model for MBO, including the SeqFlow variant for sequence designs, to address these issues directly. Although the application of normalizing flows in offline model-based optimization (MBO) remains relatively limited, they represent a promising direction for future research.

5.2 Generative Adversarial Network (GAN)

General Principle *Generative Adversarial Networks* (GANs) introduce an adversarial training framework in which a generator network $G_{\boldsymbol{\theta}}(\boldsymbol{z})$ and a discriminator network $D_{\boldsymbol{\psi}}(\boldsymbol{x})$ compete against each other (Goodfellow et al., 2014). The generator maps noise $\boldsymbol{z}$ (sampled from a simple distribution, such as $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$) into the data space, while the discriminator attempts to distinguish real data from generated samples. The standard training loss is formulated as a minimax game:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\psi}} \; \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} \left[\log D_{\boldsymbol{\psi}}(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \left[\log\big(1 - D_{\boldsymbol{\psi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))\big)\right].$$

Originally proposed for image generation, GANs are renowned for their ability to produce sharper and more realistic outputs compared to VAEs, although they are also known for training instability. Various strategies, such as Wasserstein GAN (Arjovsky et al., 2017), have been introduced to improve convergence. For sampling, GANs draw noise and use the generator to synthesize realistic samples. Generally speaking, unlike VAEs, standard GANs do not provide an explicit latent code for a given sample, though extensions like BiGANs (Donahue et al., 2017) offer a means to recover latent representations.

Conditional Generation GAN-based conditional generation is typically achieved via inverse methods, owing to the absence of an inherent latent code in standard GANs. In such frameworks, both the generator and discriminator are conditioned on target labels to steer the sampling process. For instance, Kumar & Levine (2020) employs a conditional GAN where the discriminator, parameterized as $D_{\boldsymbol{\psi}}(\mathbf{x} \mid y)$, is trained to output 1 for valid $(\mathbf{x}, y)$ pairs (i.e., when $\mathbf{x}$ comes from the data and $y = f(\mathbf{x})$) and 0 for generated pairs $\big(G_{\boldsymbol{\theta}}(\mathbf{z}, y),\, y\big)$. Here, the generator acts as the inverse mapping $G_{\boldsymbol{\theta}}(\mathbf{z}, y)$, taking both the latent noise $\mathbf{z}$ and the condition label y as inputs. This setup is optimized using the following objective:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\psi}} \;\; \mathcal{L}_p(\mathcal{D}) \;=\; \mathbb{E}_{y\sim p(y)} \Bigg[\mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x}|y)} \Big[\log D_{\boldsymbol{\psi}}(\mathbf{x} \;\mid\; y)\Big] + \mathbb{E}_{\mathbf{z}\sim p_0(\mathbf{z})} \Big[\log\Big(1 - D_{\boldsymbol{\psi}}\big(G_{\boldsymbol{\theta}}(\mathbf{z}, y) \;\mid\; y\big)\Big)\Big] \Bigg]. \quad (9)$$

This formulation corresponds to matching the true conditional distribution $p_{\mathcal{D}}(\mathbf{x} \mid y)$ with the model distribution $p_{G_{\boldsymbol{\theta}}}(\mathbf{x} \mid y)$ (obtained by marginalizing over $\mathbf{z}$). During guided sampling, for a given target y, the

latent variable $\mathbf{z}$ is first sampled and then optimized such that the output of the forward model, $f_{\boldsymbol{\phi}}\big(G_{\boldsymbol{\theta}}(\mathbf{z}, y)\big)$, closely approximates y. This process quantifies the agreement between the learned inverse map and an independently trained forward model $f_{\boldsymbol{\phi}}$, ensuring that the generated sample $G_{\boldsymbol{\theta}}(\mathbf{z}, y)$ not only satisfies the desired condition but also lies on the valid data manifold. Importance reweighting is also employed to construct a $p(y)$ that assigns high probability to high y values.

Besides these inverse methods, the discriminator of GANs is often utilized to detect whether a design is out-of-distribution, thereby regulating the optimization process of surrogate models (Qi et al., 2022b; Yao et al., 2024). Although both surrogate models and generative models have been explored in this context, these methods are technically not GAN-based forward methods, as they do not optimize the latent code within the GAN framework but rather use the discriminator to regularize the behavior of surrogate models.

5.3 Autoregressive Model

General Principle *Autoregressive models* are widely adopted for generative tasks, particularly in language modeling. Notable examples include LSTMs (Hochreiter, 1997) and Transformer-based models (Vaswani et al., 2017; Brown et al., 2020), which factorize the joint distribution in an autoregressive manner. Specifically, the joint probability of a sequence is expressed as:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \prod_{i=1}^{N} p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i \mid \boldsymbol{x}_{<i}\right),$$

and the model is typically trained by maximizing the log-likelihood, often using the cross-entropy loss for discrete tokens. During generation, elements are sampled sequentially, with each new token conditioned on the previously generated tokens.

Conditional Generation Latent representations for the design exist within autoregressive models, enabling forward methods that manipulate these latents via gradient optimization. For example, Dathathri et al. (2020) proposes using gradients from a surrogate model to adjust the language model's hidden activations, thereby guiding the generation process. However, directly manipulating these latent representations is not widely adopted, likely because design properties depend on the sequence as a whole; modifying a token's latent representation without accounting for subsequent tokens can be less robust.

In autoregressive models, inverse methods are commonly applied and can be categorized into two types. The first type models *a single sequence design*, a strategy often employed in biological sequence design. For instance, Angermüller et al. (2020) utilizes an autoregressive model for biological sequences, using a surrogate as a reward and applying reinforcement learning to generate high-performing sequences. Similarly, Kim et al. (2023) employs an LSTM-based autoregressive model to generate biological sequences, then re-trains the generator using synthetic data labeled by the surrogate, assigning higher sample weights to high-performing synthetic sequences during training.

The second type models *a sequence of designs and labels*, aiming to capture the relationship between designs. In this line, Nguyen et al. (2023) pre-trains an autoregressive transformer on related and synthetic datasets and performs in-context learning by providing the offline dataset as context. A high-score label y_c is then used as a query to guide design generation. Additionally, Mashkaria et al. (2023) constructs a trajectory dataset by sorting samples based on score and trains an autoregressive model on this trajectory. During sampling, the model generates candidate points by rolling out a trajectory that implicitly serves as the condition y_c, thereby guiding the generation process in an inverse manner.

5.4 Diffusion Model

General Principle *Diffusion models*, a subset of latent variable models, gradually perturb data by injecting Gaussian noise during the forward process. The reverse process iteratively denoises the data using a learned score estimator (Ho et al., 2020). These models can be represented in a continuous framework via a stochastic differential equation (SDE), as discussed in Song et al. (2021). The forward SDE is expressed as:

$$d\boldsymbol{x} = \boldsymbol{h}(\boldsymbol{x}, t)\, dt + g(t)\, d\boldsymbol{w}, \tag{10}$$

where $\boldsymbol{h}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ denotes the drift coefficient, $g(t) : \mathbb{R} \to \mathbb{R}$ is the diffusion coefficient, and $\boldsymbol{w}$ represents a standard Wiener process. This formulation progressively converts the data distribution into a noise distribution. The reverse process is characterized by:

$$d\boldsymbol{x} = \left[\boldsymbol{h}(\boldsymbol{x}, t) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right] dt + g(t)\, d\bar{\boldsymbol{w}}, \tag{11}$$

with $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$ indicating the score of the marginal distribution at time t, and $\bar{\boldsymbol{w}}$ symbolizing the reverse-time Wiener process. The score function is approximated using a time-dependent neural network $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$, which facilitates the transformation of noise back into samples. They have demonstrated exceptional performance in synthesizing high-fidelity images and a wide range of complex data types.

Conditional Generation Conditional generation in diffusion models has been extensively studied and can also be broadly classified into two categories: inverse methods and forward methods.

In inverse methods, the conditional diffusion model is trained using either *vanilla guidance*, where the label or condition is directly provided as an input (Chang et al., 2023), or *classifier-free guidance*, which derives guidance by contrasting the outputs of a conditional model with those of an unconditional model (Ho & Salimans, 2022). For example, Zhang et al. (2024) proposes a vanilla guidance approach that learns a weight function to assign higher weights to high-performing designs. This method focuses the training of the diffusion model $\boldsymbol{s}_\theta(\boldsymbol{x}_t, y_c)$ on high-performing designs, using them as implicit high-performing conditions y_c during sampling. However, vanilla guidance lacks an adjustable parameter to control sampling strength, which motivates the use of classifier-free guidance where the strength can be tuned via a parameter ω. The corresponding score function is defined as follows:

$$\tilde{\boldsymbol{s}}_\theta(\boldsymbol{x}_t, y_c, \omega) = (1 + \omega)\boldsymbol{s}_\theta(\boldsymbol{x}_t, y_c) - \omega \boldsymbol{s}_\theta(\boldsymbol{x}_t). \tag{12}$$

Krishnamoorthy et al. (2023) successfully apply classifier-free guidance to offline MBO by inputting the maximum value y_c from the offline dataset to produce high-performing designs. Building on this, Chen et al. (2024) explore using a surrogate model to guide the parameter ω. Similarly, Yun et al. (2024) extend the method by incorporating not only the target property y_c but also the entire trajectory into the conditional model to steer generation. In another work, Dao et al. (2025) generate synthetic data and train diffusion models to map low-performance samples to high-performing designs; during sampling, the offline samples serve as initial samples and the diffusion model progressively guides them toward higher-performance designs.

In contrast, forward methods such as *classifier guidance* (Dhariwal & Nichol, 2021) employ a surrogate model to steer the sampling process. The score function for classifier guidance is given by:

$$\tilde{\boldsymbol{s}}_\theta(\boldsymbol{x}_t, y_c, \omega) = \boldsymbol{s}_\theta(\boldsymbol{x}_t) + \omega \nabla_{\boldsymbol{x}_t} \log p_{\boldsymbol{\phi}}(y_c | \boldsymbol{x}_t). \tag{13}$$

In this context, Lee et al. (2023) investigate guided molecule generation toward high-performing regions with respect to target properties such as protein-ligand interactions, drug-likeness, and synthesizability. In Yuan et al. (2024b), gradient ascent is first employed to optimize the design. To address potential out-of-distribution issues, the method subsequently recovers the corresponding latent representation by injecting diffusion noise and then applying a denoising procedure, yielding a sample that conforms to the diffusion prior. This approach can be interpreted as a variant of classifier guidance due to its use of classifier gradient. Compared to classifier-free guidance, classifier guidance is less frequently adopted, likely due to the additional training cost of an extra surrogate model and the potential risk of reward hacking associated with the classifier.

5.5 Flow Matching

General Principle *Flow matching* learns a vector field $v(\boldsymbol{x}, t)$ that defines a deterministic flow by solving an ordinary differential equation (Lipman et al., 2023; Le et al., 2023). Notably, $v(\boldsymbol{x}, t)$ can be used to derive the score $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$ and vice versa (see Lemma 1 in Zheng et al. (2023)). This demonstrates that diffusion models and flow matching follow the same probability path under certain constraints.

Flow matching defines a time-dependent conditional probability trajectory $p_t(\boldsymbol{x} \mid \boldsymbol{x}_1)$ for $t \in [0, 1]$, starting from $p_0(\boldsymbol{x} \mid \boldsymbol{x}_1) = q(\boldsymbol{x})$ and evolving toward an approximate Dirac delta, $p_1(\boldsymbol{x} \mid \boldsymbol{x}_1) \approx \delta(\boldsymbol{x} - \boldsymbol{x}_1)$, where $\boldsymbol{x}_1$ is

drawn from p_{data}. This evolution is conditioned on a specific data sample $\boldsymbol{x}_1$ and driven by a time-dependent vector field $u_t(\boldsymbol{x} \mid \boldsymbol{x}_1)$. A neural network with parameters $\boldsymbol{\theta}$ is used to learn the marginal vector field,

$$\hat{v}(\boldsymbol{x}, t; \theta) \approx v(\boldsymbol{x}, t) = \mathbb{E}_{\boldsymbol{x}_1 \sim p_t(\boldsymbol{x}_1 \mid \boldsymbol{x})}\Big[u_t(\boldsymbol{x} \mid \boldsymbol{x}_1)\Big], \tag{14}$$

which acts as a neural ODE that transforms the noise distribution $q(\boldsymbol{x})$ into the data distribution $p_{\text{data}}(\boldsymbol{x})$.

Following Pooladian et al. (2023), the process begins by sampling a noise vector $\boldsymbol{x}_0$ from $q(\boldsymbol{x}_0)$ and linearly interpolating it with a data point $\boldsymbol{x}_1$ via $\boldsymbol{x} \mid \boldsymbol{x}_1, t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$ (with $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$). The corresponding conditional vector field is defined as $u_t(\boldsymbol{x} \mid \boldsymbol{x}_1) = \frac{\boldsymbol{x}_1 - \boldsymbol{x}}{1-t}$, which, given the interpolation, simplifies to $u_t(\boldsymbol{x} \mid \boldsymbol{x}_1) = \boldsymbol{x}_1 - \boldsymbol{x}_0$. The training objective minimizes the loss

$$\mathcal{L}(\theta) = \mathbb{E}_{t,\, p_{\text{data}}(\boldsymbol{x}_1),\, q(\boldsymbol{x}_0)} \left\| \hat{v}(\boldsymbol{x}, t; \theta) - (\boldsymbol{x}_1 - \boldsymbol{x}_0) \right\|^2. \tag{15}$$

Once trained, the vector field $\hat{v}(\boldsymbol{x}, t; \theta)$ is used to generate new samples by integrating the neural ODE.

Conditional Generation Because flow matching closely resembles diffusion models, analogous conditional generation techniques can be applied. In particular, both inverse method classifier-free guidance (Zheng et al., 2023) and forward method classifier guidance (Dao et al., 2023) are readily adaptable within the flow matching framework. Regarding the inverse method, Stärk et al. (2024) introduce Dirichlet flow matching on the simplex and extend classifier-free guidance to more effectively steer the sequence generation process. In the forward method, Yuan et al. (2024a) investigate the use of multiple surrogate models to guide flow sampling toward the Pareto-front in multi-objective optimization settings. Moreover, Chen et al. (2025) propose training an affinity predictor to steer protein conformation sampling toward stable configurations within the AlphaFlow framework (Jing et al., 2024).

Given flow matching's emerging success and its demonstrated advantages in performance and efficiency over diffusion models, we anticipate a surge in research exploring its applications to offline MBO.

5.6 Energy-Based Model

General Principle *Energy-based Models* (EBMs) define an unnormalized probability distribution over data via an energy function $E_\theta(\boldsymbol{x})$ (LeCun et al., 2006):

$$p_\theta(\boldsymbol{x}) = \frac{\exp\Big(-E_\theta(\boldsymbol{x})\Big)}{Z(\theta)},$$

where $Z(\theta)$ denotes the partition function. Training EBMs typically aims to assign lower energy to observed data while raising the energy of samples drawn from the model. A common objective is contrastive divergence:

$$\mathcal{L}_{\text{EBM}} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}\left[E_\theta(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x} \sim p_\theta(\boldsymbol{x})}\left[E_\theta(\boldsymbol{x})\right].$$

This formulation encourages the model to distinguish between real data and generated (negative) samples.

Sampling from an EBM is challenging due to the intractability of the partition function, and it generally relies on Markov Chain Monte Carlo (MCMC) techniques (Neal, 1993; Hinton, 2002; Tieleman, 2008). A widely used approach is Langevin dynamics (Welling & Teh, 2011), with the iterative update:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{\epsilon}{2}\nabla_{\boldsymbol{x}} E_\theta(\boldsymbol{x}_k) + \sqrt{\epsilon}\,\boldsymbol{\eta}_k, \quad \boldsymbol{\eta}_k \sim \mathcal{N}(0, I),$$

where ϵ is the step size. This stochastic process, which introduces Gaussian noise $\boldsymbol{\eta}_k$, is designed to asymptotically sample from $p_\theta(\boldsymbol{x})$. Often, hundreds to thousands of iterations are needed to ensure adequate mixing, and slow mixing remains a notable challenge, especially in high-dimensional spaces.

EBMs provide a flexible framework for modeling complex data distributions by directly parameterizing an energy function. While they share a fundamental relation with score-based models (Song et al., 2021)—since $\nabla_{\boldsymbol{x}} \log p_\theta(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} E_\theta(\boldsymbol{x})$—the two approaches diverge in practice: EBMs require explicit energy modeling

and rely on MCMC-based sampling (e.g., via Langevin dynamics), whereas score-based models aim to directly learn the score function without necessarily specifying an energy function. We note that GFlowNets (Bengio et al., 2023) can amortize the MCMC process, making them applicable to EBMs (Zhang et al., 2022). Although these methods enable faster mode mixing to estimate the partition function to compare with MCMC methods, thus improving the practicality of EBMs, none of these techniques have been applied to offline optimization. We believe that exploring this direction could offer promising avenues for future research.

Conditional Generation Conditional generation in the context of EBMs can similarly be classified into inverse and forward methods. For instance, Frey et al. (2024) maps protein sequences into a latent space during an initial *jump* step and trains an EBM on this latent representation—assigning lower energy to observed data while penalizing generated samples with higher energy. In the subsequent *walk* step, Langevin MCMC is employed to sample new latent codes, with a binary projection matrix ensuring that specified regions of the sequence remain unchanged.

Similarly, Yu et al. (2024) introduces a method that jointly embeds design and properties into a compact yet expressive energy-based latent space. In this approach, the highest offline dataset score, y_c, is used to sample a latent code $\boldsymbol{z}$, which is then decoded to yield the design $\boldsymbol{x}$. We categorize this method as forward, since the sampling of $\boldsymbol{z}$ is governed by

$$p(\boldsymbol{z} \mid y_c) \propto p_{\boldsymbol{\theta}}(\boldsymbol{z})\, p_{\boldsymbol{\phi}}(y_c \mid \boldsymbol{z}),$$

with an explicitly modeled surrogate whose gradient is leveraged via SVGD (Liu & Wang, 2016).

It is important to note that the distinction between forward and inverse methods in EBMs is often subtle, as both the energy function and a surrogate model essentially map a design to a scalar value. In Beckham & Pal (2023), the authors reinterpret the original forward method, COMs, as an EBM trained via contrastive divergence. COMs optimizes two losses—a mean-squared error loss for surrogate modeling and a conservative objective for the EBM—using a shared network architecture, where the resulting energy function steers the sampling process towards high-performing designs, a characteristic typically associated with inverse methods. Moreover, Beckham & Pal (2023) further proposes a decoupled version of COMs, in which separate networks are employed for the surrogate and the EBM, reinforcing its classification as a forward method.

5.7 Control by Generative Flow Network (GFlowNet)

Unlike earlier sections that focus on general generative models and their conditional generation approaches, this section introduces a *control* method for these models: *Generative Flow Networks* (GFlowNets), which have recently gained popularity. GFlowNets (Bengio et al., 2021; 2023) model the generative process as a sequential decision-making problem, where a complete candidate solution $\boldsymbol{x}$ is constructed through a sequence of transitions. Their goal is to perform amortized inference by sampling from a distribution proportional to a reward function, i.e., $p(\boldsymbol{x}) \propto R(\boldsymbol{x})$. This approach produces samples that are both high-reward and diverse, which is particularly beneficial when the reward function is an imperfect surrogate learned from an offline dataset. In contrast, standard RL methods that simply maximize $R(\boldsymbol{x})$ can be unsafe under epistemic uncertainty, making GFlowNets especially attractive in offline optimization scenarios.

It is important to note that GFlowNets represent a learning method rather than a standalone generative model. Their sequential decision-making process is typically implemented by generative models such as autoregressive models, diffusion models, or any model that provides sequential inference structures.

GFlowNets Contraints In GFlowNets, a sample $\boldsymbol{x}$ is generated by traversing a directed acyclic graph from an initial state $\boldsymbol{s}_0$ to a terminal state $\boldsymbol{s}_T = \boldsymbol{x}$ through a sequence of transitions:

$$\boldsymbol{s}_0 \;\rightarrow\; \boldsymbol{s}_1 \;\rightarrow\; \cdots \;\rightarrow\; \boldsymbol{s}_T = \boldsymbol{x}.$$

The central training criterion enforces flow consistency: for each intermediate state $\boldsymbol{s}'$, the total incoming flow must equal the total outgoing flow:

$$\sum_{\boldsymbol{s} \in \mathrm{Parents}(\boldsymbol{s}')} F(\boldsymbol{s};\theta)\, P_F\big(\boldsymbol{s}' \mid \boldsymbol{s};\theta\big) \;=\; F(\boldsymbol{s}';\theta).$$

At the terminal state, the flow is fixed to the reward,

$$F(\boldsymbol{s}_T;\theta) \;=\; R(\boldsymbol{x}),$$

which serves as the surrogate objective in this work.

A common way to implement flow consistency is via the *detailed balance* (DB) condition, introducing a backward policy $P_B(\boldsymbol{s} \mid \boldsymbol{s}';\theta)$ to match flows:

$$F(\boldsymbol{s};\theta)\, P_F\big(\boldsymbol{s}' \mid \boldsymbol{s};\theta\big) \;=\; F(\boldsymbol{s}';\theta)\, P_B\big(\boldsymbol{s} \mid \boldsymbol{s}';\theta\big), \quad \boldsymbol{s}' \in \mathrm{Children}(\boldsymbol{s}).$$

This one-step condition is similar to *temporal-difference* (TD) learning in RL. It can be extended to longer trajectories: a partial subsequence $\boldsymbol{s}_m \to \cdots \to \boldsymbol{s}_l$ (sub-trajectory balance, Sub-TB (Madan et al., 2023)) or a complete sequence $\boldsymbol{s}_0 \to \cdots \to \boldsymbol{s}_n = \boldsymbol{x}$ (trajectory balance, TB (Malkin et al., 2022)). These extensions are analogous to TD-λ and Monte Carlo learning methods. Sub-TB is formulated as:

$$F(\boldsymbol{s}_m;\theta) \prod_{i=m+1}^{l} P_F\big(\boldsymbol{s}_i \mid \boldsymbol{s}_{i-1};\theta\big) \;=\; F(\boldsymbol{s}_l;\theta) \prod_{i=m+1}^{l} P_B\big(\boldsymbol{s}_{i-1} \mid \boldsymbol{s}_i;\theta\big),$$

while the TB condition is:

$$Z_\theta \prod_{i=1}^{n} P_F\big(\boldsymbol{s}_i \mid \boldsymbol{s}_{i-1};\theta\big) \;=\; R(\boldsymbol{x}) \prod_{i=1}^{n} P_B\big(\boldsymbol{s}_{i-1} \mid \boldsymbol{s}_i;\theta\big),$$

where $Z_\theta = F(\boldsymbol{s}_0)$ is the partition function and $R(\boldsymbol{x}) = F(\boldsymbol{s}_n)$ is the terminal reward.

Recent work refines TB for posterior inference by introducing *relative trajectory balance* (RTB) (Venkatraman et al., 2024), where

$$p_{\mathrm{post}}(\boldsymbol{x}) \;\propto\; p_{\mathrm{prior}}(\boldsymbol{x})\, R(\boldsymbol{x}).$$

In RTB, the backward policy P_B is replaced by a prior distribution P_{prior}, yielding:

$$Z_\theta \prod_{i=1}^{n} P_{\mathrm{post}}\big(\boldsymbol{s}_i \mid \boldsymbol{s}_{i-1};\theta\big) \;=\; R(\boldsymbol{x}) \prod_{i=1}^{n} P_{\mathrm{prior}}\big(\boldsymbol{s}_i \mid \boldsymbol{s}_{i-1}\big).$$

This can be applied to active learning and offline optimization, allowing for a conservative strategy by restricting samples to the prior distribution learned from offline data, thus reducing risk under uncertainty.

Controlling Generative Models via GFlowNets During sampling, a GFlowNet sequentially selects transitions based on its learned policy until reaching a terminal state $\boldsymbol{x}$. This sequential framework is particularly effective for exploring high-dimensional spaces and generating diverse candidate solutions. In offline MBO, guided sampling via GFlowNets is especially natural.

For example, Jain et al. (2022) generates desirable biological sequences from scratch by defining a target reward function as a UCB score from a surrogate model. To improve robustness against the imperfections of the surrogate model at early stages, Kim et al. (2024a) conservatively search promising regions by introducing a parameter δ, which is adjusted based on the prediction model's uncertainty. Additionally, Ghari et al. (2023) utilize GFlowNets to modify existing sequences to enhance target properties, while Jain et al. (2023) employ conditional GFlowNets to generate diverse Pareto-optimal solutions for multi-objective optimization problems. Note that they use autoregressive models to produce these biological sequences and employ GFlowNets to control them effectively.

One of the main challenges in offline optimization is the exploration-exploitation trade-off. Unlike RL, GFlowNets can effectively balance exploration and exploitation using a temperature parameter β, i.e., $p(\boldsymbol{x}|\beta) \propto R(\boldsymbol{x})^\beta$. Kim et al. (2024b) propose an effective way to learn such a policy by introducing a logit scaling network and verify that it achieves high extrapolation ability in offline optimization tasks. They use directional token generative models, which can generate sequences by prepending or appending tokens, and employ GFlowNets to control them.

We can also formulate the offline optimization problem as posterior inference, where we have a pre-trained policy $p(\boldsymbol{x})$ and a reward function $R(\boldsymbol{x})$ trained on a given dataset. In this setting, we can use the RTB loss to obtain an unbiased sampler that amortizes the posterior distribution. For example, Yun et al. (2025) propose a posterior inference method with GFlowNets for diffusion models in black-box online optimization. This work may provide a key insight for applying posterior inference with GFlowNets to offline optimization.

6 Discussion and Future Direction

In this paper, we present a comprehensive review of offline MBO. Despite significant efforts to develop robust surrogate models and generative models, many practical challenges remain. Offline MBO is inherently difficult due to the limited availability of offline datasets and the epistemic uncertainty that plagues surrogate models. In this section, we outline promising future research directions aimed at addressing these challenges, including the development of more rigorous benchmarks, improved uncertainty estimation methods, innovative graphical modeling approaches for surrogate modeling, advanced generative modeling techniques, and high-impact applications in LLM alignment and AI safety.

Robust and Realistic Benchmarking Current benchmarks in offline MBO face two major challenges. First, some benchmarks—such as TFB8 and TFB10 (Barrera et al., 2016a)—offer overly constrained search spaces where even simple gradient ascent methods can achieve impressive results, making it difficult to distinguish the performance of more sophisticated algorithms. Second, benchmarks like superconductor (Hamidieh, 2018) often rely on learned oracles for evaluation, which can be vulnerable to manipulation and may not accurately reflect true performance. Moving forward, it is essential to develop benchmarks that not only present more challenging search spaces but also incorporate rigorous, reliable evaluation protocols that are resistant to exploitation.

Uncertainty Estimation of Surrogate Model In offline optimization, capturing the significant epistemic uncertainty in surrogate models is paramount, especially because identifying which inputs drive this uncertainty remains a core challenge. While uncertainty estimation is similarly important in online optimization, in offline scenarios it is even more critical for preventing reward hacking, as one cannot correct for flawed model predictions through active data collection. Various techniques—such as adversarial regularization (Trabucco et al., 2021), smoothness priors (Yu et al., 2021), and kernel-based methods (Chen et al., 2022b)—have been proposed to mitigate uncertainty and ensure safer optimization; however, there has been comparatively little focus on leveraging Bayesian methods for directly quantifying epistemic uncertainty. Moreover, existing benchmarks often emphasize overall optimization performance without clarifying whether observed gains stem from superior surrogate modeling, improved optimization strategies, or mere chance. This lack of distinction underscores the need for independent and rigorous evaluations of the uncertainty estimation capabilities of the surrogate model in newly developed algorithms. Although recent efforts—such as Jain et al. (2022) using Monte Carlo (MC) dropout (Gal & Ghahramani, 2016) and deep ensembles (Lakshminarayanan et al., 2017), and Kim et al. (2024a) incorporating uncertainty measures for conservative search—have made inroads into this area, they still fall short of providing a robust solution. Consequently, advancing offline MBO demands a fundamentally stronger approach to uncertainty quantification that transcends basic MC dropout or ensembling techniques.

Future work may build upon this line of research by exploring recent methodologies such as:

- **DEUP: Direct Epistemic Uncertainty Prediction** (Lahlou et al., 2023): This approach directly estimates the excess risk associated with model misspecification by learning a secondary predictor for generalization error, offering a more principled uncertainty measure.
- **Efficient Variational Inference Methods over Neural Network Parameters**: Efficient variational inference methods offer promising solutions for tackling the intractable posterior inference in neural networks. For example, the Variational Bayesian Last Layer (VBLL) approach (Harrison et al., 2024) performs inference solely on the final layer, reducing the computational complexity to a quadratic level. Similarly, GFlowOut, introduced by Liu et al. (2023), leverages GFlowNets—a form of hierarchical

variational inference (Malkin et al., 2023)—to approximate the posterior distribution over dropout masks, effectively addressing challenges related to multi-modality and sample dependency. Together, these methods represent compelling candidates for enhancing uncertainty estimation in surrogate models while maintaining computational efficiency.

- **Deep Kernels**: Deep kernels, which leverage deep architectures to learn expressive kernel functions, offer a scalable alternative to Gaussian processes (Wilson et al., 2016; Liu et al., 2020). They have been applied to biological sequence design scenarios by employing a denoising autoencoder to learn a discriminative deep kernel GP for Bayesian optimization of biological sequences (Stanton et al., 2022). This approach was further refined by Gruver et al. (2023), who used a discrete diffusion model and simple last-layer ensemble techniques to evaluate uncertainty. These successes in online optimization could be directly transferred to offline MBO, where uncertainty estimation helps prevent reward hacking. Such a strategy could effectively combine existing surrogate modeling and generative modeling methods in offline MBO fields.

Graphical Surrogate Model Offline surrogate models often suffer from limited data coverage, making them vulnerable to overestimation and reward hacking in regions outside the training distribution. A promising mitigation strategy is to leverage factorized graphical models—such as Functional Graphical Models (FGMs) (Kuba et al., 2024b)—to decompose a high-dimensional function into a sum of local subfunctions defined over smaller cliques. This structured factorization enables each subcomponent to be learned from a subspace with denser data coverage, thereby localizing distribution shifts and enhancing both robustness and OOD generalization (Kuba et al., 2024a). In addition to FGMs, candidate methods for graph discovery – such as those based on GFlowNets –offer alternative avenues to uncover latent causal structures (Deleu et al., 2022). Employing such graphical discovery methods will provide a principled means to address distributional shifts in offline MBO, ultimately leading to more reliable optimization outcomes under limited data coverage.

Advanced Generative Modeling Many existing offline MBO methods employ relatively simplistic generative models. For instance, several approaches use basic encoder-decoder architectures that map protein sequences into a latent logit space for optimization (Trabucco et al., 2022). While such methods capture general machine learning principles, they often overlook the rich biophysical information embedded in the data. In applications like protein design, it is crucial to leverage models that integrate domain-specific knowledge—such as pre-trained language models or structure-aware representations—to more accurately model the underlying data distribution (Watson et al., 2023). By developing generative models that are tailored to the unique characteristics of the target domain, future methods can achieve not only improved optimization performance but also greater interpretability and robustness in real-world applications.

Application to LLM Alignment and AI Safety While offline MBO has traditionally been applied to design tasks in fields such as biology or chemistry, its methodologies can also be used for post-training large language models (LLMs). Approaches like supervised fine-tuning (SFT), RLHF (Ouyang et al., 2022), or RL-based reasoning (Guo et al., 2025), aim to enhance text generation by leveraging reward signals derived from human preference models or other evaluative metrics. Because these reward models are inherently uncertain and prone to reward hacking, adopting the conservative and safe optimization techniques developed in offline MBO can help mitigate such risks. Moreover, since reward hacking can lead to catastrophic misalignment of superhuman-level intelligence toward harmful ends (Bengio et al., 2025), integrating robust optimization algorithms from the offline MBO community can further contribute to LLM safety.

References

Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. Robel: robotics benchmarks for learning with low-cost robots. In *Conf. on Robot Lea. (CoRL)*, 2020.

Rebecca F Alford, Andrew Leaver-Fay, Jeliazko R Jeliazkov, Matthew J OMeara, Frank P DiMaio, Hahnbeom Park, Maxim V Shapovalov, P Douglas Renfrew, Vikram K Mulligan, Kalli Kappel, et al. The rosetta all-atom energy function for macromolecular modeling and design. *Journal of chemical theory and computation*, 2017.

Hossain M. Amir and Takashi Hasegawa. Nonlinear mixed-discrete structural optimization. *Journal of Structural Engineering*, 115, 1989.

Christof Angermüller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*. Pmlr, 2017.

Thomas Bäck. *Evolutionary Algorithms in Theory and Practice : Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996.

Luis A Barrera, Anastasia Vedenko, Jesse V Kurland, Julia M Rogers, Stephen S Gisselbrecht, Elizabeth J Rossin, Jaie Woodard, Luca Mariani, Kian Hong Kock, Sachi Inukai, Trevor Siggers, Leila Shokri, Raluca Gordân, Nidhi Sahni, Chris Cotsapas, Tong Hao, Song Yi, Manolis Kellis, Mark J Daly, Marc Vidal, David E Hill, and Martha L Bulyk. Survey of variation in human transcription factors reveals prevalent DNA binding changes. *Science*, 351, 2016a.

Luis A Barrera et al. Survey of variation in human transcription factors reveals prevalent DNA binding changes. *Science*, 2016b.

Christopher Beckham and Christopher Pal. Conservative objective models are a special kind of contrastive divergence-based energy model, 2023.

Christopher Beckham, Alexandre Piche, David Vazquez, and Christopher Pal. Exploring validation metrics for offline model-based optimisation with diffusion models, 2024.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *The Journal of Machine Learning Research*, 24, 2023.

Yoshua Bengio, Michael Cohen, Damiano Fornasiere, Joumana Ghosn, Pietro Greiner, Matt MacDermott, Sören Mindermann, Adam Oberman, Jesse Richardson, Oliver Richardson, et al. Superintelligent agents pose catastrophic risks: Can scientist ai offer a safer path? *ArXiv preprint*, abs/2502.15657, 2025.

Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8, 2020.

P.A.N. Bosman and D. Thierens. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7, 2003.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *ArXiv preprint*, abs/1606.01540, 2016.

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, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*. Pmlr, 2019.

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59, 2019.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Ziyi Chang, George Alex Koulieris, and Hubert PH Shum. On the design fundamentals of diffusion models: A survey. *ArXiv preprint*, abs/2306.04542, 2023.

Yassine Chemingui, Aryan Deshwal, Trong Nghia Hoang, and Janardhan Rao Doppa. Offline model-based optimization via policy-guided gradient search. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*. AAAI Press, 2024.

Can Chen, Xi Chen, Chen Ma, Zixuan Liu, and Xue Liu. Gradient-based bi-level optimization for deep learning: A survey. *ArXiv preprint*, abs/2207.11719, 2022a.

Can Chen, Yingxue Zhang, Jie Fu, Xue (Steve) Liu, and Mark Coates. Bidirectional learning for offline infinite-width model-based optimization. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022b.

Can Chen, Christopher Beckham, Zixuan Liu, Xue (Steve) Liu, and Chris Pal. Parallel-mentoring for offline model-based optimization. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023a.

Can Chen, Yingxue Zhang, Xue Liu, and Mark Coates. Bidirectional learning for offline model-based biological sequence design. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*. Pmlr, 2023b.

Can Chen, Jingbo Zhou, Fan Wang, Xue Liu, and Dejing Dou. Structure-aware protein self-supervised learning. *Bioinformatics*, 39, 2023c.

Can Chen, Christopher Beckham, Zixuan Liu, Xue Liu, and Christopher Pal. Robust guided diffusion for offline black-box optimization. *Transactions on Machine Learning Research*, 2024.

Can Chen, Karla-Luise Herpoldt, Chenchao Zhao, Zichen Wang, Marcus Collins, Shang Shang, and Ron Benson. Affinityflow: Guided flows for antibody affinity maturation. *ArXiv preprint*, abs/2502.10365, 2025.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*. Pmlr, 2018.

Franklin Y. Cheng and X. S. Li. Generalized center method for multiobjective engineering optimization. *Engineering Optimization*, 31, 1999.

C A Coello Coello and G T Pulido. Multiobjective structural optimization using a microgenetic algorithm. *Structural and Multidisciplinary Optimization*, 30, 2005.

Farhan Damani, David H Brookes, Theodore Sternlieb, Cameron Webster, Stephen Malina, Rishi Jajoo, Kathy Lin, and Sam Sinai. Beyond the training set: an intuitive method for detecting distribution shift in model-based optimization. *ArXiv preprint*, abs/2311.05363, 2023.

Steven J. Daniels, Alma A. M. Rahat, Richard M. Everson, Gavin R. Tabor, and Jonathan E. Fieldsend. A suite of computationally expensive shape optimisation problems using computational fluid dynamics. In *Parallel Problem Solving from Nature – PPSN XV*. Springer International Publishing, 2018.

Cuong Dao, Phi Le Nguyen, Truong Thao Nguyen, and Nghia Hoang. Incorporating surrogate gradient norm to improve offline optimization techniques. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a.

Manh Cuong Dao, Phi Le Nguyen, Truong Thao Nguyen, and Trong Nghia Hoang. Boosting offline optimizers with surrogate sensitivity. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024b.

Manh Cuong Dao, The Hung Tran, Phi Le Nguyen, Thao Nguyen Truong, and Trong Nghia Hoang. From Low To High-value Designs: Offline Optimization Via Generalized Diffusion, 2025.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *ArXiv preprint*, abs/2307.08698, 2023.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18, 2014.

Kalyanmoy Deb and Aravind Srinivasan. Innovization: innovating design principles through optimization. *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006.

Kalyanmoy Deb and Santosh Tiwari. Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research*, 185, 2008.

Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. *Scalable Test Problems for Evolutionary Multiobjective Optimization*, pp. 105–145. Springer London, 2005.

Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. In *Uncertainty in Artificial Intelligence, Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022, 1-5 August 2022, Eindhoven, The Netherlands*, volume 180 of *Proceedings of Machine Learning Research*. Pmlr, 2022.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350, 2012.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021.

Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems: First International Workshop*, 2000.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Katharina Eggensperger, Philipp Müller, Neeratyoy Mallik, Matthias Feurer, René Sass, Aaron Klein, Noor Awad, Marius Lindauer, and Frank Hutter. Hpobench: A collection of reproducible multi-fidelity benchmark problems for hpo, 2021.

Clara Fannjiang and Jennifer Listgarten. Autofocused oracles for model-based design. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

A. Farhang-Mehr and Shapour Azarm. Entropy-based multi-objective genetic algorithm for design optimization. *Structural and Multidisciplinary Optimization*, 24, 2002.

Peter I Frazier. A tutorial on bayesian optimization. *ArXiv preprint*, abs/1807.02811, 2018.

Nathan C. Frey, Daniel Berenberg, Karina Zadorozhny, Joseph Kleinhenz, Julien Lafrance-Vanasse, Isidro Hötzel, Yan Wu, Stephen Ra, Richard Bonneau, Kyunghyun Cho, Andreas Loukas, Vladimir Gligorijevic, and Saeed Saremi. Protein discovery with discrete walk-jump sampling. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

Nathan C Frey, Isidro Hotzel, Samuel D Stanton, Ryan L Kelly, Robert G Alberstein, Emily K Makowski, Karolis Martinkus, Dan Berenberg, Jack Bevers III, Tyler Bryson, et al. Lab-in-the-loop therapeutic antibody design with deep learning. *bioRxiv*, 2025.

Justin Fu and Sergey Levine. Offline model-based optimization via normalized maximum likelihood estimation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Tianfan Fu, Wenhao Gao, Cao Xiao, Jacob Yasonik, Connor W Coley, and Jimeng Sun. Differentiable scaffolding tree for molecular optimization. *ArXiv preprint*, abs/2109.10469, 2021.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2016.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Jimenez Rezende, and S. M. Ali Eslami. Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*. Pmlr, 2018.

Anna Gaulton, Louisa J Bellis, A Patricia 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 Res*, 40, 2011.

Pouya M Ghari, Alex Tseng, Gökcen Eraslan, Romain Lopez, Tommaso Biancalani, Gabriele Scalia, and Ehsan Hajiramezanali. Generative flow networks assisted biological sequence editing. In *NeurIPS 2023 Generative AI and Biology (GenBio) Workshop*, 2023.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4, 2018.

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: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 2014.

Robert B. Gramacy and Herbert K. H. Lee. Cases for the nugget in modeling computer experiments, 2010.

Nate Gruver, Samuel Stanton, Nathan C. Frey, Tim G. J. Rudner, Isidro Hötzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew Gordon Wilson. Protein design with guided discrete diffusion. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv preprint*, abs/2501.12948, 2025.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2018.

Rishin Haldar and Debajyoti Mukhopadhyay. Levenshtein distance technique in dictionary lookup methods: An improved approach. *arXiv preprint arXiv:1101.1232*, 2011.

Kam Hamidieh. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154, 2018.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 1990.

James Harrison, John Willes, and Jasper Snoek. Variational bayesian last layers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14, 2002.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *ArXiv preprint*, abs/2207.12598, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Minh Hoang, Azza Fadhel, Aryan Deshwal, Jana Doppa, and Trong Nghia Hoang. Learning surrogates for offline black-box optimization via gradient matching. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024.

S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

Chenqing Hua, Yong Liu, Dinghuai Zhang, Odin Zhang, Sitao Luan, Kevin K Yang, Guy Wolf, Doina Precup, and Shuangjia Zheng. Enzymeflow: Generating reaction-specific enzyme catalytic pockets through flow matching and co-evolutionary dynamics. *ArXiv preprint*, abs/2410.00327, 2024.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *ArXiv preprint*, abs/2102.09548, 2021.

Evan J. Hughes. Radar waveform optimisation as a many-objective application benchmark. In *Evolutionary Multi-Criterion Optimization*. Springer Berlin Heidelberg, 2007.

Hisao Ishibuchi, Naoya Akedo, and Yusuke Nojima. Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. *IEEE Transactions on Evolutionary Computation*, 19, 2015.

Himanshu Jain and Kalyanmoy Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18, 2014.

Moksh Jain, Emmanuel Bengio, Alex Hernández-García, Jarrid Rector-Brooks, Bonaventure F. P. Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological sequence design with gflownets. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*. Pmlr, 2022.

Moksh Jain, Sharath Chandra Raparthy, Alex Hernández-García, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective gflownets. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*. Pmlr, 2023.

Momin Jamil and Xin She Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4:150, 2013.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Bowen Jing, Bonnie Berger, and Tommi S. Jaakkola. Alphafold meets flow matching for generating protein ensembles. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596, 2021.

Bhairavi Kannan and S. N. Kramer. An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design*, 116, 1994.

Hyeonah Kim, Minsu Kim, Taeyoung Yun, Sanghyeok Choi, Emmanuel Bengio, Alex Hernández-García, and Jinkyoo Park. Improved off-policy reinforcement learning in biological sequence design. In *NeurIPS 2024 Workshop on AI for New Drug Modalities*, 2024a.

Jungtaek Kim. BayesO Benchmarks: Benchmark functions for Bayesian optimization, 2023.

Jungtaek Kim and Seungjin Choi. Bayeso: A bayesian optimization framework in python. *Journal of Open Source Software*, 2023.

Minsu Kim, Federico Berto, Sungsoo Ahn, and Jinkyoo Park. Bootstrapped training of score-conditioned generator for offline design of biological sequences. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Minsu Kim, Joohwan Ko, Taeyoung Yun, Dinghuai Zhang, Ling Pan, Woochang Kim, Jinkyoo Park, Emmanuel Bengio, and Yoshua Bengio. Learning to scale logits for temperature-conditional gflownets. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024b.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Andrew Kirjner, Jason Yim, Raman Samusevich, Shahar Bracha, Tommi S. Jaakkola, Regina Barzilay, and Ila R. Fiete. Improving protein optimization with smoothed fitness landscapes. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

Justin R Klesmith, John-Paul Bacik, Ryszard Michalczyk, and Timothy A Whitehead. Comprehensive Sequence-Flux mapping of a levoglucosan utilization pathway in e. coli. *ACS Synth Biol*, 4, 2015.

Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43, 2020.

Takehisa Kohira, Hiromasa Kemmotsu, Oyama Akira, and Tomoaki Tatsukawa. Proposal of benchmark problem based on real-world car structure design optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Gecco '18. Association for Computing Machinery, 2018.

Sathvik Kolli. Conflict-averse gradient optimization of ensembles for effective offline model-based optimization, 2023.

Siddarth Krishnamoorthy, Satvik Mehul Mashkaria, and Aditya Grover. Diffusion models for black-box optimization. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*. Pmlr, 2023.

Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, 2012.

Jakub Grudzien Kuba, Pieter Abbeel, and Sergey Levine. Cliqueformer: Model-based optimization with structured transformers. *ArXiv preprint*, abs/2410.13106, 2024a.

Kuba Grudzien Kuba, Masatoshi Uehara, Sergey Levine, and Pieter Abbeel. Functional graphical models: Structure enables offline data-driven optimization. In *International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain*, volume 238 of *Proceedings of Machine Learning Research*. Pmlr, 2024b.

Christopher Kuenneth and Rampi Ramprasad. polybert: a chemical language model to enable fully machine-driven ultrafast polymer informatics. *Nature communications*, 14, 2023.

Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Aviral Kumar, Amir Yazdanbakhsh, Milad Hashemi, Kevin Swersky, and Sergey Levine. Data-driven offline optimization for architecting hardware accelerators. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

Salem Lahlou, Moksh Jain, Hadi Nekoei, Victor I Butoi, Paul Bertin, Jarrid Rector-Brooks, Maksym Korablyov, and Yoshua Bengio. DEUP: Direct epistemic uncertainty prediction. *Transactions on Machine Learning Research*, 2023.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017.

J. Lampinen and Ivan Zelinka. Mixed integer-discrete-continuous optimization by differential evolution - part 2: a practical example, 2000.

Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, and Wei-Ning Hsu. Voicebox: Text-guided multilingual universal speech generation at scale. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1, 2006.

Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*. Pmlr, 2023.

Seunghun Lee, Jinyoung Park, Jaewon Chu, Minseo Yoon, and Hyunwoo J. Kim. Latent bayesian optimization via autoregressive normalizing flows. In *The Thirteenth International Conference on Learning Representations*, 2025.

Xingtao Liao, Qing Li, Xujing Yang, Weigang Zhang, and Wei Li. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary Optimization*, 35, 2008.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

Dianbo Liu, Moksh Jain, Bonaventure F. P. Dossou, Qianli Shen, Salem Lahlou, Anirudh Goyal, Nikolay Malkin, Chris Chinenye Emezue, Dinghuai Zhang, Nadhir Hassen, Xu Ji, Kenji Kawaguchi, and Yoshua Bengio. Gflowout: Dropout with generative flow networks. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*. Pmlr, 2023.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Jeremiah Z. Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016.

Ronny Lorenz, Stephan H. Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F. Stadler, and Ivo L. Hofacker. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6, 2011.

Zhichao Lu, Ran Cheng, Yaochu Jin, Kay Chen Tan, and Kalyanmoy Deb. Neural architecture search as multiobjective optimization benchmarks: Problem formulation and performance assessment. *IEEE transactions on evolutionary computation*, 2023.

Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018.

Shitong Luo, Yufeng Su, Xingang Peng, Sheng Wang, Jian Peng, and Jianzhu Ma. Antigen-specific antibody design and optimization with diffusion-based generative models for protein structures. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Thibaut Lust and Jacques Teghem. The multiobjective traveling salesman problem: A survey and a new approach. In *Advances in Multi-Objective Nature Inspired Computing*. Springer, 2010.

Xiaoliang Ma, Yanan Yu, Xiaodong Li, Yutao Qi, and Zexuan Zhu. A survey of weight vector adjustment methods for decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2020.

Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*. Pmlr, 2023.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward J. Hu, Katie Everett, Dinghuai Zhang, and Yoshua Bengio. Gflownets and variational inference. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

Satvik Mehul Mashkaria, Siddarth Krishnamoorthy, and Aditya Grover. Generative pretraining for black-box optimization. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*. Pmlr, 2023.

Daniel Melamed, David L Young, Caitlin E Gamble, Christina R Miller, and Stanley Fields. Deep mutational scanning of an RRM domain of the saccharomyces cerevisiae poly(a)-binding protein. *Rna*, 19, 2013.

Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks, 2015.

Radford M Neal. Probabilistic inference using markov chain monte carlo methods, 1993.

Tung Nguyen, Sudhanshu Agrawal, and Aditya Grover. Expt: Synthetic pretraining for few-shot experimental design. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Pierce J. Ogden, Eric D. Kelsic, Sam Sinai, and George M. Church. Comprehensive aav capsid fitness landscape reveals a viral gene and enables machine-guided design. *Science*, 366, 2019.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Michael Parsons and Randall Scott. Formulation of multicriterion design optimization problems for solution with scalar numerical optimization methods. *Journal of Ship Research*, 48, 2004.

Victor Picheny, Tobias Wagner, and David Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Struct. Multidiscip. Optim.*, 48, 2013.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*. Pmlr, 2023.

Han Qi, Yi Su, Aviral Kumar, and Sergey Levine. Data-driven offline decision-making via invariant representation learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022a.

Han Qi, Yi Su, Aviral Kumar, and Sergey Levine. Data-driven model-based optimization via invariant representation learning. In *Proc. Adv. Neur. Inf. Proc. Syst (NeurIPS)*, 2022b.

Han Qi, Xinyang Geng, Stefano Rando, Iku Ohama, Aviral Kumar, and Sergey Levine. Latent conservative objective models for data-driven crystal structure prediction, 2023.

Hong Qian, Yiyi Zhu, Xiang Shu, Xin An, Yaolin Wen, Shuo Liu, Huakang Lu, Aimin Zhou, Ke Tang, and Yang Yu. SOO-bench: Benchmarks for evaluating the stability of offline black-box optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Tapabrata Ray and K. Liew. Liew, k.: A swarm metaphor for multiobjective design optimization. engineering optimization 34, 141-153. *Engineering Optimization*, 34, 2002.

Tapabrata Ray, K. Tai, and KIN SEOW. Multiobjective design optimization by an evolutionary algorithm. *Engineering Optimization*, 33, 2001.

Zhizhou Ren, Jiahan Li, Fan Ding, Yuan Zhou, Jianzhu Ma, and Jian Peng. Proximal exploration for model-guided protein sequence design. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*. Pmlr, 2022.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *ArXiv preprint*, abs/1609.04747, 2016.

Ruslan Salakhutdinov. Deep learning. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. Acm, 2014.

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. *Nat Biotechnol*, 37, 2019a.

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*, 2019b.

Karen S. Sarkisyan, Dmitry A. Bolotin, Margarita V. Meer, Dinara R. Usmanova, Alexander S. Mishin, George V. Sharonov, Dmitry N. Ivankov, Nina G. Bozhanova, Mikhail S. Baranov, Onuralp Soylemez, Natalya S. Bogatyreva, Peter K. Vlasov, Evgeny S. Egorov, Maria D. Logacheva, Alexey S. Kondrashov, Dmitry M. Chudakov, Ekaterina V. Putintseva, Ilgar Z. Mamedov, Dan S. Tawfik, Konstantin A. Lukyanov, and Fyodor A. Kondrashov. Local fitness landscape of the green fluorescent protein. *Nature*, 533, 2016a.

Karen S Sarkisyan et al. Local fitness landscape of the green fluorescent protein. *Nature*, 2016b.

Joost Schymkowitz, Jesper Borg, Francois Stricher, Robby Nys, Frederic Rousseau, and Luis Serrano. The foldx web server: an online force field. *Nucleic acids research*, 33, 2005.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Samuel Stanton, Wesley J. Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Accelerating bayesian optimization for biological sequence design with denoising autoencoders. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*. Pmlr, 2022.

Samuel Stanton, Robert Alberstein, Nathan Frey, Andrew Watkins, and Kyunghyun Cho. Closed-form test functions for biophysical sequence optimization algorithms. *ArXiv preprint*, abs/2407.00236, 2024.

Lea M. Starita, Jonathan N. Pruneda, Russell S. Lo, Douglas M. Fowler, Helen J. Kim, Joseph B. Hiatt, Jay Shendure, Peter S. Brzovic, Stanley Fields, and Rachel E. Klevit. Activity-enhancing mutations in an e3 ubiquitin ligase identified by high-throughput mutagenesis. *Proceedings of the National Academy of Sciences*, 110, 2013.

Hannes Stärk, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi S. Jaakkola. Dirichlet flow matching with applications to DNA sequence design. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024.

S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets, 2013.

Rong-Xi Tan, Ke Xue, Shen-Huan Lyu, Haopu Shang, yaowang, Yaoyuan Wang, Fu Sheng, and Chao Qian. Offline model-based optimization by learning to rank. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ryoji Tanabe and Hisao Ishibuchi. An easy-to-use real-world multi-objective optimization problem suite. *Applied Soft Computing*, 89, 2020.

Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*. Acm, 2008.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

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, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*. Pmlr, 2021.

Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*. Pmlr, 2022.

Rajkumar Vaidyanathan, Kevin Tucker, Nilay Papila, and Wei Shyy. Cfd-based design optimization for single element rocket injector. *NASA Technical Reports Server*, 2003.

B. Valladares and Others. Bayesian optimization of active materials for lithium-ion batteries. In *Proceedings of IEEE IECON*, 2021.

David A. van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithm test suites. In *Proceedings of the 1999 ACM Symposium on Applied Computing*, Sac '99. Association for Computing Machinery, 1999.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017.

Siddarth Venkatraman, Moksh Jain, Luca Scimeca, Minsu Kim, Marcin Sendera, Mohsin Hasan, Luke Rowe, Sarthak Mittal, Pablo Lemos, Emmanuel Bengio, Alexandre Adam, Jarrid Rector-Brooks, Yoshua Bengio, Glen Berseth, and Nikolay Malkin. Amortizing intractable inference in diffusion models for vision, language, and control. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Vanessa Volz, Boris Naujoks, Pascal Kerschke, and Tea Tušar. Single- and multi-objective game-benchmark for evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Gecco '19. Association for Computing Machinery, 2019.

Zichen Wang, Steven A Combs, Ryan Brand, Miguel Romero Calvo, Panpan Xu, George Price, Nataliya Golovach, Emmanuel O Salawu, Colby J Wise, Sri Priya Ponnapalli, et al. Lm-gvp: an extensible sequence and structure informed deep learning framework for protein property prediction. *Scientific reports*, 12, 2022.

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620, 2023.

Jochen Weile, Song Sun, Atina G Cote, Jennifer Knapp, Marta Verby, Joseph C Mellor, Yingzhou Wu, Carles Pons, Cassandra Wong, Natascha van Lieshout, Fan Yang, Murat Tasan, Guihong Tan, Shan Yang, Douglas M Fowler, Robert Nussbaum, Jesse D Bloom, Marc Vidal, David E Hill, Patrick Aloy, and Frederick P Roth. A framework for exhaustively mapping functional missense variants. *Mol Syst Biol*, 13, 2017.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. Omnipress, 2011.

Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in neural information processing systems*, 8, 1995.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, volume 51 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2016.

Emily E Wrenbeck, Laura R Azouz, and Timothy A Whitehead. Single-mutation fitness landscapes for an enzyme on multiple substrates reveal specificity is globally encoded. *Nature Communications*, 8, 2017.

Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*. Pmlr, 2020.

Minghao Xu, Zuobai Zhang, Jiarui Lu, Zhaocheng Zhu, Yangtian Zhang, Chang Ma, Runcheng Liu, and Jian Tang. PEER: A comprehensive and multi-task benchmark for protein sequence understanding. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Ke Xue, Rong-Xi Tan, Xiaobin Huang, and Chao Qian. Offline multi-objective optimization. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024.

Michael S. Yao, Yimeng Zeng, Hamsa Bastani, Jacob R. Gardner, James C. Gee, and Osbert Bastani. Generative adversarial model-based optimization via source critic regularization. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Michael S Yao, James C Gee, and Osbert Bastani. Diversity by design: Leveraging distribution matching for offline model-based optimization. *ArXiv preprint*, abs/2501.18768, 2025.

Fei YE, Zaixiang Zheng, Dongyu Xue, Yuning Shen, Lihao Wang, Yiming Ma, Yan Wang, Xinyou Wang, Xiangxin Zhou, and Quanquan Gu. Proteinbench: A holistic evaluation of protein foundation models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Peiyu Yu, Dinghuai Zhang, Hengzhi He, Xiaojian Ma, Ruiyao Miao, Yifan Lu, Yasi Zhang, Deqian Kong, Ruiqi Gao, Jianwen Xie, et al. Latent energy-based odyssey: Black-box optimization via expanded exploration in the energy-based latent space. *ArXiv preprint*, abs/2405.16730, 2024.

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: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Ye Yuan, Can Chen, Zixuan Liu, Willie Neiswanger, and Xue (Steve) Liu. Importance-aware co-teaching for offline model-based optimization. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Ye Yuan, Can Chen, Christopher Pal, and Xue Liu. Paretoflow: Guided flows in multi-objective optimization. *ArXiv preprint*, abs/2412.03718, 2024a.

Ye Yuan, Youyuan Zhang, Can Chen, Haolun Wu, Zixuan Li, Jianmo Li, James J. Clark, and Xue Liu. Design editing for offline model-based optimization, 2024b.

Taeyoung Yun, Sujin Yun, Jaewoo Lee, and Jinkyoo Park. Guided trajectory generation with diffusion models for offline model-based optimization. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Taeyoung Yun, Kiyoung Om, Jaewoo Lee, Sujin Yun, and Jinkyoo Park. Posterior inference with diffusion models for high-dimensional black-box optimization. *ArXiv preprint*, abs/2502.16824, 2025.

Sandra Zajac and Sandra Huber. Objectives and methods in multi-objective routing problems: a survey and classification scheme. *European Journal of Operational Research*, 290, 2021a.

Sandra Zajac and Sandra Huber. Objectives and methods in multi-objective routing problems: a survey and classification scheme. *European Journal of Operational Research*, 2021b.

Dinghuai Zhang, Nikolay Malkin, Zhen Liu, Alexandra Volokhova, Aaron C. Courville, and Yoshua Bengio. Generative flow networks for discrete probabilistic modeling. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*. Pmlr, 2022.

Qiang Zhang, Ruida Zhou, Yang Shen, and Tie Liu. From function to distribution modeling: A pac-generative approach to offline optimization, 2024.

Zuobai Zhang, Chuanrui Wang, Minghao Xu, Vijil Chenthamarakshan, Aurélie Lozano, Payel Das, and Jian Tang. A systematic study of joint representation learning on protein sequences and structures. *ArXiv preprint*, abs/2303.06275, 2023.

Yiyang Zhao, Linnan Wang, Kevin Yang, Tianjun Zhang, Tian Guo, and Yuandong Tian. Multi-objective optimization by learning space partition. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky TQ Chen. Guided flows for generative modeling and decision making. *ArXiv preprint*, abs/2311.13443, 2023.

E Zitzler, K Deb, and L Thiele. Comparison of multiobjective evolutionary algorithms: empirical results. *Evol Comput*, 8, 2000.

Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms — a comparative case study. In *Parallel Problem Solving from Nature — PPSN V*. Springer Berlin Heidelberg, 1998.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.