Offline Data-Driven Optimization: Benchmarks, Algorithms and Applications

Xinyang Geng

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-219
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-219.html

August 11, 2023

Offline Data-Driven Optimization: Benchmarks, Algorithms and Applications

By

Xinyang Geng

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Sergey Levine, Chair
Professor Nilah Ioannidis
Professor Jiantao Jiao
Professor Abhishek Gupta

Summer 2023

Offline Data-Driven Optimization:
Benchmarks, Algorithms and Applications

By

Xinyang Geng

Abstract

Offline Data-Driven Optimization:
Benchmarks, Algorithms and Applications

By

Xinyang Geng

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Sergey Levine, Chair

Black-box model-based optimization problems, where the goal is to find a design input that maximizes an unknown objective function, are ubiquitous in a wide range of domains, such as the design of proteins, DNA sequences, aircraft, and robots. Solving model-based optimization problems typically requires actively querying the unknown objective function on design proposals, which means physically building the candidate molecule, aircraft, or robot, testing it to obtain the result. This process can be expensive and time consuming, and one might instead prefer to optimize for the best design using only the data one already has. This setting, called offline model-based optimization (MBO), poses substantial and different algorithmic challenges than more commonly studied online techniques. In this thesis, I will cover how to build benchmarks and algorithms to tackle these challenges. In particular, I will first define the offline MBO problem formally, and identify the common challenging properties associated with real-world offline MBO problems. I will then present Design-Bench, a benchmark for evaluating offline MBO methods with a suite of diverse and realistic tasks derived from real-world optimization problems. With the benchmark set up, I will describe conservative objective models (COMs), a surprisingly simple but effective method for tackling offline MBO problems. Finally, I will cover applications of offline MBO in computational chemistry and synthetic biology to demonstrate how variants of COMs can be applied to solve real-world scientific problems.

To my parents.

Contents

Contents **ii**

1 Introduction **1**

2 Design-Bench: Benchmarks for Data-Driven Offline Model-Based Optimization **4**
2.1 Introduction 4
2.2 Offline Model-Based Optimization (Offline MBO) Problem Statement . 6
2.3 Related Work 8
2.4 Design-Bench Benchmark Tasks 9
2.5 Task Properties, Challenges and Considerations 12
2.6 Algorithm Implementations 15
2.7 Benchmarking Prior Methods 17
2.8 Discussion and Conclusion 19

3 Conservative Objective Models for Offline MBO **20**
3.1 Introduction 20
3.2 Preliminaries 22
3.3 Conservative Objective Models for Offline Model-Based Optimization . 23
3.3.1 Learning Conservative Objective Models (COMs) 23
3.3.2 Optimizing a Conservative Objective Model 25
3.3.3 Using COMs for MBO: Additional Decisions 26
3.3.4 Overall Algorithm and Practical Implementation 27
3.4 Theoretical Analysis of COMs 27
3.5 Related Work 30
3.6 Experimental Evaluation 31
3.6.1 Empirical Performance on Benchmark Tasks 31
3.6.2 Ablation Experiments 33
3.7 Discussion and Conclusion 35

4 Latent Conservative Objective Models for Offline Data-Driven Crystal Structure Prediction **37**

4.1 Introduction . . . 37
4.2 Background and Definitions for Crystal Structures and Materials . . . 39
4.3 Problem Statement, Dataset, and Evaluation . . . 40
4.3.1 Datasets for Training . . . 41
4.3.2 Held-Out Evaluation Datasets . . . 41
4.3.3 Evaluation Protocol . . . 42
4.4 LCOMs: Latent Conservative Objective Models for Structure Prediction 42
4.4.1 Transforming Crystal Structures to a Latent Representation . . 43
4.4.2 Conservative Optimization in Latent Space . . . 44
4.4.3 Implementation Details . . . 45
4.5 Related Work . . . 45
4.6 Experimental Evaluation . . . 47
4.7 Discussion, Future Directions, and Limitations . . . 51

5 Designing Cell Type-Specific Promoter Sequences via Conservative Model-Based Optimization 52
5.1 Introduction . . . 52
5.2 Preliminaries of Offline Model-Based Optimization . . . 53
5.3 Problem Setup . . . 54
5.4 CPD: Conservative Promoter Design . . . 55
5.4.1 Pre-Training on Promoter Driven Expression Datasets . . . 56
5.4.2 Finetuning on Target Cell Type Dataset with Conservatism . . 57
5.4.3 Balancing Optimality and Diversity in Promoter Design . . . 57
5.5 Experiments . . . 59
5.5.1 Prior Approaches and Baselines . . . 60
5.5.2 Main Results . . . 60
5.5.3 Ablation Studies . . . 61
5.5.4 Motif composition of designed sequences . . . 63
5.6 Conclusion . . . 63

6 Conclusion 66

A Appendix for Design-Bench 83
A.1 Data Collection . . . 83
A.1.1 TF Bind 8 and TF Bind 10 . . . 83
A.1.2 ChEMBL . . . 83
A.1.3 Superconductor . . . 84
A.1.4 Ant & D'Kitty Morphology . . . 85
A.1.5 NAS . . . 85
A.1.6 Hopper Controller . . . 85
A.2 Oracle Functions . . . 86
A.2.1 TF Bind 8 and TF Bind 10 . . . 86
A.2.2 ChEMBL . . . 86

A.2.3 Superconductor . . . 87
A.2.4 Ant & D'Kitty Morphology . . . 87
A.2.5 NAS . . . 87
A.2.6 Hopper Controller . . . 87
A.3 Experimental Details . . . 88
A.3.1 Objective Normalization . . . 88
A.3.2 50th Percentile Experiment Results . . . 88
A.3.3 Unnormalized Experimental Results . . . 89
A.3.4 Computation Resources . . . 89
A.4 Additional MBO Tasks That Were Discarded From Our Benchmark . . . 89
A.4.1 GFP . . . 90
A.4.2 UTR . . . 90
A.4.3 Additional Experimental Results . . . 91
A.5 Normalization Of Inputs and Outputs Is Important for Gradient Ascent 91
A.6 Hyperparameter Selection Workflow . . . 93
A.6.1 Strategy For Autofocused CbAS . . . 93
A.6.2 Strategy For CbAS . . . 94
A.6.3 Strategy For MINs . . . 94
A.6.4 Strategy For Gradient Ascent . . . 95
A.6.5 Strategy For REINFORCE . . . 95
A.6.6 Strategy For Bayesian Optimization . . . 95
A.6.7 Strategy For Covariance Matrix Adaptation (CMA-ES) . . . 96
A.6.8 Strategy For Conservative Objective Models (COMs) . . . 96

B Appendix for Conservative Objective Models 97
B.1 Method Details . . . 97
B.1.1 Additional Results . . . 97
B.1.2 Implementation details . . . 97
B.1.3 Benchmarking Details . . . 99
B.2 Proof of Theorem 3.4.1 . . . 100
B.3 Network Details . . . 102
B.4 Data Collection . . . 102
B.4.1 TF Bind 8 . . . 102
B.4.2 GFP . . . 103
B.4.3 UTR . . . 103
B.4.4 Superconductor . . . 104
B.4.5 Hopper Controller . . . 104
B.4.6 Ant & D'Kitty Morphology . . . 105
B.5 Oracle Functions . . . 105
B.5.1 TF Bind 8 . . . 105
B.5.2 GFP . . . 106
B.5.3 UTR . . . 106
B.5.4 Superconductor . . . 106

B.5.5 HopperController 106
B.5.6 Ant & D'Kitty Morphology 107

C Appendix for Crystal Structure Design **108**
C.1 Additional Ablation Study 108
C.2 Details of Our Simulator 109
C.3 Experiment Details 109

D Appendix for Promoter Design **111**
D.1 Additional Results 111
D.2 Hyperparameters and Experiment Details 111
D.2.1 Details for Pre-training 111
D.2.2 Details for Ensemble Oracle Model 112
D.2.3 Details for CPD 113
D.2.4 Details for Motif Tiling 113
D.2.5 Details for DENs 114

Acknowledgments

Throughout my journey at Berkeley, I am very fortunate to have many people to thank.

First and foremost, I am extremely grateful to my Advisor, Sergey Levine, for his guidance and support throughout my PhD program. I was lucky to have Sergey as my advisor ever since I was an undergraduate student at Berkeley. Throughout the years, Sergey has sharpened my research skills and shaped my research tastes. He has taught me to focus on the most impactful problems of research and gave me great freedom to explore all the directions I am interested in.

I would like to thank my dissertation committee members, Nilah Ioannidis, Abhishek Gupta and Jiantao Jiao for supporting me at this crucial milestone of my PhD. They have given me great feedback for improving this dissertation.

Before pursuing my PhD, I spent two years doing research as an undergraduate student at Berkeley. I was fortunate to work with Sergey Levine, Pieter Abbeel, Alexei Efros and their postdocs and students (at that time) Marvin Zhang, Carlos Florensa, David Held, Junyan Zhu, Phillip Isola, Richard Zhang. The wonderful experience of undergraduate research made me decide to continue my research journey and stay at Berkeley to pursue my PhD. I am especially grateful to Marvin Zhang for introducing me to research and mentoring me on my first research project.

Over the course of my PhD, I had the fortune to work with many amazing collaborators. My collaborators, Aviral Kumar, Hao Liu, Tianhe Yu, Charlie Snell, Aniketh Reddy, Ailin Chen, Amy Lu, Eric Wallace, Arnav Gudibande, Michael Herschl, Stefano Rando, Benjamin Eysenbach, Rishabh Agarwal, Yutong Bai, Yi Su, Kristian Hartikainen, Tuomas Haarnoja, Lisa Lee, George Tucker, Jianlan Luo, Charles Xu, Gilbert Feng, Kuan Fang, Liam Tan, James Bradbury, Rafi Witten, Stefan Schaal, Dawn Song, Russ Salakhutdinov, Dale Schuurmans, Patrick Hsu, Grace Gu, Abhishek Gupta, Chelsea Finn, Pieter Abbeel and Nilah Ioannidis have all taught me so much and helped me build my research skills and vision.

RAIL lab has one of the best undergraduate research programs in Berkeley, and I had the fortune to mentor some of the best undergraduates, Brandon Trabucco, Russell Mendonca, Kevin Li, Sathvik Kolli and Han Qi. It has been a great experience working with all of them.

The Berkeley AI Research lab has been a great place to study and do research. I thank Michael Janner, Dibya Ghosh, Laura Smith, Colin Li, Katie Kang, Michael Chang, Mitsuhiko Nakamoto, Joey Hong, Kevin Black, Vivek Myers, Seohong Park, Kyle Stachowicz, Philip Ball, Manan Tomar, Oleh Rybkin, Marwa Abdulhai, Kuba Grudzien, Dhruv Shah, Homer Walke, Simon Zhai, Ilya Kostrikov, Coline Devin, Anusha Nagabandi, Natasha Jaques, Dinesh Jarayaman, Rowan McAllister, Vitchyr Pong, Kelvin Xu, Justin Fu, JD Co-Reyes, Jason Peng, Siddharth Reddy, Amy Zhang,

Glen Berseth, Frederik Ebert, Aurick Zhou, Avi Singh, Ashvin Nair, Sandy Huang, Alex Lee, Karl Pertsch, Erin Grant, Sasha Sax, Philippe Hansen-Estruch, Philipp Wu, Liamnin Zheng, Hao Zhang, Zhuohan Li, Xingyu Lin, Ademi Adeniji, Yuqing Du, Fangchen Liu, Olivia Watkins, Sherry Yang, Kevin Zakka, Ajay Jain, Gregory Kahn, Kate Rakelly, Ignasi Clavera for making my stay at the lab so wonderful.

During the summer of my 4th year at Berkeley, I took a research detour to work on self-supervised learning and language models during my internship at Google. I would like to thank Igor Mordatch, Lisa Lee and Sharan Narang for mentoring me in this new direction.

Between my undergraduate and PhD, I spent one year in the Google AI Residency program. I am grateful to my mentor Jeffrey Pennington, Hossein Mobahi, Ofir Nachum and my fellow residents and colleagues at Google. I am especially grateful to Lechao Xiao for introducing me to many theoretical aspects of machine learning.

My journey would have never been so wonderful without my friends. I thank Can Koc, Cem Koc, Alan Li, Rahul Verma, Brian Su, Kevin Arfin, Can Kabuloglu, Lucas Zhang, Jazlyn Li, Qiyin Wu, Zihao Jing, Ziyun Wang, Weiyi Liu, Zheng Dai, Zhenyang Zhang, Jialun Zhang, Yichao Feng, Qin Yang, Yunjia Zhou, Yide Shentu, Haoran Tang, Yan Duan, Xi Chen, Hanqing Liu and Gefei Li and many others for making my journey so enjoyable.

Finally, I would like to thank my parents and family for their unwavering support throughout my journey and keeping me sane in the most difficult times.

Chapter 1

Introduction

Computational design problems, where the goal is to find a design that maximizes a given objective function, is probably one of the most common problems across many scientific and engineering disciplines. In most settings, the exact form of the objective function is unknown, and the objective values of a novel design can only be determined by running a computer simulation or performing a real-world experiment. This problem of optimizing an unknown function by only observing its input and output is known as black-box model-based optimization (MBO), and is typically tackled via data-driven approaches. From protein engineering [97], molecule design [36] to robot engineering [70], researchers have made significant progresses in applying machine learning based methods to optimization problems over structured design spaces and generated better designs than what human experts can produce.

Typically, existing methods for black-box MBO problem learn a proxy function from the observed data to represent the unknown objective function, and then optimize the learned proxy function to propose new designs. These learned proxy functions usually have errors compared to the ground truth. In order to prevent these errors from affecting the optimization results, existing methods often require *online* data collection, where the proposed new designs are evaluated under the ground truth objective and the results are appended to the dataset to correct the errors of the learned proxy for the next iteration of optimization. This strategy is very effective in settings where the ground truth objective function is fast and inexpensive to evaluate, such as via computer simulation. However, in many important real-world problems, online data collection is often expensive, time consuming and can even be dangerous: evaluating the designs in protein engineering requires synthesizing the candidate protein structures, which can take months to complete in the wet lab; evaluating a new drug design might involve testing it on animal or human subjects. Such high cost of evaluation often prohibits the use of online MBO methods, and therefore a desirable alternative approach is to develop *offline* MBO methods that produces optimized designs by only leveraging previously collected datasets without querying the objective function.

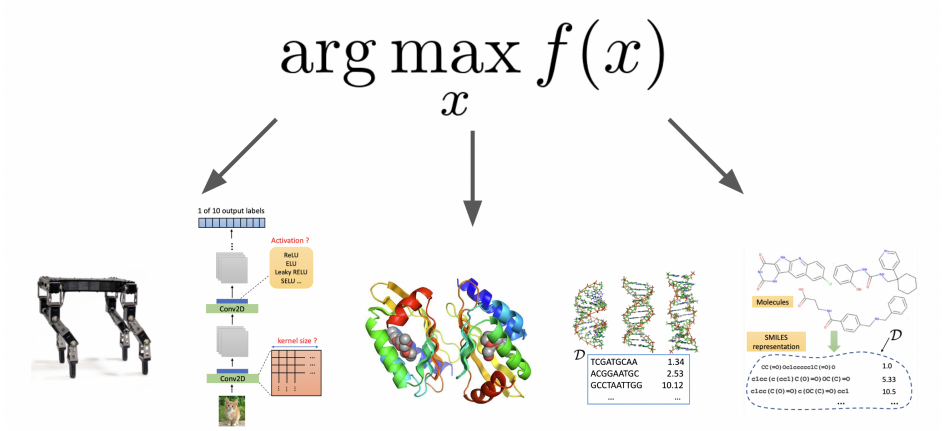


Figure 1.1: Model-based optimization problems are ubiquitous across many scientific and engineering domains. Finding good designs in robotics, biology, neural network architecture and chemical compounds are all instances of black-box model-based optimization problems.

Compared to the online variant of MBO problems which have been extensively studied over the past decades, offline MBO problems have received comparatively less attention. Traditionally, offline optimization problems are most commonly studied in the setting of offline reinforcement learning (RL), where one optimizes a policy using an offline dataset of temporal experiences, or batch Bayesian optimization, where the problem allows for a few limited iterations of online data collections. Due to these differences in problem setting, existing offline RL or batch Bayesian optimization methods cannot be directly applied to offline MBO, and only a small number of recent works have been proposed specifically for offline MBO [12, 66, 29, 31]. Even with only a few existing offline MBO methods, it is hard to compare and track progress, as methods are typically proposed and evaluated on different tasks with distinct evaluation protocols. To the best of our knowledge, there is no commonly adopted benchmark for offline MBO. In the first part of this thesis, we will address the evaluation problem of offline MBO method, where we introduce Design-Bench, a suite of benchmark tasks for offline MBO with a standard evaluation protocol. In proposing Design-Bench, we first identify a set of core challenges in real-world offline MBO problems, and then carefully choose a set of realistic tasks across a wide variety of domains, from synthetic biology to material science, to represent these core challenges in the benchmark. We then evaluate existing methods on our benchmark and report the findings.

Most prior methods of offline MBO often involve learning a objective model to predict the ground truth objective value and a generative or density model to capture the dataset distribution [12, 66, 29, 31]. While the objective model can be easy to learn via simple supervised learning, the generative or density model is often difficult to train, since good generative models are often strongly coupled with the data modality and requires careful tuning to work reliably. In the second part of this thesis, we introduce conservative objective models (COMs), a simple but effective method for offline MBO

that only requires learning a objective model. By leveraging recent advances of value conservatism [67], we augment the objective model with a conservatism loss to prevent over-estimation on out-of-distribution examples, removing the need to capture the dataset distribution explicitly. We show that COMs is easy to implement and stable to train, and obtains better performance over prior methods reliably across many tasks in Design-Bench.

In the last part of this thesis, we apply variants of our COMs model in two applications and demonstrate that our algorithm can successfully tackle real-world offline MBO problems. We first apply a variant of COMs in computational chemistry to tackle the crystal structure prediction problem, and show that our method can reliably predict the minimal energy structure of chemical compounds with little computation cost. In addition to computation chemistry, we also apply COMs to synthetic biology, where we demonstrate the COMs can help biologists design good promoter DNA sequences.

The structure of this thesis is organized as follows:

- In Chapter 2, we formally define the offline MBO problem and identify its core challenges. We present Design-Bench, a suite of realistic benchmark tasks and a standardized evaluation protocol created to benchmark offline MBO methods.
- In Chapter 3, we present conservative objective models (COMs), a simple but effective method for offline MBO that does not require training a generative or explicit density model of the dataset. We show that COMs outperforms prior methods on Design-Bench reliably.
- In Chapter 4, we present a real-world application of offline MBO in computational chemistry. We show that latent conservative objective models (LCOMs), a variant of the COMs method, can reliably solve the crystal structure prediction problem with significantly less computation cost compared to prior method.
- Chapter 5, we present another real-world application of offline MBO in synthetic biology. We show that COMs can be adapter to design differentially expressive promoter DNA sequences purely from a large batch of offline data.

We conclude this thesis in Chapter 6 by discussing some of the lessons we learned from these investigations and promising new directions of research and applications in offline MBO.

Chapter 2

Design-Bench: Benchmarks for Data-Driven Offline Model-Based Optimization

2.1 Introduction

Automatically synthesizing designs that maximize a desired objective function is one of the most important challenges in scientific and engineering disciplines. From protein design in molecular biology [102] to superconducting material discovery in physics [48], researchers have made significant progress in applying machine learning to optimization problems over structured design spaces.

Commonly, the exact form of the objective function is unknown, and the objective value for a novel design can only be found by either running computer simulations or real world experiments. This process of optimizing an unknown function by only observing samples from this function is known as black-box optimization, and is typically solved in an **online** iterative manner, where in each iteration the solver proposes new designs and queries the objective function for feedback in order to inform better design proposals at the next iteration [126]. In many domains however, the objective function is prohibitively expensive to evaluate because it requires manually conducting experiments in the real world. In this setting, one cannot query the true objective function, and cannot receive feedback on design proposals. Instead, a collection of past records of designs and corresponding objective values might be available, and the optimization method must instead leverage existing data to synthesize the most optimal designs. This is called **offline model-based optimization** (offline MBO).

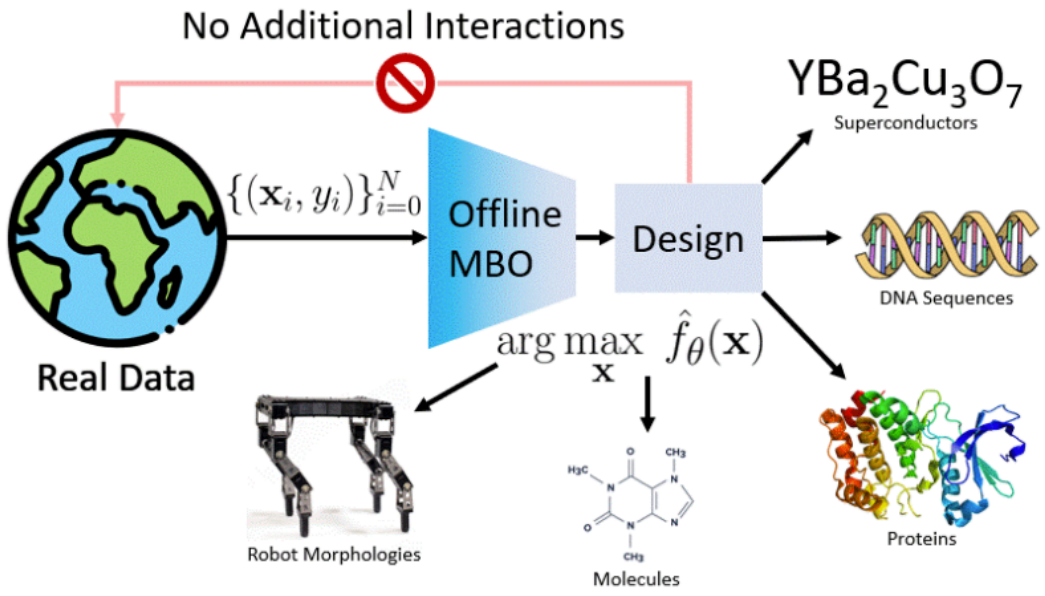


Figure 2.1: Offline model-based optimization (MBO) requires generating designs $\mathbf{x}$ that optimize a black-box objective function $f(\mathbf{x})$ using a given static dataset of designs, without any active queries to the ground truth function.

Although online black-box optimization has been studied extensively, offline MBO has received comparatively less attention, and only a small number of recent works study offline MBO in the setting with high-dimensional design spaces [12, 66, 29, 31, 116]. This is partly because online techniques cannot be directly applied in settings where offline MBO is used, especially in high-dimensional settings. Online techniques, such as Bayesian optimization [105], often require iterative feedback via queries to the objective function. Such online optimizers exhibit optimistic behavior: they rely on active queries at completely unseen designs irrespective of whether such a design is good or not. When access to these queries is removed, certain considerations change: optimism is no longer desirable and distribution shift becomes a major challenge [66].

Even with only a few existing offline MBO methods, it is hard to compare and track progress, as methods are typically proposed and evaluated on different tasks with distinct evaluation protocols. To the best of our knowledge, there is no commonly adopted benchmark for offline MBO. To address, we introduce a suite of tasks for offline MBO with a standardized evaluation protocol. We include a diverse set of tasks that span a wide range of application domains—from synthetic biology to robotics–that aims at representing the core challenges in real-world offline MBO. While the tasks are not intended to directly enable solving the corresponding real-world problems, which would require a lot of machinery in real hardware setup (e.g., a real robot or access to a wetlab for molecule design), they are intended to provide algorithm designers with a representative sampling of challenges that reflect the difficulties with real-world MBO. That is to say, the tasks are not intended to be *real*, but are intended to be

realistically challenging. Further, the diversity of the tasks measures how they generalize across multiple domains and verifies they are not specialized to a single task. Our benchmark incorporates a variety of challenging factors, such as high dimensionality and sensitive discontinuous objective functions, which help identify the strengths and weaknesses of MBO methods. Along with this benchmark suite, we present reference implementations of a number of existing offline MBO and baseline optimization methods. We systematically evaluate them on all of the proposed benchmark tasks and report results. We hope that our work can provide insight into the progress of offline MBO methods and serve as a meaningful metric to galvanize research in this area.

2.2 Offline Model-Based Optimization (Offline MBO) Problem Statement

In online model-based optimization, the goal is to optimize a (possibly stochastic) black-box objective function $f(\mathbf{x})$ with respect to its input. The objective can be written as $\arg\max_{\mathbf{x}} f(\mathbf{x})$. Methods for online MBO typically optimize the objective iteratively, proposing design $\mathbf{x}_k$ at the kth iteration and query the objective function to obtain $f(\mathbf{x}_k)$. Unlike its online counterpart, access to the true objective f is not available in offline MBO. Instead, the algorithm $\mathfrak{A}$ is provided access to a static dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ of designs $\mathbf{x}_i$ and a corresponding measurement of the objective value y_i. The algorithm consumes this dataset and produces an optimized candidate design $\mathbf{x}^*$ which is evaluated against the true objective function. This paradigm is illustrated in Figure 2.1. Abstractly, the objective for offline MBO is:

$$\arg\max_{\mathfrak{A}} f(\mathbf{x}^*) \text{ where } \mathbf{x}^* = \mathfrak{A}(\mathcal{D}). \tag{2.1}$$

In practice, producing a single optimal design entirely from offline data is very difficult, so offline MBO methods are more commonly evaluated [66] in terms of "P percentile of top K" performance, where the algorithm produces K candidates and the P percentile objective value determines final performance. Next we discuss two important aspects pertaining to offline MBO, namely, why offline MBO algorithms can improve beyond the best design observed in the offline dataset despite no active queries, and the associated challenges with devising offline model-based optimization algorithms.

Would offline MBO even produce designs better than the best observed design in the dataset?
A natural question to ask is whether it is even reasonable to expect offline MBO algorithms to improve over the performance of the best design seen in the dataset. As we will show in our benchmark results, many of the tasks that we propose do already admit solutions from existing algorithms that exceed the performance of the best sample in the dataset. To provide some intuition for how this can be possible, consider a simple example of offline MBO problems, where the objective function $f(\mathbf{x})$ can be

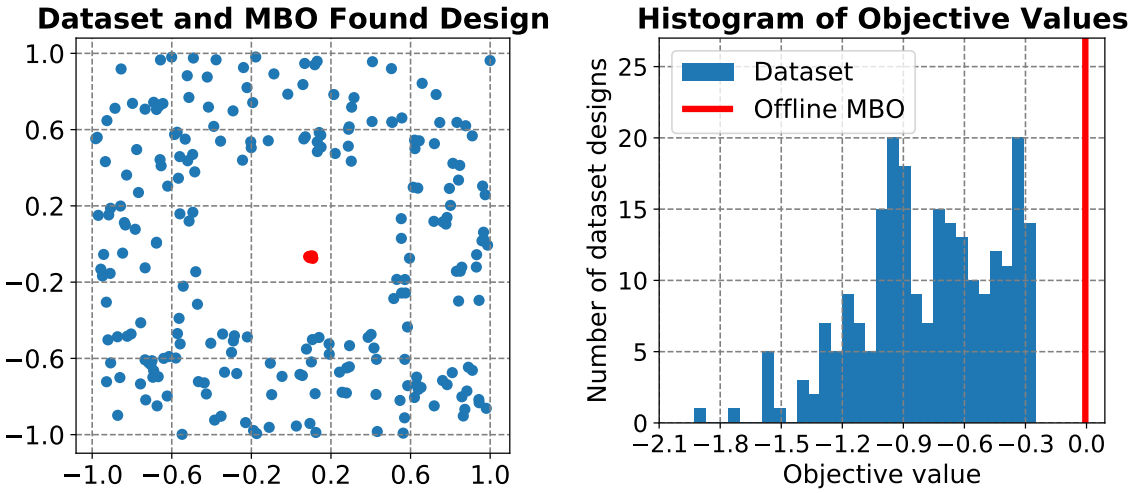


Figure 2.2: Offline MBO finds designs better than the best in the observed dataset by exploiting compositional structure of the objective function. **Left:** datapoints in a toy quadratic function MBO task over 2D space with optimum at $(0.0, 0.0)$ in blue, MBO found design in red. **Right:** Objective value for optimal design is much higher than that observed in the dataset.

represented as a sum of functions of independent partitions of the design variables, i.e., $f(\mathbf{x}) = f_1(\mathbf{x}[1]) + f_2(\mathbf{x}[2]) + \cdots + f_N(\mathbf{x}[N]))$, where $\mathbf{x}[1], \cdots, \mathbf{x}[N]$ denotes disjoint subsets of design variables $\mathbf{x}$. The dataset of the offline MBO problem contains optimal design variable for each partition, but not the combination. If an offline MBO algorithm can identify the compositional structure of independent partitions, it would be able to combine the optimal design variable for each partition together to form the overall optimal design and therefore improving the performance over the best design in the dataset. To better demonstrate this idea, we created a toy problem in two dimensions, where the objective function is simply $f(x, y) = -x^2 - y^2$. We then run a naïve gradient ascent algorithm, as we will describe later in this chapter. In Figure 2.2, we can clearly see that our offline MBO algorithm is able to learn to combine the best x and y and produce designs significantly better than the best sample in the dataset. Such a condition appears in a number of scenarios in practice e.g., in reinforcement learning (RL), where the Markov structure provides a natural decomposition satisfying this composition criterion [32] and effective offline RL algorithms are known to exploit this structure [32] or in protein design, where objective such as fluorescence naturally decompose into functions of neighboring amino acids [12].

What makes offline MBO especially challenging?

The offline nature of the problem prevents the algorithm $\mathfrak{A}$ from querying the ground truth objective f with its proposed design candidates, and this makes the offline MBO problem much more difficult than the online design optimization problem. One naïve approach to tackle this problem is to learn a model of the objective function using the dataset, which we can denote $\hat{f}(\mathbf{x})$, and then convert this offline MBO problem into a regular online MBO problem by treating the learned objective model as the true objective. However, this generally does not work: optimizing the design $\mathbf{x}$ with

Dataset Name	Size	Dimensions	Categories	Type	Oracle
TF Bind 8	32898	8	4	Discrete	Exact
TF Bind 10	50000	10	4	Discrete	Exact
NAS	1771	64	5	Discrete	Exact
ChEMBL	1093	31	591	Discrete	Random Forest
Superconductor	21263	86	N/A	Continuous	Random Forest
Ant Morphology	25009	60	N/A	Continuous	Exact
D'Kitty Morphology	25009	56	N/A	Continuous	Exact
Hopper Controller	3200	5126	N/A	Continuous	Exact

Table 2.1: Overview of the tasks in our benchmark suite. Design-Bench includes a variety of tasks from different domains, including several from prior work, and multiple new tasks, with both discrete and continuous design spaces, making it suitable for benchmarking offline MBO methods. In addition to the provided tasks, we explore several from prior work in Appendix A.4 that we chose not to include in the final benchmark.

respect to a learned proxy $\hat{f}(\mathbf{x})$ will produce *out-of-distribution* designs that "fool" $\hat{f}(\mathbf{x})$ into outputting a high value, analogously to adversarial examples. Indeed, it is well known that optimizing naïvely with respect to model inputs to obtain a desired output will usually simply "fool" the model [66]. A naïve strategy to address this out-of-distribution issue is to constrain the design to stay close to the data, but this is also problematic, since in order to produce a design that is better than the best training point, it is usually necessary to deviate from the training data at least somewhat. In almost all practical MBO problems, such as optimization over drug molecules or robot morphologies as we discuss in section 2.5, designs with the highest objective values typically lie on the tail of the dataset distribution and we may not find them by staying extremely close to the data distribution. This conflict between the need to remain close to the data to avoid out-of-distribution inputs and the need to deviate from the data to produce better designs is one of the core challenges of offline MBO. This challenge is often exacerbated in real-world settings by the high dimensionality of the design space and the sparsity of the available data, as we will show in our benchmark. A good offline MBO method needs to carefully balance these two sides, producing optimized designs that are good, but not too far from the data distribution.

2.3 Related Work

Prior work has extensively focused on online or active MBO, which requries active querying on the ground truth function, including algorithms using Bayesian optimization and their scalable variants [72, 105, 107, 100, 85], direct search [65], genetic or evolutionary algorithms [125, 81, 135], the cross-entropy method [93], simulated annealing [120], etc. These methods may not be well suited for real-world problems where the ground truth function is expensive to evaluate and therefore prohibitive for active querying. Offline MBO utilizes an already existing database of designs and objective values, which

might be obtained from previously conducted experiments. This presents an attractive algorithmic paradigm towards approaching such scenarios. Since offline MBO prohibits any ability to query the true objective with new designs, it presents different challenges from those typically studied in online MBO problem, as we discuss in Section 2.5. These new challenges in turn require new benchmarks, motivating our work.

The most important components for a good offline MBO benchmark are datasets that capture the challenges of real-world problems. Fortunately, researchers working on a wide variety of scientific fields have already collected many datasets of designs which we can use for training offline MBO algorithms. ChEMBL [36] provides a dataset for drug discovery, where molecule activities are measured against a target assay. Hamidieh [48] analyze the critical temperatures for superconductors and provide a dataset to search for room-temperature superconductors with potential in the construction of quantum computers. Some of these datasets have already been employed in the study of offline MBO methods [66, 12, 29]. However, these studies all use different sets of tasks and their evaluation protocols are highly domain-specific, making it difficult to compare across methods. In our benchmark, we incorporate modified variants of some of these datasets along with our own tasks, and provide a standardized evaluation protocol.

Recently, several methods have been proposed for specifically addressing the offline MBO problem. These methods [66, 12, 29] typically learn models of the objective function and optionally, a generative model [61, 41, 76] of the design manifold and use them for optimization. We discuss these methods in detail in Section 2.6 and benchmark their performance in Section 4.6.

2.4 Design-Bench Benchmark Tasks

In this section, we describe the set of tasks included in our benchmark. An overview of the tasks is provided in Table 2.1. Each task in our benchmark suite comes with a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$, along with a ground-truth oracle objective function f that can be used for evaluation. An offline MBO algorithm should not query the ground-truth oracle function during training, even for hyperparameter tuning. We first discuss the nature of oracles used in Design-Bench.

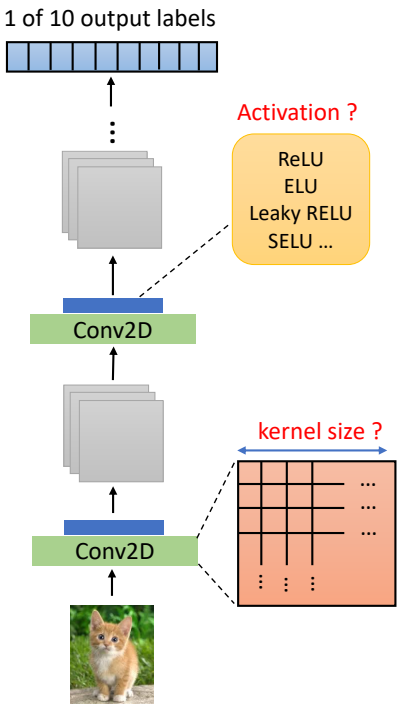
Expert model as oracle function. While in some of the tasks in our benchmark, such as tasks pertaining to robotics (D'Kitty Morphology, and Ant Morphology), the oracle functions are evaluated by running computer simulations to obtain the true objective values, in the other tasks, the true objective values can only be obtained by conducting expensive physical experiments. While the eventual aim of offline MBO is to make it possible to optimize designs in precisely such settings, requiring real physical experiments for evaluation makes the design and benchmarking of new algorithms difficult and time consuming. Therefore, to facilitate benchmarking, we follow the

evaluation methodology in prior work [12, 29] and use models built by domain experts as our ground-truth oracle functions. Note, however, that the training data provided for offline MBO is still real data – the domain expert model is used *only* to evaluate the result for benchmarking purposes. In many cases, these expert models are *also* learned, but with representations that are hand-designed, with built-in domain-specific inductive biases. The ground-truth oracle models are also trained on much more data than is made available for solving the offline MBO problem, which increases the likelihood that this expert model can provide an accurate evaluation of solutions found by offline MBO, even if they lie outside the training distribution. While this approach to evaluation diminishes the realism of our benchmark since these proxy "true functions" may not always be accurate, we believe that this trade off is worthwhile to make benchmarking practical. The main purpose of our benchmark is to facilitate the evaluation and development of offline MBO algorithms, and we believe that it is important to include tasks in domains where the true objective values can only be obtained via physical experiments, which make up a large portion of the real-world MBO problems.

We now provide a detailed description of the tasks in our benchmark. A description of the data collection strategy and pre-processing can be found in Appendix A.1.

NAS: neural architecture search on CIFAR10.

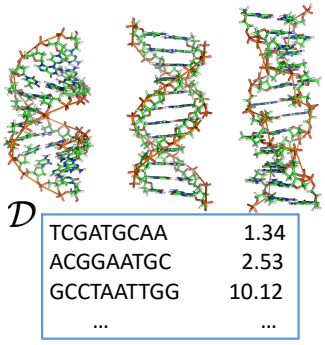
The goal of this task is to search for a good neural network architecture [137] to optimize the test accuracy on the CIFAR10 [53] dataset. The model is a 32-layer convolutional neural network with residual connections, and the task requires searching over the kernel sizes and activation function types for each of the 32 layers. Given the small image size of CIFAR10, we choose the list of possible kernel sizes to be $\{2, 3, 4, 5, 6\}$. The possible choices of activation functions are ReLU, ELU, leaky ReLU, SELU [63] and SiLU [25]. The combination of kernel sizes and activation functions give us a 64 dimensional discrete space with 5 categories per dimension. The dataset is collected by randomly sampling architectures in the search space. We evaluate the design by training the produced architecture on the training CIFAR10 dataset for 20 epochs and evaluating the accuracy on the test set.



Hopper Controller: robot neural network controller optimization. The goal in this task is to optimize the weights of a neural network policy so as to maximize the expected discounted return on the Hopper-v2 locomotion task in OpenAI Gym [10]. While this might appear similar to reinforcement learning (RL), our formulation is

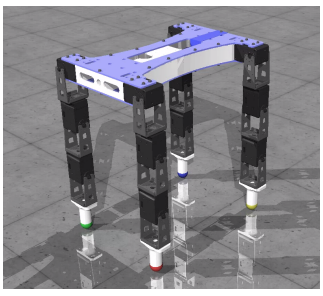
distinct: unlike RL, we don't have access to any form of trajectory data in the dataset. Instead, our dataset only comprises of neural network controller weights and the corresponding return values, which invalidates the applicability of conventional RL methods. We evaluate the true objective value of any design by running 1000 steps of simulation in the MuJoCo simulator conventionally ussed with this environment. The design space of this task is high-dimensional with 5126 continuous variables corresponding to the flattened weights of a neural network controller. The dataset is collected by training a PPO [99] and recording the agent's weights every 10,000 samples.

TF Bind 8 and TF Bind 10: DNA sequence optimization. The goal of TF Bind 8 and TF Bind 10 is to find the length-8 DNA sequence with maximum binding affinity with a particular transcription factor (`SIX6_REF_R1` by default). The ground truth binding affinities for all 65,792 and 1,048,576 designs for the two tasks are available [8]. The design space consists of sequences of one of four categorical variables, one for each nucleotide. For TF Bind 8, we sample 32898 of all the sequences, and for TF Bind 10 we sample 50000 sequences to form the training set.



Superconductor: critical temperature maximization. The Superconductor task is taken from the domain of materials science, where the goal is to design the chemical formula for a superconducting material that has a high critical temperature. We adapt a real-world dataset proposed by [48]. The dataset contains 21263 superconductors annotated with critical temperatures. Prior work has employed this dataset for the study of offline MBO methods [29], and we follow their convention using a random forest regression model, detailed in [48], for our oracle. The model achieves a final Spearman's rank-correlation coefficient with a held-out validation set of 0.9210. The design space for Superconductor is a vector with 86 real-valued components representing the mixture of elements by number of atoms in the chemical formula of each superconductor.

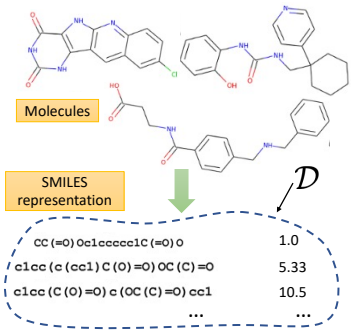
Ant and D'Kitty Morphology: robot morphology optimization.



The goal is to optimize the morphological structure of two simulated robots: Ant from OpenAI Gym [10] and D'Kitty from ROBEL [2]. For Ant Morphology, the we need to optimize the morphology of a quadruped robot to run as fast as possible. For D'Kitty Morphology, the goal is to optimize the morphology of D'Kitty robot (shown on the right) to navigate the robot to a fixed location. Thus the goal is to find robot morphologies optimal for the given tasks. In order to control the robot with the generated morphology, we use a controller that

has been optimized for the given morphology with the Soft Actor Critic algorithm [46]. The morphology parameters of both robots include size, orientation, and location of the limbs, giving us 60 continuous values in total for Ant and 56 for D'Kitty. To evaluate a given design, we run robotic simulation in the MuJoCo [113] simulator for 100 time steps, averaging 16 independent trials giving us reliable but cheap to compute estimates.

ChEMBL: molecule activity maximization for drug discovery. The ChEMBL task in Design bench is derived from a large-scale drug property database from which the task name is derived [36]. This database consists of pairs of molecules and assays tested for a particular chemical properties. We choose the assay whose ChEMBL id is `CHEMBL3885882` and measure its `MCHC` value. The goal of the resulting optimization problem is to design a molecule that, when paired with assay `CHEMBL3885882`, achieves a high `MCHC` value. The training set is restricted to molecules whose SMILES [124] encoding has fewer than 30 tokens. This results in a training set with 1093 samples, and a design space of length 31 sequences of categorical variables that take one of 591 values.



2.5 Task Properties, Challenges and Considerations

The primary goal of our benchmark is to provide a general test bench for developing, evaluating, and comparing algorithms for offline MBO. While in principle any online active black-box optimization problem can be turned into an offline MBO problem by collecting a dataset of designs and corresponding objective measurements, it is important to pick a subset of tasks that represent the challenges of real-world problems in order to convincingly evaluate algorithms and obtain insights about algorithm behavior. Therefore, several factors must be considered when choosing the tasks, which we discuss next.

Diversity and realistically challenging. First of all, the tasks need to be diverse and realistically challenging in order to prevent offline MBO algorithms from overfitting to a particular problem domain and to expect that methods performing well on this benchmark suite would also perform well on real-world offline MBO problems. Design-Bench consists of tasks that are diverse in many respects. It includes both tasks with *discrete* and with *continuous* design spaces. Continuous design spaces, equipped with metric space and ordering structures, could make the problem easier to solve than discrete design spaces. However, discrete design spaces are finite and therefore might enjoy better dataset coverage than some continuous tasks. While our tasks are not intended to directly solve real-world problems (e.g., we don't actually expect the best

robot morphology in our benchmark to actually correspond to the best possible real robot morphology due to a variety of factors including limitations of the simulator), they are intended to provide designers with a representative sampling of challenges that reflect the kinds of difficulties they would face with real-world datasets, making them realistically challenging.

High-dimensional design spaces. In many real-world offline MBO problems, such as drug discovery [36], the design space is *high-dimensional* and good designs sparsely lie on a *thin manifold* in this high-dimensional space. This poses a challenge for many MBO methods: to be effective on such problem domains, MBO methods need to capture the thin manifold to be able to produce good designs. Prior work [66] has noted that this can be very hard in practice. In our benchmark, we include a task derived from ChEMBL with up 31 dimensions and 591 categories per dimension to capture this challenge. To intuitively understand this challenge, we performed a study on some tasks in Figure 2.3, where we sampled 3200 designs uniformly at random from the design space and plotted a histogram of the objective values against those in the dataset we provide, which only consists of valid designs. Observe the discrepancy in objective values, where randomly sampled designs generally attain objective values lower than the best dataset sample. This suggests that performant designs only lie on a thin manifold in the design space and therefore we are very unlikely to hit a performant design by uninformed random sampling.

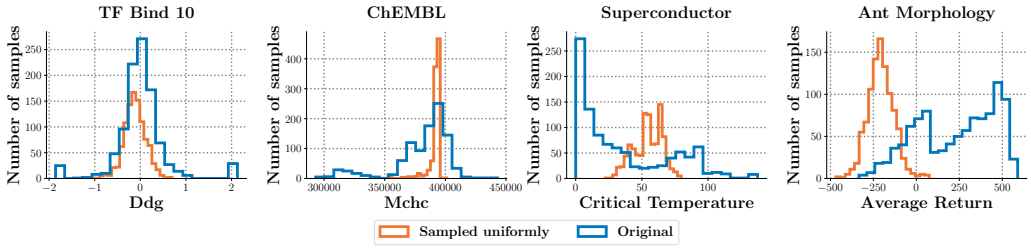


Figure 2.3: Histogram (frequency distribution) of objective values in the dataset compared to a uniform re-sampling of the dataset from the design space. In every case, re-sampling skews the distribution of values to the left, suggesting that there exists a thin manifold of valid designs in the high-dimensional design space, and most of the volume in this space is occupied by low-scoring designs. The distribution of objective values in the dataset are often heavy-tailed, for instance, in the case of ChEMBL and Superconductor.

Highly sensitive objective function. Another important challenge that should be taken into consideration is the *high sensitivity* of objective functions, where closeness of two designs in design space need not correspond to closeness in their objective values, which may differ drastically. This challenge is naturally present in practical problems like drug discovery [37], where the change of a single atom could significantly alter the property of the molecule. The DKitty Morphology and Ant Morphology tasks in our benchmark suite are also particularly challenging in this respect. To visualize the high sensitivity of the objective function, we plot a one dimensional slice of the objective function around a single sample in our dataset in Figure 2.4. Observe that

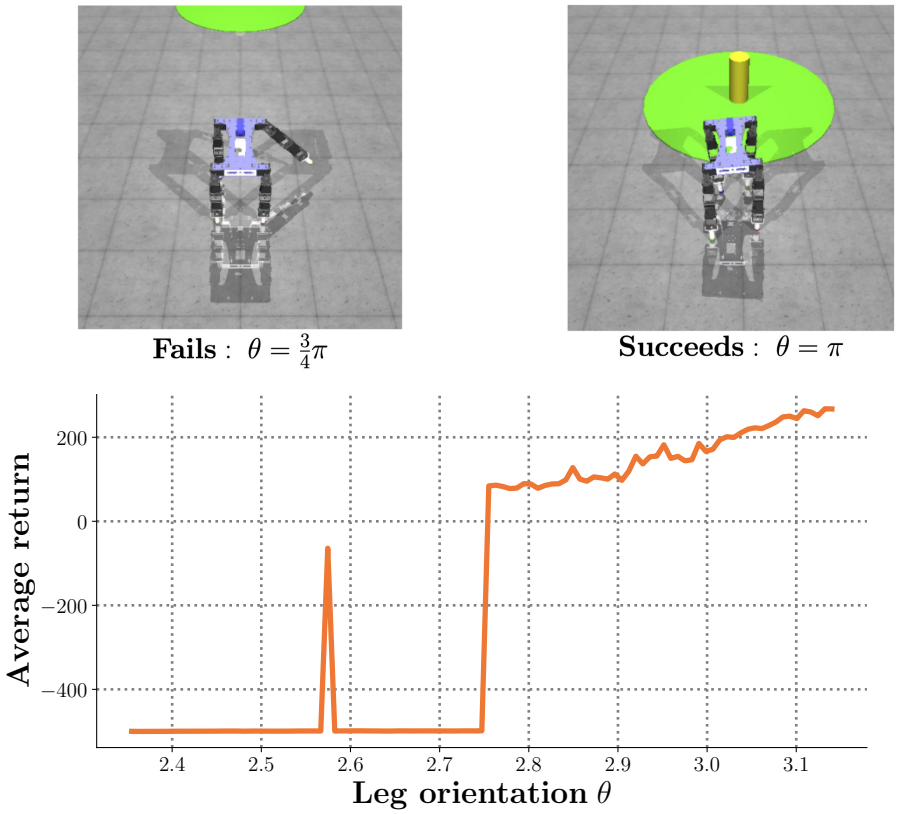


Figure 2.4: Highly sensitive landscape of the ground truth objective function in DKittyMorphology. A small change in a single dimension of the design space, for instance changing the orientation θ (x-axis) of the base of the robot's front right leg, critically impacts the performance value (y-axis). The robot's design on the left is the original D'Kitty design and is held constant while varying θ uniformly from $\frac{3}{4}\pi$ to π.

with other variables kept constant, slightly altering one variable can significantly reduce the objective value, making it hard for offline MBO methods to produce the optimal design.

Heavy-tailed data distributions. Finally, another challenging property for offline MBO methods is the shape of the data distribution. Learning algorithms are likely to exhibit poor learning behavior when the distribution of objective values in the dataset is heavy-tailed. This challenge is often present in black-box optimization [20] and can hurt the performance of MBO algorithms that use a generative model as well as those that use a learned model of the objective function. As shown in Figure 2.3 tasks in our benchmark exhibit this heavy-tailed structure.

2.6 Algorithm Implementations

To provide a baseline for comparisons in future work, we benchmark a number of recently proposed offline MBO algorithms on each of our tasks. Since some of our tasks have a high input dimensionality, we chose prior methods that can handle *both* the case of offline training data (i.e., no active interaction) and high-dimensional inputs. Thus, we include MINs [66], CbAS [12], autofocusing CbAS [29] and REINFORCE/CMA-ES [128] in our comparisons, along with a baseline naïve "gradient ascent" method that approximates the true function $f(\mathbf{x})$ with a deep neural network and then performs gradient ascent on the output of this model. In this section, we briefly discuss these algorithms, before performing a comparative evaluation in the next section. Our implementation of these algorithms are open sourced and can be found at github.com/rail-berkeley/design-baselines.

Gradient ascent (Grad). This is a simple baseline that learns a model of the objective function, $\hat{f}(\mathbf{x})$, and optimizes $\mathbf{x}$ against this learned model via gradient ascent. Formally, the optimal solution $\mathbf{x}^*$ generated by this method can be computed as a fixed point of the following update: $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \alpha \nabla_{\mathbf{x}} \hat{f}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t}$. In practice we perform $T = 200$ gradient steps, and report $\mathbf{x}_T$ as the final solution. Such methods are susceptible to producing invalid solutions, since the learned model does not capture the manifold of valid-designs and hence cannot constrain the resulting $\mathbf{x}_T$ to be on the manifold. We additionally evaluate a variant (**Grad. Min**) optimizing over the minimum prediction of $N = 5$ learned objective functions in an ensemble of learned objective functions and (**Grad. Mean**) that optimizes the mean ensemble prediction. We discuss additional tricks (e.g., normalization of inputs and outputs) that we found beneficial with this baseline in Appendix A.5.

Covariance matrix adaptation (CMA-ES). CMA-ES Hansen [49] is a simple optimization algorithm that maintains a belief distribution over the optimal design, and gradually refines this distribution by adapting the covariance matrix using feedback from a (learned) objective function, $\hat{f}(\mathbf{x})$. Formally, let $\mathbf{x}_t \sim \mathcal{N}(\mu_t, \Sigma_t)$ be the samples obtained from the distribution at an iteration t, then CMA-ES computes the value of learned $\hat{f}(\mathbf{x}_t)$ on samples $\mathbf{x}_t$, and fits Σ_{t+1} to the highest scoring fraction of these samples and repeats this multiple times. The learned $\hat{f}(\mathbf{x})$ is trained via supervised regression.

REINFORCE [128]. We also evaluated a method that optimizes a learned objective function, $\hat{f}(\mathbf{x})$, using the REINFORCE-style policy-gradient estimator. REINFORCE is capable of handling non-smooth and highly stochastic objectives, making it an effective choice. This method parameterizes a distribution $\pi_\theta(\mathbf{x})$ over the design space and then updates the parameters θ of this distribution towards the design that maximizes $\hat{f}(\mathbf{x})$, using the gradient, $\mathbb{E}_{\mathbf{x} \sim \pi_\theta(\mathbf{x})}[\nabla_\theta \log \pi_\theta(\mathbf{x}) \cdot \hat{f}(\mathbf{x})]$. We train an ensemble of $\hat{f}(\mathbf{x})$ models and pick the subset of models that satisfy a validation loss threshold τ. This threshold is task-specific; for example, $\tau \leq 0.25$ is sufficient for Superconductor-v0.

Conditioning by adaptive sampling (CbAS) [12]. CbAS learns a density model in the space of design inputs, $p_0(\mathbf{x})$ that approximates the data distribution and gradually adapts it towards the optimized solution $\mathbf{x}^*$. In a particular iteration t, CbAS alternates between **(1)** training a variational auto-encoder (VAE) [61] on a set of samples generated from the previous model $\mathcal{D}_t = \{\mathbf{x}_i\}_{i=1}^m; \mathbf{x}_i \sim p_{t-1}(\cdot)$ using a weighted version of the standard ELBO objective biased towards *estimated* better designs and **(2)** generating new design samples from the autoencoder to serve as $\mathcal{D}_{t+1} = \{\mathbf{x}_i | \mathbf{x}_i \sim p_t(\cdot)\}$. In order to estimate the objective values for designs sampled from the learned density model $p_t(\mathbf{x})$, CbAS utilizes separately trained models of the objective function, $\hat{f}(\mathbf{x})$ trained via supervised regression. This training process, at a given iteration t, is:

$$p_{t+1}(\mathbf{x}) := \arg\min_p \frac{1}{m} \sum_{i=1}^{m} \frac{p_0(\mathbf{x}_i)}{p_t(\mathbf{x}_i)} P(\hat{f}(\mathbf{x}_i) \geq \tau) \log p_t(\mathbf{x}_i)$$
$$\text{where} \quad \{\mathbf{x}_i\}_{i=1}^m \sim p_t(\cdot). \tag{2.2}$$

Autofocused CbAS (Auto. CbAS) [29]. Since CbAS uses a learned model of the objective function $\hat{f}(\mathbf{x})$ to iteratively adapt the generative model $p(\mathbf{x})$ towards the optimized design, the function$\hat{f}(\mathbf{x})$ will inevitably be required to make predictions on shifting design distributions $p_t(\mathbf{x})$. Hence, any inaccuracy in these values can adversely affect the optimization procedure. Autofocused CbAS aims to correct for this shift by re-training $\hat{f}(\mathbf{x})$ (now denoted $\hat{f}_t(\mathbf{x})$) under the design distribution given by the current model, $p_t(\mathbf{x})$ via importance sampling, which is then fed into CbAS.

$$\hat{f}_{t+1} := \arg\min_{\hat{f}} \quad \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \frac{p_t(\mathbf{x}_i)}{p_0(\mathbf{x}_i)} \cdot \left(\hat{f}(\mathbf{x}_i) - y_i\right)^2,$$

Model inversion networks (MINs) [66]. MINs learn an inverse map from the objective value to a design, $\hat{f}^{-1} : \mathcal{Y} \to \mathcal{X}$ by using objective-conditioned inverse maps, search for optimal y values during optimization and finally query the learned inverse map to produce the corresponding optimal design. MIN minimizes a divergence measure $\mathcal{L}_p(\mathcal{D}) := \mathbb{E}_{y \sim p_{\mathcal{D}}(y)} \left[D(p_{\mathcal{D}}(\mathbf{x}|y), \hat{f}^{-1}(\mathbf{x}|y)) \right]$ to train such an inverse map. During optimization, MINs obtains the optimized design by sampling from the inverse map conditioned on the optimal y-value.

Bayesian optimization (BO-qEI). We perform offline Bayesian optimization to maximize the value of a learned objective function $\hat{f}(\mathbf{x})$ by fitting a Gaussian Process, proposing candidate solutions, and labeling these candidates using $\hat{f}(\mathbf{x})$. To improve efficiency, we use the quasi-Expected-Improvement acquisition function [129] based on the BoTorch framework [7].

Conservative Objective Models (COMs) [116]. COMs utilizes a single learned model of the objective function $\hat{f}(\mathbf{x}_i)$ for offline model-based optimization. COMs learns a conservative model of the objective function using an augmented regression

objective that penalizes overestimation of the performance on off-manifold designs $\mathbf{x}$. Solutions are obtained by initializing $\mathbf{x}_0$ to a design from an observed training set $\mathcal{D}$, and performing T steps of gradient ascent $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \nabla_{\mathbf{x}} \alpha \hat{f}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t}$ on the conservative objective model's predictions with respect to the design $\mathbf{x}$.

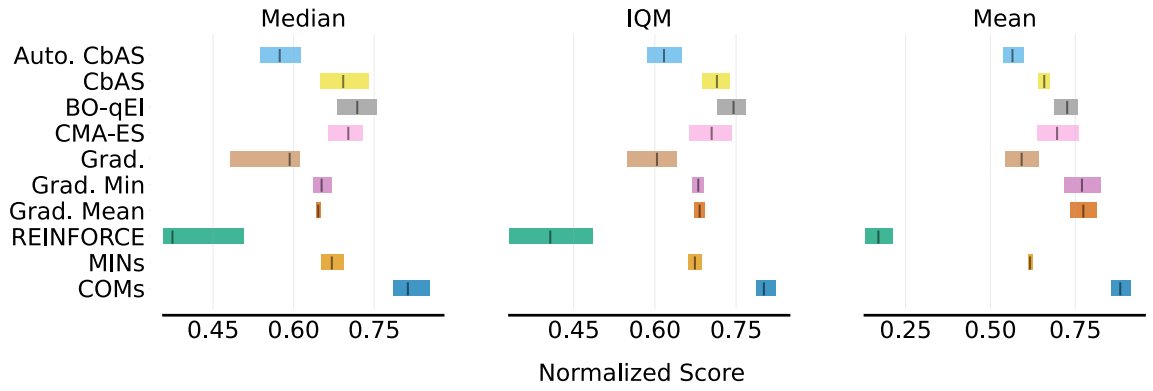


Figure 2.5: Median, IQM and mean [1] aggregated 100th percentile normalized scores (with 95% Stratified Bootstrap CIs) for the tasks in Design-Bench.

	TF Bind 8	TF Bind 10	ChEMBL	NAS	Superconductor	Ant Morph.	DKitty Morph.	Hopper
$\mathcal{D}$ (best)	0.439	0.467	0.605	0.436	0.400	0.565	0.884	1.0
Auto. CbAS	0.910 ± 0.044	0.630 ± 0.045	0.249 ± 0.305	0.506 ± 0.074	0.421 ± 0.045	0.882 ± 0.045	0.906 ± 0.006	0.137 ± 0.005
CbAS	0.927 ± 0.051	0.651 ± 0.060	0.473 ± 0.264	0.683 ± 0.079	0.503 ± 0.069	0.876 ± 0.031	0.892 ± 0.008	0.141 ± 0.012
BO-qEI	0.798 ± 0.083	0.652 ± 0.038	0.596 ± 0.226	1.079 ± 0.059	0.402 ± 0.034	0.819 ± 0.000	0.896 ± 0.000	0.550 ± 0.118
CMA-ES	0.953 ± 0.022	0.670 ± 0.023	0.085 ± 0.225	0.985 ± 0.079	0.465 ± 0.024	1.214 ± 0.732	0.724 ± 0.001	0.604 ± 0.215
Grad.	0.977 ± 0.025	0.657 ± 0.039	0.307 ± 0.308	0.433 ± 0.000	0.518 ± 0.024	0.293 ± 0.023	0.874 ± 0.022	1.035 ± 0.482
Grad. Min	0.984 ± 0.012	0.649 ± 0.032	0.653 ± 0.024	0.433 ± 0.000	0.506 ± 0.009	0.479 ± 0.064	0.889 ± 0.011	1.391 ± 0.589
Grad. Mean	0.986 ± 0.012	0.645 ± 0.018	0.652 ± 0.005	0.433 ± 0.000	0.499 ± 0.017	0.445 ± 0.080	0.892 ± 0.011	1.586 ± 0.454
REINFORCE	0.948 ± 0.028	0.663 ± 0.034	0.164 ± 0.285	-1.895 ± 0.000	0.481 ± 0.013	0.266 ± 0.032	0.562 ± 0.196	-0.020 ± 0.067
MINs	0.905 ± 0.052	0.616 ± 0.021	0.000 ± 0.000	0.717 ± 0.046	0.499 ± 0.017	0.445 ± 0.080	0.892 ± 0.011	0.424 ± 0.166
COMs	0.973 ± 0.016	0.730 ± 0.136	0.633 ± 0.000	0.459 ± 0.139	0.439 ± 0.033	0.944 ± 0.016	0.949 ± 0.015	2.056 ± 0.314

Table 2.2: 100th percentile evaluations. Results are averaged over 8 trials, and ± indicates the standard deviation of the performance. The objective value normalization procedure is described in Appendix A.3.1.

2.7 Benchmarking Prior Methods

In this section, we provide a comparison of prior algorithms discussed in Section 2.6 on our proposed tasks. For purposes of standardization, easy benchmarking, and future algorithm development, we present results for all Design-Bench tasks in Table 2.2. As discussed in Section 2.2, we allow each method to produce $K = 128$ optimized design candidates. These candidates are then evaluated with the oracle function, and we report the 100^{th} percentile performance among them averaged over 8 independent runs, following the conventions of prior works [29, 12, 66]. We also provide unnormalized and 50^{th}%ile results in Appendices A.3.3, A.3.2.

Algorithm setup and hyperparameter tuning. Since our goal is to generate high-performing solutions without *any* knowledge of the ground truth function, any

form of hyperparameter tuning on the parameters of the learned model should crucially respect this evaluation boundary and tuning must be performed completely offline, agnostic of the objective function. We provide a recommended method for tuning each algorithm described in Section 2.6 in Appendix A.6, which also serves as a set of guidelines for tuning future methods with similar components.

To briefly summarize, **for CbAS**, hyperparameter tuning amounts to tuning a VAE where samples from the prior distribution map to on-manifold designs after reconstruction. We empirically found that a β-VAE was essential for stability of CbAS and high values of $\beta > 1$ are especially important for modelling high-dimensional spaces. As a general task-agnostic principle for selecting β, we choose the smallest β such that the VAE's latent space does not collapse during importance sampling. Collapsing latent-spaces seem to coincide with diverging importance sampling, and the VAE's reconstructions collapsing to a single mode. **For MINs**, hyperparameter tuning amounts to fitting a good generative model. We observe that MINs is particularly sensitive to the scale of y_i when conditioning, which we resolve by normalizing the objective values. We implement MINs using WGAN-GP, and find that similar hyperparameters work well across domains. **For Gradient Ascent**, while prior works report poor performance for naïve gradient ascent optimization on top of learned models of the objective function, we find that by normalizing the designs $\mathbf{x}$ and objective values y to have unit normal statistics and scaling the learning rate as $\alpha \leftarrow \alpha\sqrt{d}$ where d is the dimension of the design space (discussed in Appendix A.5), a naïve gradient ascent based procedure performs reasonably well on most tasks without task-specific tuning. For discrete tasks, only the objective values are normalized, and optimization is performed over log-probabilities of designs. We then obtain optimized designs by running 200 steps of gradient ascent starting from the top scoring 128 samples in each dataset. We provide further details in Appendix A.6.

Results. The results for all tasks are provided in Table 2.2. There are several takeaways from these results. First, these results confirm that three prior offline MBO methods (MINs, CbAS, and Autofocused CbAS), are very successful at solving a wide range of offline MBO problems of varying dimensional and modality. Furthermore, perhaps somewhat surprisingly, a classical CMA-ES baseline is competitive with several highly sophisticated MBO methods in 4 out of 8 tasks (Table 2.2). This result suggests that it might be difficult for generative models to capture high-dimensional task distributions with enough precision to be used for optimization, and in a number of tasks, these components might be unnecessary. Additionally a naive gradient ascent baseline is competitive with complex approaches utilizing generative modelling on 4 of the 8 tasks. However, on the other hand, as described in Appendix A.5 and A.6.4, baseline is also sensitive to certain design choices such as input normalization schemes and the number of optimization steps T. Therefore, while not a full-fledged offline MBO method, we believe that gradient ascent has potential to form a fundamental building block for future offline MBO methods.

Finally, we remark that the performance of methods in Table 2.2 differ from the those reported by prior works. This difference stems from the standardization procedure employed in dataset generation (which we discuss in Appendix A.1).

2.8 Discussion and Conclusion

Offline model-based optimization carries the promise to convert existing databases of designs into powerful optimizers, without the need for expensive real-world experiments for actively querying the ground truth objective function. However, due to the lack of standardized benchmarks and evaluation protocols, it has been difficult to accurately track the progress of offline MBO methods. To address this problem, we introduce Design-Bench, a benchmark suite of offline MBO tasks that covers a wide variety of domains, and both continuous and discrete, low and high dimensional design spaces. We provide a comprehensive evaluation of existing methods under identical assumptions. The comparatively high efficacy of even simple baselines such as CMA-ES and naïve gradient ascent suggests the need for careful tuning and standardization of methods in this area. An interesting avenue for future work in offline MBO is to devise methods that can be used to perform model and hyperparameter selection. One promising approach to address this problem is to devise methods for offline evaluation of produced solutions. We hope that our benchmark will be adopted as the standard metric in evaluating offline MBO algorithms and provides insight in future algorithm development.

Acknowledgements

We thank members of the RAIL lab at UC Berkeley, Jennifer Listgarten and Clara Fannjiang for informative discussions and suggestions for tasks appearing in this benchmark. We thank anonymous reviewers from NeurIPS and ICLR for feedback on a previous version of this manuscript. This research is supported by Intel, Schmidt Futures, the Office of Naval Research and compute resources from Google Cloud and Microsoft Azure.

Chapter 3

Conservative Objective Models for Offline MBO

3.1 Introduction

Black-box model-based optimization (MBO) problems are ubiquitous in a wide range of domains, such as protein [12] or molecule design [37], designing controllers [9] or robot morphologies [70], optimizing neural network designs [138], and aircraft design [55]. Existing methods to solve such model-based optimization problems typically learn a proxy function to represent the unknown objective landscape based on the data, and then optimize the design against this learned objective function. In order to prevent errors in the learned proxy function from affecting optimization, these methods often critically rely on periodic *active* data collection [105] over the course of training. Active data collection can be expensive or even dangerous: evaluating a real design might involve a complex real-world procedure such as synthesizing candidate protein structures for protein optimization or building the robot for robot design optimization. While these problems can potentially be solved via computer simulation, a high fidelity simulator often requires considerable effort from experts across multiple domains to build, making it impractical for most problems. Therefore, a desirable alternative approach for a broad range of MBO problems is to develop *data-driven, offline* methods that can optimize designs by training highly general and expressive deep neural network models on data from previously conducted experiments, consisting of inputs ($\mathbf{x}$) and their corresponding objective values (y), without access to the true function or any form of active data collection [66]. In a number of these practical domains, such as protein [97] or molecule design [37], plenty of prior data already exists and can be utilized for fully offline, data-driven model-based optimization.

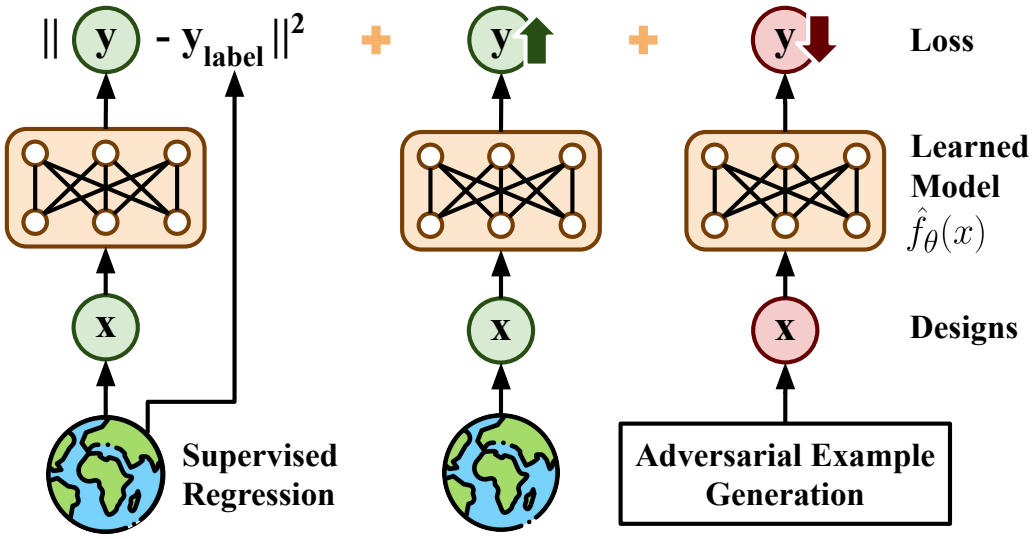


Figure 3.1: Overview of COMs. Our method trains a model of the objective function by training a neural net with supervised regression on the training data augmented two additional loss terms to obtian conservative predictions. These additional terms aim to maximize the predictions of the neural net model on the training data, and minimize the predictions on adversarially generated designs. This principle prevents the optimizer from producing bad designs with erroneously high values at unseen and poor designs.

Typical approaches for addressing MBO problems learn a model of the unknown objective function $\hat{f}$ that maps an input $\mathbf{x}$ (or a representation of the input [40]) to its objective value $\hat{f}(\mathbf{x})$ via supervised regression on the training dataset [105]. Then, these methods optimize the input against this learned model via, for instance, gradient ascent. For MBO problems where the space of valid inputs forms a narrow manifold in a high-dimensional space, any overestimation errors in the learned model will erroneously drive the optimization procedure towards out-of-distribution, invalid, and low-valued design designs that "fool" the model into producing a high values [66].

How can we prevent offline MBO methods from falling into such out-of-distribution solutions? If we can instead learn a *conservative* model of the objective function that does not overestimate the objective value on out-of-distribution inputs, optimizing against this conservative model would produce the best solutions for which we are *confident* in the value. In this chapter, we propose a method to learn such *conservative objective models* (COMs), and then optimize the design against this conservative model using a naïve gradient-ascent procedure. Analogously to adversarial training approaches in supervised learning [42], and building on recent works in offline reinforcmeent learning [69, 67], COMs first explicitly mine for out-of-distribution inputs with erroneously overestimated values and then penalize the predictions on these inputs. Theoretically, we show that this approach mitigates overestimation in the learned objective model near the manifold of the dataset. Empirically, we find that this leads to good performance across a range

of offline model-based optimization tasks.

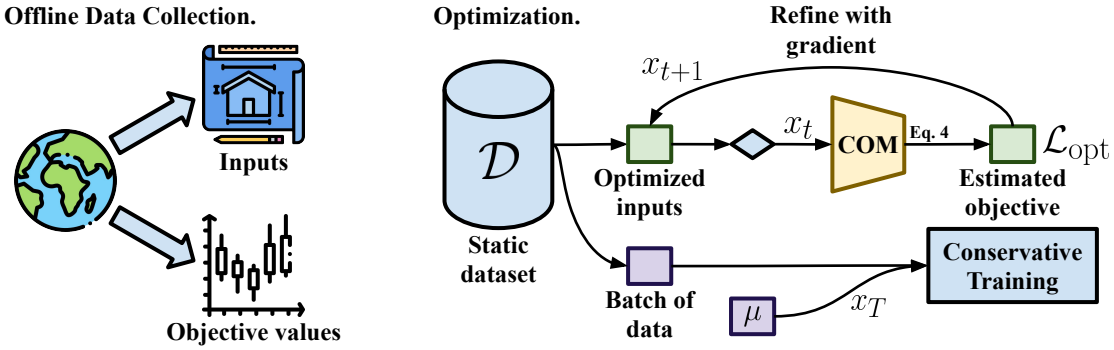


Figure 3.2: Training and optimization using COMs. The section on the left indicates that each task provides a static dataset that is collected offline without ayn MBO algorithm in-the-loop. The section on the right shows how a conservative objective model is used to produce promising optimized designs using gradient ascent, and how these designs are inputs to a conservative regularizer.

The primary contribution proposed in this chapter, COMs, is a novel approach for addressing data-driven model-based optimization problems by learning a conservative model of the unknown objective function that lower-bounds the groundtruth function on out-of-distribution inputs, and then optimizing the input against this conservative model via a simple gradient-ascent style procedure. COMs are simple to implement, utilizing a supervised learning procedure that resembles adversarial training, without the need for complex generative modeling to estimate dataset support as in prior work on model-based optimization. We theoretically analyze COMs and show that they never overestimate the values at out-of-distribution inputs close to the dataset manifold and we empirically demonstrate the efficacy of COMs on seven complex MBO tasks that span a wide range of real-world tasks including biological sequence design, neural network parameter optimization, and superconducting material design. COMs is optimal on **4/7** tasks, and outperforms the best prior method by a factor of **1.3x** in a high-dimensional setting, and by a factor of **1.16x** overall.

3.2 Preliminaries

The goal in data-driven, offline model-based optimization [66] is to find the best possible solution, $\mathbf{x}^*$, to optimization problems of the form

$$\mathbf{x}^* \leftarrow \arg\max_{\mathbf{x}} \; f(\mathbf{x}), \tag{3.1}$$

where $f(\mathbf{x})$ is an unknown (possibly stochastic) objective function. An offline MBO algorithm is provided access to a static dataset $\mathcal{D}$ of inputs and their objective values, $\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$. While a variety of MBO methods have been developed [40, 12, 66, 29], most methods for tackling MBO problems fit a parametric model to the

samples of the true objective function in $\mathcal{D}$, $\hat{f}_\theta(\mathbf{x})$, via supervised training: $\hat{f}_\theta(\mathbf{x}) \leftarrow \arg\min_\theta \sum_i (\hat{f}_\theta(\mathbf{x}_i) - y_i)^2$, and find $\mathbf{x}^*$ in Equation 3.1 by optimizing $\mathbf{x}$ against this learned model $\hat{f}_\theta(\mathbf{x})$, typically with some mechanism to additionally minimize distribution shift. One choice for optimizing $\mathbf{x}$ in Equation 3.1 is gradient descent on the learned function, as given by

$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \eta \nabla_x \hat{f}_\theta(\mathbf{x})|_{x=\mathbf{x}_k}, \quad \text{for} \;\; k \in [1, T], \qquad \mathbf{x}^\star = \mathbf{x}_T. \tag{3.2}$$

The fixed point of the above procedure $\mathbf{x}_T$ is then the output of the MBO procedure. In high-dimensional input spaces, where valid $\mathbf{x}$ values lie on a thin manifold in a high-dimensional space, such an optimization procedure is prone to producing low-scoring inputs, which may not even be valid. This is because $\hat{f}$ may erroneously overestimate objective values at out-of-distribution points, which would naturally lead the optimization to such invalid points. Prior methods have sought to address this issue via generative modeling or explicit density estimation, so as to avoid out-of-distribution inputs. In the next section, we will describe how our method, COMs, instead trains the objective model in such a way that overestimation is prevented directly.

3.3 Conservative Objective Models for Offline Model-Based Optimization

In this section, we present our approach, conservative objective models (COMs). COMs learn estimates of the true function that do not overestimate the value of the ground truth objective on out-of distribution inputs in the vicinity of the training dataset. As a result, COMs prevent erroneous overestimation that would drive the optimizer (Equation 3.2) to produce out-of-distribution inputs with low values under the groundtruth objective function. We first discuss a procedure for learning such conservative estimates and then explain how these conservative models can be used for offline MBO.

3.3.1 Learning Conservative Objective Models (COMs)

The key idea behind our approach is to augment the objective for training of the objective model, $\hat{f}_\theta(\mathbf{x})$, with a regularizer that minimizes the expected value of this function on "adversarial" inputs where the value of the learned function $\hat{f}_\theta$ may be erroneously large. Such adversarial inputs are likely to be found by the optimizer during optimization, and hence, we need to train the learned function to not overestimate their values. How can we compute such adversarial inputs? Building on simple techniques for generating adversarial examples in supervised learning [42], we can run multiple steps of gradient ascent on the current snapshot of the learned function $\hat{f}(\mathbf{x})$ starting from various inputs in the training dataset to obtain such adversarial inputs. For concise notation in the exposition, we denote the distribution of all adversarial inputs found via this gradient ascent procedure as $\mu(\mathbf{x})$. Samples from $\mu(\mathbf{x})$ are obtained by

sampling a datapoint from the training set and running several steps of gradient ascent on $\hat{f}(\mathbf{x})$.

$$\mu(\mathbf{x}) = \sum_{\mathbf{x}_0 \in \mathcal{D}} \delta_{\mathbf{x}=\mathbf{x}_T} : \mathbf{x}_{t+1} = \mathbf{x}_t + \eta \nabla_{\mathbf{x}} \hat{f}_\theta(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_t} \tag{3.3}$$

While simply minimizing the function values under this adversarial distribution $\mu(\mathbf{x})$ should effectively reduce the value of the learned $\hat{f}$ at these inputs, this can result in systematic underestimation even for in-distribution points. To balance out this regluarization, our approach additionally *maximizes* the expected value of this function on the training dataset. This can be formalized as maximizing the value of $\hat{f}(\mathbf{x})$ under the empirical distribution of inputs $\mathbf{x} \in \mathcal{D}$ given by: $\hat{D}(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{D}} \delta_{\mathbf{x}=\mathbf{x}_i}$.

In Section 3.4, we will show that the minimization and maximization terms balance out, and this objective learns a function $\hat{f}_\theta(\mathbf{x})$ that is a lower bound on the true function $f(\mathbf{x})$ for inputs that are encountered during the optimization process, under several assumptions. This approach is inspired by recent work in offline RL [67], where a similar objective is used to learn conservative value functions. We will elaborate on this connection in Section 3.5.

Formally, our training objective is given by the following equation, where α is a parameter that trades off conservatism for regression:

$$\hat{f}_\theta^\star \leftarrow \arg\min_{\theta \in \Theta} \overbrace{\alpha \left(\mathbb{E}_{\mathbf{x} \sim \mu(\mathbf{x})} \left[\hat{f}_\theta(\mathbf{x}) \right] - \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[\hat{f}_\theta(\mathbf{x}) \right] \right)}^{\text{COMs regularizer}} + \frac{1}{2} \underbrace{\mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\left(\hat{f}_\theta(\mathbf{x}) - y \right)^2 \right]}_{\text{standard supervised regression}}, \tag{3.4}$$

This idea is schematically depicted in Figure 3.1. The value of α and the choice of distribution $\mu(\mathbf{x})$ play a crucial role in determining the behavior of this approach. If the chosen α is very small, then the resulting $\hat{f}_\theta^\star(\mathbf{x})$ may not be a conservative estimate of the actual function $f(\mathbf{x})$, whereas if the chosen α is too large, then the learned function will be too conservative, and not allow the optimizer to deviate away from the dataset at all. We will discuss our strategy for choosing α in the next section. As noted earlier, our choice of $\mu(\mathbf{x})$ specifically focuses on adversarial inputs that the optimizer is likely to encounter while optimizing the input. We compute this distribution $\mu(\mathbf{x})$ by sampling a starting point $\mathbf{x}_0$ from the dataset $\mathcal{D}$, and then performing several steps of gradient ascent on $\hat{f}_\theta$ starting from this point.

Algorithm 1 COM: Training Conservative Models

1: Initialize $\hat{f}_\theta$. Pick η, α and initialize dataset $\mathcal{D}$.
2: **for** $i = 1$ to training_steps **do**
3: Sample $(\mathbf{x}_0, y) \sim \mathcal{D}$
4: Find $\mathbf{x}_T(\mathbf{x}_0)$ via gradient ascent from $\mathbf{x}_0$:
$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \nabla_{\mathbf{x}} \hat{f}_\theta(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_t}; \quad \mu(\mathbf{x}) = \sum_{\mathbf{x}_0 \in \mathcal{D}} \delta_{\mathbf{x}=\mathbf{x}_T(\mathbf{x}_0)}$.
5: Minimize $\mathcal{L}(\theta; \alpha)$ with respect to θ.
$\mathcal{L}(\theta;\alpha) = \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{D}}(\hat{f}_\theta(\mathbf{x}_0) - y)^2 - \alpha \mathbb{E}_{\mathbf{x}_0}[\hat{f}_\theta(\mathbf{x}_0)] + \alpha \mathbb{E}_{\mu(\mathbf{x})}[\hat{f}_\theta(\mathbf{x})]$
$\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}(\theta; \alpha)$
6: **end for**

Algorithm 2 COM: Finding $\mathbf{x}^\star$

1: Initialize optimizer at the optimum in $\mathcal{D}$:
$\tilde{\mathbf{x}} = \arg\max_{(\mathbf{x},y) \in \mathcal{D}} \; y$
2: Find $\mathbf{x}^\star$ via gradient ascent from $\tilde{\mathbf{x}}$:
$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \nabla_{\mathbf{x}} \mathcal{L}_{\text{opt}}(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_t}$
where $\mathcal{L}_{\text{opt}}(\mathbf{x}) := \hat{f}^\star_\theta(\mathbf{x})$
3: Return the solution $\mathbf{x}^\star = \mathbf{x}_T$.

3.3.2 Optimizing a Conservative Objective Model

Once we have a trained conservative model from Equation 3.4, we must use this learned model for finding the best possible input, $\mathbf{x}^\star$. Prior works [66, 12] use a standard (non-conservative) model of the objective function in conjunction with generative models or density estimators to restrict the optimization to in-distribution values of $\mathbf{x}^\star$. However, since our conservative training method trains $\hat{f}^\star_\theta$ to explicitly assign low values to out-of-distribution inputs, we can use a simple gradient-ascent style procedure in the input space to find the best possible solution.

Specifically, our optimizer runs gradient-ascent for T iterations starting from an input in the dataset ($\mathbf{x}_0 \in \mathcal{D}$), in each iteration trying to move the design in the direction of the gradient of the learned model $\hat{f}^\star_\theta$. Starting from the best point in the dataset, $\mathbf{x}_0 \in \mathcal{D}$, our optimizer performs the following update (also shown in Algorithm 2, Line 2):

$$\forall\, t \in [T], \mathbf{x}_0 \in \mathcal{D}; \qquad \mathbf{x}_{t+1} = \mathbf{x}_t + \eta \nabla_{\mathbf{x}} \mathcal{L}_{\text{opt}}(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_t} \quad \text{where} \quad \mathcal{L}_{\text{opt}}(\mathbf{x}) := \hat{f}^\star_\theta(\mathbf{x}). \tag{3.5}$$

Equation 3.5 ensures that the value of the learned function $\hat{f}_\theta(\mathbf{x}_{t+1})$ is larger than the value at its previous iterate $\mathbf{x}_t$. Furthermore, the number of iterations T of gradient ascent during optimization in Equation 3.5 is the identical to the number of steps that

we use to generate adversarial examples, $\mu(\mathbf{x})$ in Equation 3.3. This ensures that the optimizer only queries the region when the learned function $\hat{f}_\theta(\mathbf{x})$ is indeed conservative and a valid lower bound.

3.3.3 Using COMs for MBO: Additional Decisions

Next we discuss other design decisions that appear in COMs training (Equation 3.4) or when optimizing the input against a learned conservative model (Equation 3.5).

Choosing α. The hyperparameter α in Equation 3.4 plays an important role in weighting conservatism against accuracy. Without access to additional active data collection for evaluation, tuning this hyperparameter for each task can be challenging. Therefore, in order to turn COMs into a task-agnostic algorithm for offline MBO, we devise an automated procedure for selecting α. As discussed previously, if α is too large, $\hat{f}^\star_\theta$ is expected to be too conservative, since it would assign higher values to points in the dataset, and low values to *all* other points. Selecting a single value of α that works for many problems is difficult, since its effect depends strongly on the magnitude of the objective function. Instead, we use a modified training procedure that poses Equation 3.4 as a constrained optimization problem, with α assuming the role of a Lagrange dual variable for satisfying a constraint that controls the difference in values of the learned objective under $\mu(\mathbf{x})$ and $\mathcal{D}(\mathbf{x})$. This corresponds to solving the following optimization problem:

$$\hat{f}^\star_\theta \leftarrow \arg\min_{\theta \in \Theta} \quad \frac{1}{2}\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\left(\hat{f}_\theta(\mathbf{x}) - y\right)^2\right] \tag{3.6}$$

$$\text{s.t.} \quad \left(\mathbb{E}_{\mathbf{x}\sim\mu(\mathbf{x})}\left[\hat{f}_\theta(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\hat{f}_\theta(\mathbf{x})\right]\right) \leq \tau. \tag{3.7}$$

While Equation 3.6 introduces a new hyperparameter τ in place of α, this parameter is easier to select by hand, since its optimal value does not depend on the magnitude of the objective function as we can normalize the objective values to the same range before use in Equation 3.6, and therefore, a single choice works well across a diverse range of tasks. We find that a single value of τ is effective on every continuous task ($\tau = 0.5$) and discrete task ($\tau = 2.0$) respectively, and empirically ablate the choice of τ in Figure 3.3.

Selecting optimized designs $\mathbf{x}^\star$. So far we have discussed how COMs can be trained and used for optimization; however, we have not established a way to determine which $\mathbf{x}_t$ (Equation 3.5) encountered in the optimization trajectory should be used as our final solution $\mathbf{x}^\star$. The most natural choice is to pick the final $\mathbf{x}_T$ found by the optimizer as the solution. We uniformly choose $T = 50$ steps. While the choice of T should, in principle, affect the solution found by any gradient-ascent style optimizer, we found COMs to be quite stable to different values of T, as we will elaborate empirically on in Section 3.6.2, Figure 3.3. Of course, there are many other possible ways of selecting T,

including ideas inspired from offline model-selection methods in offline reinforcement learning [112], but our simple procedure, which is also popular in offline RL [32], ensures that the optimizer only queries the regions of the input space where the learned function is indeed trained to be conservative and is also sufficient to obtain good optimization performance.

3.3.4 Overall Algorithm and Practical Implementation

Finally, we combine the individual components discussed so far to obtain a complete algorithm for offline model-based optimization. Pseudocode for our algorithm is shown in Algorithm 1. COMs parameterize the objective model, $\hat{f}_\theta(\mathbf{x})$, via a feed-forward neural network with parameters θ. Our method then alternates between approximately generating samples $\mu(\mathbf{x})$ via gradient ascent (Line 4), and optimizing parameters θ using Equation 3.5 (Line 5). Finally, at the end of training, we run the gradient ascent procedure over the learned objective model $\hat{f}^\star_\theta(\mathbf{x})$ for a large T number of ascent steps and return the final design $\mathbf{x}_T$ as $\mathbf{x}^\star$.

Implementation details. Full implementation details for our method can be found in Appendix B.1. Briefly, for all of our experiments, the conservative objective model $\hat{f}_\theta$ is modeled as a neural network with two hidden layers of size 2048 each and leaky ReLU activations. More details on the network structure can be found in Appendix B.3. In order to train this conservative objective model, we use the Adam optimizer [60] with a learning rate of 10^{-3}. Empirically, we found that if η is too large, gradient ascent begins to produce inputs $\mathbf{x}_T$ that do not maximize the values of $\hat{f}^\star_\theta(\mathbf{x}_T)$, so we select the largest η such that successive $\mathbf{x}_t$ follow the gradient vector field of $\hat{f}^\star_\theta(\mathbf{x}_t)$. For computing samples $\mu(\mathbf{x})$, we used 50 gradient ascent steps starting from a given design in the dataset, $\mathbf{x}_0 \in \mathcal{D}$. During optimization, we used the gradient-ascent optimizer with a learning rate of 0.05 for continuous tasks and 2.0 for discrete tasks. As we will also show in our experiments (Section 3.6.2), this produces stable optimization behavior for all tasks we attempted. Finally, in order to choose the step T in Equation 3.5 that is supposed to provide us with the final solution $\mathbf{x}^\star = \mathbf{x}_T$, we pick a universal step of $T = 50$.

3.4 Theoretical Analysis of COMs

We will now theoretically analyze conservative objective models, and show that the conservative training procedure (Equation 3.4) indeed learns a conservative model of the objective function. To do so, we will show that under Equation 3.4, the values of all inputs in regions found within T steps of gradient ascent starting from any input $\mathbf{x}_0 \in \mathcal{D}$ are lower-bounds on their actual value. For analysis, we will denote $\overline{\mathcal{D}}(\mathbf{x})$ as the smoothed density of $\mathbf{x}$ in the dataset $\mathcal{D}$ (see Appendix B.2 for a formal definition).

We will express Equation 3.4 in an equivalent form that factorizes the distribution $\mu(\mathbf{x})$ as $\mu(\mathbf{x}) = \sum_{\mathbf{x}_0 \sim \mathcal{D}} \overline{\mathcal{D}}(\mathbf{x}_0)\mu(\mathbf{x}_T|\mathbf{x}_0)$:

$$\min_{\theta \in \Theta} \ \alpha \left(\mathbb{E}_{\mathbf{x}_0 \sim \overline{\mathcal{D}}, \mathbf{x}_T \sim \mu(\mathbf{x}_T|\mathbf{x}_0)} \left[\hat{f}_\theta(\mathbf{x}_T) \right] - \mathbb{E}_{\mathbf{x} \sim \overline{\mathcal{D}}} \left[\hat{f}_\theta(\mathbf{x}) \right] \right) \tag{3.8}$$

$$+ \frac{1}{2} \mathbb{E}_{(\mathbf{x}, y) \sim \overline{\mathcal{D}}} \left[\left(\hat{f}_\theta(\mathbf{x}) - y \right)^2 \right]. \tag{3.9}$$

While $\mu(\mathbf{x}_T|\mathbf{x}_0)$ is a Dirac-delta distribution in practice (Section 3.3), for our analysis, we will assume that it is a distribution centered at $\mathbf{x}_T$ and $\mu(\mathbf{x}_T|\mathbf{x}_0) > 0 \ \forall \ \mathbf{x}_T \in \mathcal{X}$. This condition can be easily satisfied by adding random noise during gradient ascent while computing $\mathbf{x}_T$. We will train $\hat{f}_\theta$ using gradient descent and denote $k = 1, 2, \cdots$ as the iterations of this training procedure for $\hat{f}_\theta$.

We first summarize some assumptions used in our analysis. We assume that the true function $f(\mathbf{x})$ is L-Lipschitz over the input space $\mathbf{x}$. We also assume that the learned function $\hat{f}_\theta(\mathbf{x})$ is $\widehat{L}$-Lipschitz and $\widehat{L}$ is sufficiently larger than L. For analysis purposes, we will define a conditional distribution, $\overline{\mathcal{D}}(\mathbf{x}'|\mathbf{x})$, to be a Gaussian distribution centered at $\mathbf{x}$: $\mathcal{N}(\mathbf{x}'|\mathbf{x}, \sigma^2)$. We will not assume a specific parameterization for the objective model, $\hat{f}_\theta$, but operate under the neural tangent kernel (NTK) [56] model of neural nets. The neural tangent kernel of the function $\hat{f}(\mathbf{x})$ be defined as: $\mathbf{G}_f(\mathbf{x}_i, \mathbf{x}_j) := \nabla_\theta \hat{f}_\theta(\mathbf{x}_i)^T \nabla_\theta \hat{f}_\theta(\mathbf{x}_j)$. Under these assumptions, we build on the analysis of conservative Q-learning [67] to prove our theoretical result in Theorem 3.4.1, shown below:

Proposition 3.4.1 (Conservative training lower-bounds the true function)**.** *Assume that $\hat{f}_\theta(\mathbf{x})$ is trained with conservative training by performing gradient descent on θ with respect to the objective in Equation 3.8 with a learning rate η. The parameters in step k of gradient descent are denoted by θ^k, and let the corresponding conservative model be denoted as $\hat{f}^k_\theta$. Let $\mathbf{G}$, μ, $\widehat{L}$, L, $\overline{\mathcal{D}}$ be defined as discussed above. Then, under assumptions listed above, $\forall \ \mathbf{x} \in \mathcal{D}, \mathbf{x}'' \in \mathcal{X}$, the conservative model at iteration $k+1$ of training satisfies:*

$$\hat{f}^{k+1}_\theta(\mathbf{x}'') := \max \ \Big\{ \hat{f}^{k+1}_\theta(\mathbf{x}) - \widehat{L}||\mathbf{x}'' - \mathbf{x}||_2, \\ \tilde{f}^{k+1}_\theta(\mathbf{x}'') - \eta\alpha \mathbb{E}_{\mathbf{x} \sim \overline{\mathcal{D}}, \mathbf{x}' \sim \mu}[\mathbf{G}^k_f(\mathbf{x}'', \mathbf{x}')] \\ + \eta\alpha \mathbb{E}_{\mathbf{x} \sim \overline{\mathcal{D}}, \mathbf{x}' \sim \overline{\mathcal{D}}}[\mathbf{G}^k_f(\mathbf{x}'', \mathbf{x}')] \Big\},$$

where $\tilde{f}^{k+1}_\theta(\mathbf{x}'')$ is the resulting $(k+1)$-th iterate of $\hat{f}_\theta$ if conservative training were not used. Thus, if α is sufficiently large, the expected value of the asymptotic function, $\hat{f}_\theta := \lim_{k \to \infty} \hat{f}^k_\theta$, on inputs $\mathbf{x}_T$ found by the optimizer, lower-bounds the value of the true function $f(\mathbf{x}_T)$:

$$\mathbb{E}_{\mathbf{x}_0 \sim \mathcal{D}, \mathbf{x}_T \sim \mu(\mathbf{x}_T|\mathbf{x}_0)}[\hat{f}_\theta(\mathbf{x}_T)] \leq \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{D}, \mathbf{x}_T \sim \mu(\mathbf{x}_T|\mathbf{x}_0)}[f(\mathbf{x})].$$

A proof for Proposition 3.4.1 including a complete formal statement can be found in Appendix B.2. The intuition behind the proof is that inducing conservatism in the

	GFP	TF Bind 8	UTR	# Optimal	Norm. avg. perf.
$\mathcal{D}$ (**best**)	0.789	0.439	0.593		
Auto. CbAS	**0.865 ± 0.000**	0.910 ± 0.044	0.691 ± 0.012	1 / 7	0.687
CbAS	**0.865 ± 0.000**	0.927 ± 0.051	**0.694 ± 0.010**	3 / 7	0.699
MINs	**0.865 ± 0.001**	0.905 ± 0.052	**0.697 ± 0.010**	**4 / 7**	0.745
BO-qEI	0.254 ± 0.352	0.798 ± 0.083	0.684 ± 0.000	0 / 7	0.629
CMA-ES	0.054 ± 0.002	0.953 ± 0.022	**0.707 ± 0.014**	2 / 7	0.674
Grad.	0.864 ± 0.001	**0.977 ± 0.025**	**0.695 ± 0.013**	3 / 7	0.750
Grad. Min	0.864 ± 0.000	**0.984 ± 0.012**	**0.696 ± 0.009**	3 / 7	0.829
Grad. Mean	0.864 ± 0.000	**0.986 ± 0.012**	0.693 ± 0.010	2 / 7	0.852
REINFORCE	**0.865 ± 0.000**	0.948 ± 0.028	0.688 ± 0.010	1 / 7	0.541
COMs (Ours)	*0.864 ± 0.000*	*0.945 ± 0.033*	**0.699 ± 0.011**	**4 / 7**	**0.985**

	Superconductor	Ant Morphology	D'Kitty Morphology	Hopper Controller	
$\mathcal{D}$ (**best**)	0.399	0.565	0.884	1.0	
Auto. CbAS	0.421 ± 0.045	0.882 ± 0.045	0.906 ± 0.006	0.137 ± 0.005	
CbAS	**0.503 ± 0.069**	0.876 ± 0.031	0.892 ± 0.008	0.141 ± 0.012	
MINs	0.469 ± 0.023	**0.913 ± 0.036**	**0.945 ± 0.012**	0.424 ± 0.166	
BO-qEI	0.402 ± 0.034	0.819 ± 0.000	0.896 ± 0.000	0.550 ± 0.118	
CMA-ES	0.465 ± 0.024	**1.214 ± 0.732**	0.724 ± 0.001	0.604 ± 0.215	
Grad.	**0.518 ± 0.024**	0.293 ± 0.023	0.874 ± 0.022	1.035 ± 0.482	
Grad. Min	**0.506 ± 0.009**	0.479 ± 0.064	0.889 ± 0.011	1.391 ± 0.589	
Grad. Mean	**0.499 ± 0.017**	0.445 ± 0.080	0.892 ± 0.011	1.586 ± 0.454	
REINFORCE	0.481 ± 0.013	0.266 ± 0.032	0.562 ± 0.196	-0.020 ± 0.067	
COMs (Ours)	0.439 ± 0.033	**0.944 ± 0.016**	**0.949 ± 0.015**	**2.056 ± 0.314**	

Table 3.1: Comparative evaluation of COMs against prior methods in terms of the mean 100th-percentile score and its standard deviation over 8 trials. Tasks include **Superconductor-RandomForest-v0**, **HopperController-Exact-v0**, **AntMorphology-Exact-v0**, and **DKittyMorphology-Exact-v0**, which have a continuous design space and **GFP-Transformer-v0**, **TFBind8-Exact-v0**, and **UTR-ResNet-v0** with a discrete design space. COMs perform strictly better on high-dimensional tasks, obtaining about **1.3x** gains on Hopper Controller, and compelling gains on Ant Morphology and D'Kitty Morphology tasks. In addition, COMs is able to consistently find solutions that outperform the best training point for each task, given by $\mathcal{D}$ **(best)**. For each task, algorithms within one standard deviation of having the highest performance are bolded. COMs attain the optimal performance in 4/7 tasks ("# Optimal") attaining a normalized average performance of **0.985** compared to 0.852 for the next best method, outperforming other methods as indicated.

function $\hat{f}_\theta$ at each gradient step of optimizing Equation 3.8 makes the asymptotic function be conservative. Moreover, the larger the value of α, the more conservative the function $\hat{f}_\theta$ is on points $\mathbf{x}'$ found via gradient ascent, i.e., points with high density under $\mu(\mathbf{x}_T|\mathbf{x}_0)$, in expectation. Finally, when gradient ascent is used to find $\mathbf{x}^\star$ on the learned conservative model, $\hat{f}_\theta$, and the number of steps of gradient ascent steps is less than T, as we do in practice via Equation 3.5, this bound with additional offset will hold for the point $\mathbf{x}^\star$ in expectation, and therefore the estimated value of this point will not overestimate its true value. This additional offset depends on the Lipschitz constant $\widehat{L}$ and the distance between $\mathbf{x}^*$ and the the optimized solutions $\mathbf{x}_T$ found for other data points, $\mathbf{x}_0 \in \mathcal{D}$.

3.5 Related Work

We now briefly discuss prior works in MBO, including prior work on active model-based optimization and work that utilizes offline datasets for data-driven MBO.

Bayesian optimization. Most prior work on model-based optimization has focused on the active setting, where derivative free methods such as the cross-entropy method [92] and other methods derived from the REINFORCE trick [127, 91], reward-weighted regression [86], and Gaussian processes [106, 101, 105] have been utilized. Most of these methods focus mainly on low-dimensional tasks with active data collection. Practical approaches have combined these methods with Bayesian neural networks [106, 105], latent variable models [58, 34, 33], and ensembles of learned score models [3, 4, 75]. These methods still require actively querying the true function $f(\mathbf{x})$. Further, as shown by [12, 29, 66], these Bayesian optimization methods are susceptible to producing invalid out-of-distribution inputs in the offline setting. Unlike these methods, COMs are specifically designed for the offline setting with high-dimensional inputs, and avoid out-of-distribution inputs.

Offline model-based optimization. Recent works have also focused on optimization in the completely offline setting. Typically these methods utilize a generative model [62, 41] that models the manifold of inputs. [12, 29] use a variational autoencoder [62] to model the space of $\mathbf{x}$ and use it alongside a learned objective function. [66] use a generative model to parameterize an inverse map from the scalar objective y to input $\mathbf{x}$ and search for the optimal one-dimensional y during optimization. Modeling the manifold of valid inputs globally can be extremely challenging (see Ant, Hopper, and DKitty results in Section 4.6), and as a result these generative models often need to be tuned for each domain [114]. In contrast, COMs do not require any generative model, and fit an approximate objective function with a simple regularizer, providing both a simpler, easier-to-use algorithm and better empirical performance. Fu and Levine [30] also avoid training a generative model, but instead use normalized maximum likelihood, which requires training multiple discriminative models—COMs only requires one—and quantizing y, which COMs does not.

Adversarial examples. As discussed in Section 3.2, MBO methods based on learned objective models naturally query the learned function on "adversarial" inputs, where the learned function erroneously overestimates the true function. This is superficially similar to adversarial examples in supervised learning [42], which can be generated by maximizing the input against the loss function. While adversarial examples have been formalized as out-of-distribution inputs lying in the vicinity of the data distribution and prior works have attempted to correct for them by encouraging smoothness [117] of the learned function, and there is evidence that robust objective models help mitigate over estimation [96], these solutions may be ineffective in MBO settings when the true function is itself non-smooth. Instead making conservative predictions on such adversarially generated inputs may prevent poor performance.

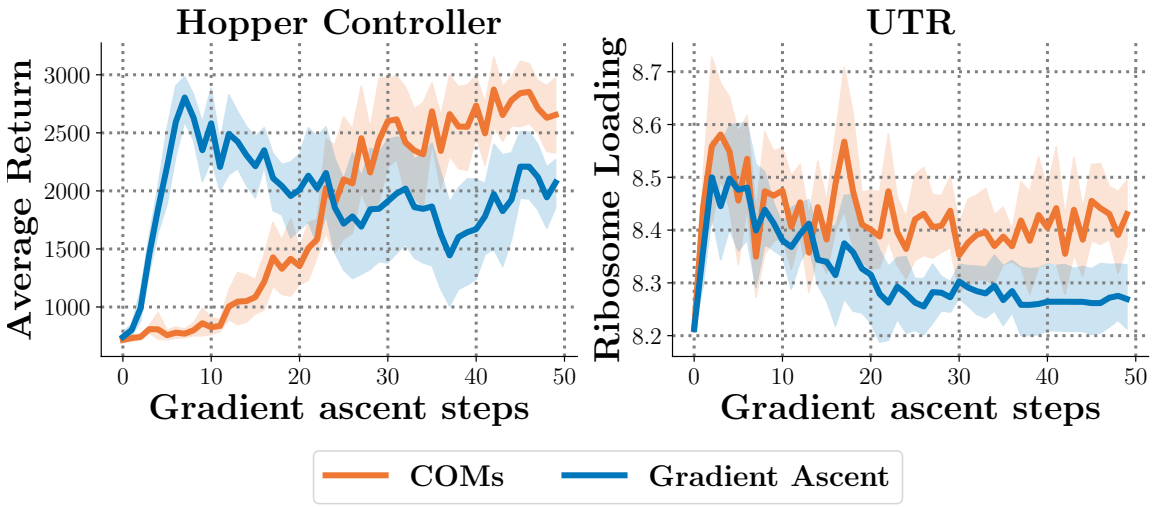


Figure 3.3: Stability of COMs versus naïve gradient ascent. The x-axis shows the number of gradient ascent steps taken on the design $\mathbf{x}^*$, and the y-axis shows the 100th percentile of the ground truth task objective function evaluated at every gradient step, which is used only for analysis only and is unavailable to the algorithm. In both cases, COMs reach solutions that remain at higher performance stably, indicating that COMs are less sensitive to varying numbers of gradient ascent steps performed during optimization.

3.6 Experimental Evaluation

To evaluate the efficacy of COMs for offline model-based optimization, we first perform a comparative evaluation of COMs on four continuous and three discrete offline MBO tasks based on problems in physical sciences, neural network design, material design, and robotics, proposed in the design-bench benchmark [114], that we also describe shortly. In addition, we perform an empirical analysis on COMs that aims to answer the following questions: **(1)** Is conservative training essential for improved performance and stability of COMs? How do COMs compare to a naïve objective model in terms of stability?, **(2)** How sensitive are COMs are to various design choices during optimization?, **(3)** Are COMs robust to hyperparameter choices and consistent to evaluation conditions? We answer these questions by studying the behavior of COMs under controlled conditions and using visualizations for our analysis. Code for reproducing our results is at https://github.com/brandontrabucco/design-baselines

3.6.1 Empirical Performance on Benchmark Tasks

We first compare COMs to a range of recently proposed methods for offline MBO in high-dimensional input spaces: CbAS [12], MINs [66] and and autofocused CbAS [29], that augments CbAS with a re-weighted objective model. Additionally, we also compare COMs to more standard baseline algorithms including REINFORCE [128], CMA-ES [49] , and BO-qEI, Bayesian Optimization with the quasi-expected improvement acquisition

function [129]. We also compare to a naïve **gradient ascent** baseline that first learns a model of the actual function via supervised regression (with no conservative term like COMs) and then optimizes this learned proxy via gradient ascent. CbAS variants and MINs train generative models such as VAEs [62] and GANs [41], which generally require task-specific neural net architectures, as compared to the substantially simpler discriminative models used for COMs. In fact, we use the same architecture for COMs on all the tasks. In addition, we instantiate this gradient ascent baseline with an ensemble of learned models of the objective function, with either a minimum (Grad. Min.) or mean (Grad. Mean) over the ensemble to obtain a learned prediction that is then optimized via gradient ascent.

Evaluation protocol. Our evaluation protocol follows prior work [12, 114]: we query each method to obtain the top $N = 128$ most promising optimized samples $\mathbf{x}_1^\star, \cdots, \mathbf{x}_N^\star$ according to the model, and then report the 100^{th} percentile ground truth objective values on this set of samples, $\max(\mathbf{x}_1^\star, \cdots, \mathbf{x}_N^\star)$, as well as the 50^{th} percentile objective values (See Appendix B.1 for numbers), averaged over 8 trials. We would argue that such an evaluation scheme is reasonable as it is typically followed in real-world MBO problems, where a set of optimized inputs are produced by the model, and the best performing one of them is finally used for deployment.

Offline MBO tasks. The tasks we use can be found in the design-bench benchmark [114] at github.com/brandontrabucco/design-bench. Here we briefly summarize the tasks: **(A)** Superconductor [29], where the goal is to optimize over 86-dimensional superconductor designs to maximize the critical temperature using 21263 points, **(B)** Hopper Controller [66], where the goal is to optimize over 5126-dimensional weights of a neural network policy on the Hopper-v2 gym domain using a dataset of 3200 points, and **(C)** Ant and **(D)** D'Kitty Morphology, where the goal is to design the 60 and 56-dimensional morphologies, respectively, of robots to maximize policy performance using datasets, both of size 25009. We also evaluate COMs on tasks with a discrete input space: **(E)** GFP [97], where the goal is to generate the protein sequence with maximum fluorescence, **(F)** TF Bind 8, where the goal is to design a length 8 DNA sequence with high binding affinity with particular transcriptions factors and **(G)** UTR [8], where the goal is to design a length 50 human 5'UTR DNA sequence with high ribosome loading. We represent discrete inputs in a transformed space of continuous-valued log probabilities for these tasks. Results for all baseline methods are based on numbers reported by Trabucco et al. [114]. Additional details for the setup of these tasks is provided in Appendix Section B.4.

Results on continuous tasks. Our results for different domains are shown in Table 3.1. On three out of four continuous tasks, COMs attain the best results, in some cases (e.g. (B) HopperController) attaining the performance of over **1.3x** the best prior method. In addition, COMs are shown to be the only method to attain higher performance that the best training point on every task. A naïve objective model without the conservative term, which is prone to falling off-the-manifold of valid inputs,

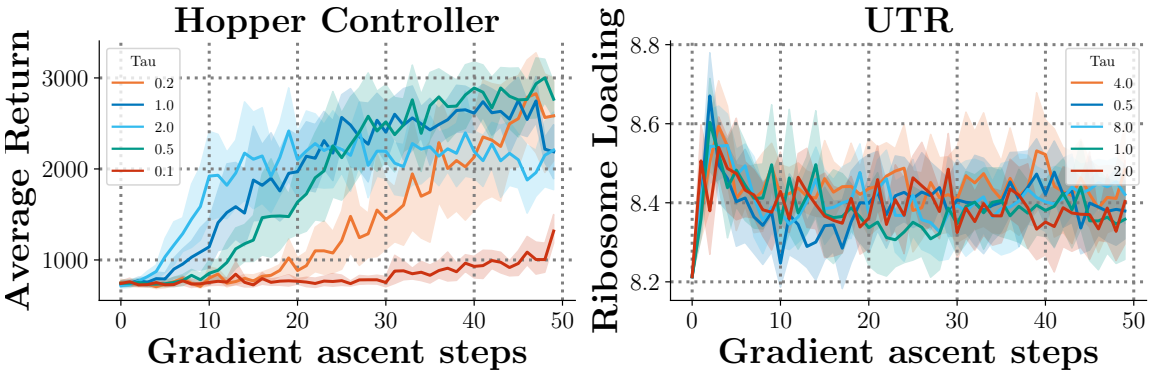


Figure 3.4: Ablation of stability and universality of τ. In each of the two plots, we instantiate COMs on the HopperController and UTR tasks, and vary τ that controls the degree of conservatism (Equation 3.6). The x-axis denotes the number of gradient ascent steps taken on the design $\mathbf{x}^*$ with respect to $\hat{f}_\theta$, and the y-axis indicates the 100th percentile of the ground truth function $\mathbf{x}$, which remains unobserved by the COMs algorithm, and only serves as an ablative visualization. The results demonstrate that increasing τ improves stability of COMs, and that COMs is robust to the particular choice of τ. We select $\tau = 0.5$ universally for continuous tasks, and $\tau = 2.0$ universally for discrete tasks.

struggles in especially high-dimensional tasks. Similarly, methods based on generative models, such as CbAS and MINs perform really poorly in the task of optimization over high-dimensional neural network weights in the HopperController task. These results indicate that COMs can serve as simple yet powerful method for offline MBO across a variety of domains. Furthermore, note that COMs only require training a parametric model $y = \hat{f}_\theta(\mathbf{x})$ of the objective function with a regularizer, without any need for training a generative model, which may be harder in practice to effectively tune.

Results on tasks with a discrete input space. COMs perform competitively with the best performing methods on GFP and TF Bind8, clearly outperforming the best sample in the observed task dataset. COMs attain almost the best performance on the GFP task and outperform CbAS variants and MINs on the TF Bind8 task. In addition, COMs outperform prior methods on the UTR task, attaining performance within one standard deviation of the highest performing method on that task.

Overall, COMs attain the best performance on **4/7** tasks, achieving a normalized average objective value of **0.985**, improving over the next best method by **16%** on average.

3.6.2 Ablation Experiments

In this section, we perform an ablative experimental analysis of COMs to answer questions posed at the beginning of Section 4.6. First, we evaluate the efficacy of

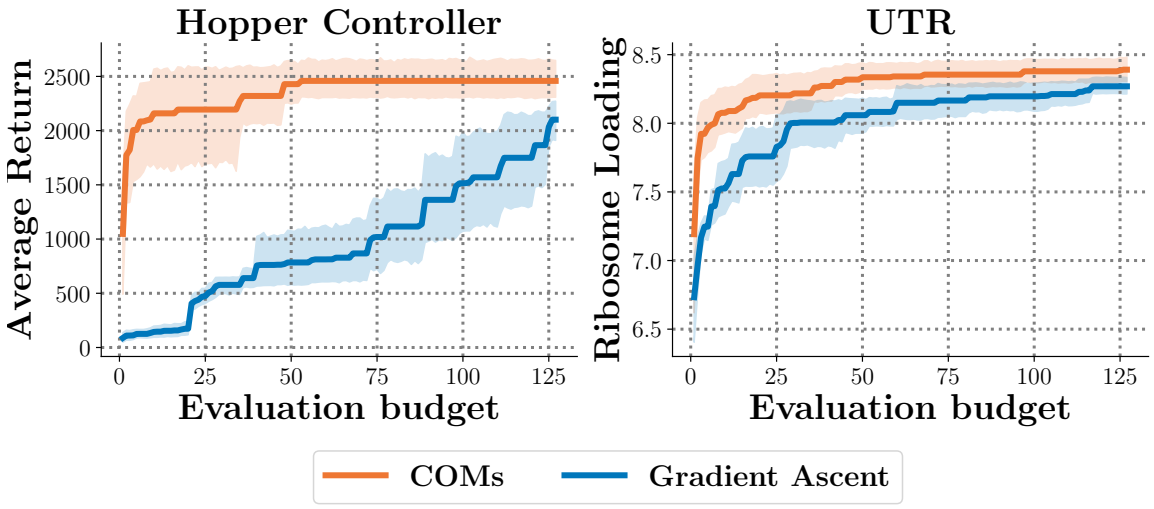


Figure 3.5: Ablation of consistency of COMs by visualizing sensitivity to the post-optimization evaluation budget. How does the performance of COMs and naïve gradient ascent vary as the evaluation budget is reduced? In our standard evaluation, we allow each offline MBO algorithm a "budget" of 128 evaluations for determining 100th and 50th percentile performance. The x-axis indicates the number N of allowed evaluations, and the y-axis indicates the 100th percentile performance of the chosen N points. As this evaluation budget is reduced, COMs is resilient, and remains superior to the naïve objective trained via supervised regression and optimized via standard gradient ascent. In the case of HopperController, COMs is nearly invariant to budgets down to size 55. This indicates COMs consistently produce optimized $\mathbf{x}^\star$ that attain high values under the true function.

using conservative training for learning a model of the objective function by comparing COMs to a naïve gradient ascent baseline and show that COMs are more *stable*, i.e., the optimization performance of COMs is much less sensitive to the number of gradient ascent steps used for optimization. Second, we evaluate the effect of varying values of the Lagrange threshold τ in Equation 3.6. Third, we demonstrate the *consistency* of COMs by evaluating the sensitivity of the optimization performance with respect to the number of samples N, that are used to compute the evaluation metric $\max(\mathbf{x}_1^*, \cdots, \mathbf{x}_N^*)$.

COMs are more stable than naïve gradient ascent. In order to better compare COMs and a naïve objective model optimized using gradient ascent, we visualize the true objective value for each $\mathbf{x}_t$ encountered during optimization (t in Line 2, Algorithm 2) in Figure 3.3. Observe that a naïve objective model can attain good performance for a "hand-tuned" number of gradient ascent steps, but it soon degrades in performance with more steps. This indicates that COMs are much more stable to the choice of number of gradient ascent steps performed than a naïve objective model.

Ablation of τ in Equation 3.6. In Figure 3.4, we evaluate the sensitivity of the performance of COMs as a function of the value of τ. As shown in Figure 3.4, we find that within the range of values evaluated, a higher value of τ gives rise to more stable

optimization behavior, and we were able to utilize a universal value of $\tau = 0.5$ for all tasks with a continuous input space and $\tau = 2.0$ for all tasks with a discrete input space.

COMs consistently produce well-performing inputs. Finally, we evaluate the sensitivity of COMs to the evaluation procedure itself. Standard evaluation practice in offline MBO dictates evaluating a batch of N most promising candidate inputs produced by the algorithm with the ground truth objective, where N remains constant across all algorithms [114, 12], and using the maximum value attained over these inputs as the performance of the algorithm, i.e., $\max(\mathbf{x}_1^*, \cdots, \mathbf{x}_N^*)$. This measures if the algorithm performs well within a provided "evaluation budget" of N evaluations. An algorithm is more *consistent* if it attains higher values of the groundtruth function with a smaller value of the evaluation budget, N. We used $N = 128$ for evaluating all methods in Table 3.1, but the value of N is technically a hyperparameter and an effective offline MBO method should be resilient to this value, ideally. COMs are resilient to N: as we vary N from 1 to 128 in Figure 3.5, COMs not only perform well at larger values of N, but are also effective with smaller budgets, reaching near-optimal performance on HopperController in with a budget of 55, while a naïve objective model needs a budget twice as large to reach its own optimal performance, which is lower than that of COMs.

3.7 Discussion and Conclusion

We proposed conservative objective models (COM), a simple method for offline model-based optimization, that learns a conservative estimate of the actual objective function and optimizes the input against this estimate. Empirically, we find that COMs give rise to good offline optimization performance and are considerably more stable than prior MBO methods, returning solutions that are comparable to and even better than the best existing MBO algorithms on four benchmark tasks. In this evaluation, COMs are consistently high performing, and in high-dimensional cases such as the Hopper Controller task, COMs improves on the next best method by a factor of **1.3x**. The simplicity of COMs combined with their empirical strength make them a promising optimization backbone to find solutions to challenging and high-dimensional offline MBO problems. In contrast to certain prior methods, COMs are designed to mitigate overestimation of out-of-distribution inputs close to the input manifold, and show improved stability at good solutions.

While our results suggest that COMs are effective on a number of MBO problems, there exists room for improvement. The somewhat naïve gradient-ascent optimization procedure employed by COMs can likely be improved by combining it with manifold modelling techniques, which can accelerate optimization by alleviating the need to traverse the raw input space. Similar to offline RL and supervised learning, learned

objective models in MBO are prone to overfitting, especially in limited data settings. Understanding different mechanisms by which overfitting can happen and correcting for it is likely to greatly amplify the applicability of COMs to a large set of practical MBO problems that only come with small datasets. Understanding why and how samples found by gradient ascent become off-manifold could result in a more powerful gradient-ascent optimization procedure that does not require a model-selection scheme.

Acknowledgements

We thank anonymous ICML reviewers, Aurick Zhou and Justin Fu for discussions and feedback on the tasks and the method described in this chapter, and all other members from RAIL at UC Berkeley for their suggestions, feedback and support. This work was supported by National Science Foundation, the DARPA Assured Autonomy Program, C3.ai DTI, and compute support from Google, Microsoft and Intel.

Chapter 4

Latent Conservative Objective Models for Offline Data-Driven Crystal Structure Prediction

4.1 Introduction

Data-driven optimization problems arise in many areas of science and engineering. In these settings, we have an unknown function that we would like to optimize with respect to its inputs, provided only with a dataset of input-output pairs. Examples include drug design, where inputs might be molecules and outputs are the efficacy of a drug, protein design, where inputs correspond to protein sequences and outputs are some metric such as fluorescence [97] or, as in our experiments, prediction of crystal structures, where inputs consist of crystal structures and outputs correspond to their formation energy. Such data driven optimization problems present several challenges. First, naïvely training a predictive model to predict the output from the input and then optimizing against such a model may lead to exploitation: a sufficiently strong optimizer can typically discover inputs that lead any learned model to extrapolate erroneously, and then exploit these errors to find inputs that "fool" the model into making the desired predictions. Second, even if a model can be suitably robustified, many of the most important design and optimization problems in science and engineering, including crystal structure prediction, require optimizing over complex sets and non-Euclidean manifolds, such that naïvely applying gradient-based methods in the input space is unlikely to result in a meaningful improvement.

In this chapter, we study these challenges in the context of crystal structure prediction

(CSP) [130]. Crystals are a class of solid-state materials characterized by the periodic placement of atoms. These structures form the basis of a wide variety of applications such as designing super-conductors, batteries [134], and solar cells [123]. Computationally identifying stable crystal geometries given a particular chemical formula typically involves minimizing (an estimate of) the crystal's formation energy to find the minimal energy structure. Conventional approaches to this problem rely on slow and compute-intensive DFT simulators [19], but more recent machine learning approaches dispense with DFT-based simulators and use databases of structures and their corresponding energies to train models that estimate crystal formation energy directly [35, 64]. However, the CSP problem suffers from both issues outlined above: crystal structures typically exhibit highly complex geometries characterized by periodicity of the lattice that forms the crystal and discrete (e.g., number and types of atoms in the chemical compound) and continuous features (e.g., positions of atoms in 3D space), which make it hard to produce reliable estimates of energies across the entire manifold of possible structures. Optimizing the structure using such inaccurate models then bears the risk of the optimization procedure "exploiting" these inaccuracies, resulting in structures that erroneously appear promising in this learned model but are not stable.

In this chapter, we aim to develop a data-driven optimization approach to overcome these challenges. First, to avoid the complexities associated with optimization over the complex manifold of crystal structures that consists of both discrete and continuous objects, our optimization procedure utilizes a crystal diffusion variational auto-encoder (CD-VAE) [133] to convert crystal structures into latent representations, which are much more amenable to simple gradient-based optimization. Second, to prevent the optimizer from getting "fooled" by the errors in the learned surrogate model, we extend conservative objective models [115], a robustification procedure, to our surrogate energy prediction model. This procedure explicitly pushes down over-estimated out-of-distribution designs in the latent space. Using a combination of these techniques, we develop a method for finding stable crystal structures that alleviates the time and compute costs associated with using DFT simulators, while also addressing the inaccuracies in a purely offline approach for designing crystal structures.

Our main contribution is a data-driven optimization approach, that we call latent conservative objective models (LCOM), for the problem of crystal structure prediction for solid materials. Our method leverages both advances in generative modeling over periodic solid-state materials for latent space learning [133] and recent advances in model-based optimization for robustifying the learned model to make it amenable to direct optimization of formation energies [115]. We summarize the approach in Figure 4.1. We instantiate our approach, latent conservative objective models (LCOMs), using crystal diffusion variational auto-encoders (CD-VAE) [133] for learning the latent space and conservative objective models (COMs) [115] for optimization. Empirically, we demonstrate that LCOMs are able to match the performance of the best prior method from Cheng et al. [17] while significantly reducing the total wall-clock time needed for

optimization by 40 times. In particular, a single optimization cycle in our framework takes an average of 2 seconds. This allows our model to provide predictions for more than 100 compounds in 4 minutes, much faster than prior works.

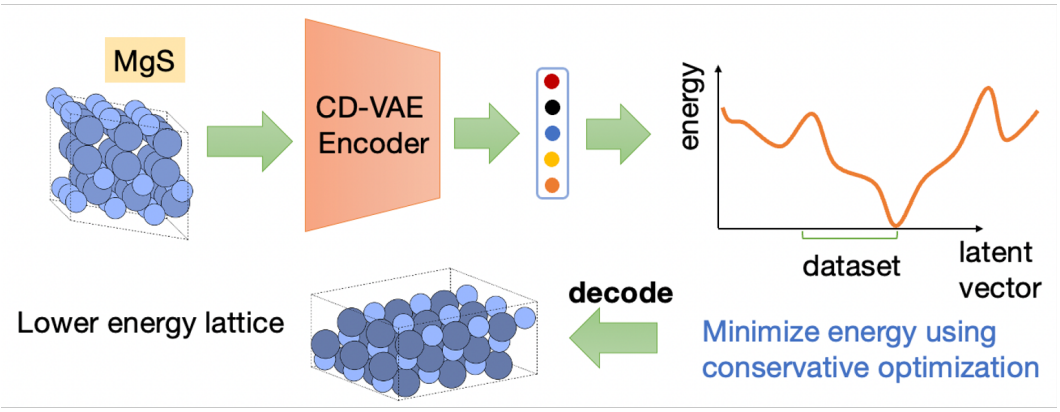


Figure 4.1: Overview of LCOMs. We train a graph-based CD-VAE to construct a latent space the represents crystal structure, conditioned on the molecular structure of the compound. Different points in this latent space correspond to different crystal structures, and we then optimize over the structure with simple gradient-based optimization methods operating on this latent space. To do so, we train a surrogate energy prediction model on the learned latent space via conservative training [115] to make it robust on out-of-distribution inputs, thus preventing the optimizer from discovering latent space points far from the training data for which the energy predictions yield erroneously low energies. The optimized latent vector is then decoded into a structure. Since the entire optimization is performed in the latent space, the comparatively complex encoder and decoder only need to be used once during optimization (to encode the initial structure and decode the final one).

4.2 Background and Definitions for Crystal Structures and Materials

In this section, we present the background definitions associated with crystals and solid-state materials. A crystal is a solid-state material characterized by a periodic placement of its constituents, which are chemical elements. The stoichiometry or the composition of a crystal, like NaCl, consists of the elements that make up the solid-state material (i.e., Na and Cl in this case) and in what ratio. In real-world applications of solid-state materials it is not enough to develop a material with a suitable chemical composition, but we must also account for the crystalline periodic structure of the solid and the atoms' positions with respect to it to assess the stability of a given compound.

Mathematically, we can describe the periodic structure of a chemical by defining its lattice L in 3D space, which repeats periodically. To characterize a lattice, we define its base vectors $\mathbf{v}, \mathbf{w}, \mathbf{z}$. Every point in the lattice is a linear combination of these vectors

using only integer coefficients.

$$p \in L \iff \exists n, m, k \in \mathbb{Z} \,|\, p = n\mathbf{v} + m\mathbf{w} + k\mathbf{z}. \tag{4.1}$$

Given a lattice L, the unit cell is the volume of 3D space contained between the base vectors, defined formally as follows:

$$\left\{ p \in \mathbb{R}^3 \,|\, \exists x, y, z \in [0, 1]\,,\; p = x\mathbf{v} + y\mathbf{w} + z\mathbf{z} \right\}. \tag{4.2}$$

We can obtain the entire lattice of a crystal by periodically repeating this unit cell in 3D space. Given a lattice in 3D space, a crystal is additionally characterized by how many atoms n are in the unit cell. We observe that the number n is always a multiple of the number of elements in the chemical composition of the material. For example, for a formula MgO_3, which consists of 4 atoms, we can have 4, 8, 12, or generally $4k$ elements in a unit cell (one element corresponding to one atom), but not 3 or 6 elements, which are not a multiple of 4.

Finally, we can describe atoms' types and positions with two matrices $A \in \mathbb{R}^{n \times 128}$, $X \in \mathbb{R}^{n \times 3}$. The matrix A identifies different atoms in the unit cell using a one-hot representation. Specifically, A_i is a vector with a 1 at position Z corresponding to the atomic number of the i-th element, and 0 everywhere else. The matrix X provides fractional coordinates for the atoms. These are coordinates between 0 and 1 with respect to the basis defined by the lattice base vectors. More specifically, the vector X_i tells us that the i-th element is at $X_i^1\mathbf{v} + X_i^2\mathbf{w} + X_i^3\mathbf{z}$ in the unit cell.

To summarize, a crystal is defined by three quantities: **(1)** A 3×3 matrix L representing the lattice, the rows of which corresponding to the base vectors of the lattice; **(2)** number (n) and types ($A \in \mathbb{R}^{n \times 128}$) in the chemical; and **(3)** atoms positions $X \in \mathbb{R}^{n \times 3}$, which is specified in terms of fractional coordinates between 0 and 1. In the following sections, we will denote a crystal with the variable $\mathbf{x}$ and we will use subscripts to refer to lattice parameters $\mathbf{x}_L$, atoms' types $\mathbf{x}_A$, and fractional coordinates $\mathbf{x}_X$. We finally remark that the majority of crystal and lattice configurations for a given chemical compound are "unstable" and would collapse to a different configuration when synthesized.

4.3 Problem Statement, Dataset, and Evaluation

Crystal structure prediction (CSP) is the problem of finding a crystal of a given chemical composition (e.g. NaCl, or MgO_3) that attains the global minimum of the crystal formation energy. Such a crystal is sythesizable and can be utilized for various downstream applications.

Problem 4.3.1 (**Crystal structure prediction**)**.** Given a chemical composition c, find the crystal $\mathbf{x}^* = (L^*, A^*, X^*)$ with lattice matrix L^*, atom types A^*, and atom coordinates X^* such that $\mathbf{x}^*$ minimizes the formation energy function E for the chemical composition.

$$\mathbf{x}^* = \text{argmin}_{\mathbf{x}}\, E\left(\mathbf{x}, c\right).$$

Why is solving CSPs hard? Only very few crystal structures are actually stable and only one of these stable structures is at a global minimum, which makes crystal structure prediction equivalent to searching for a "needle in a haystack". The difficulty of solving a CSP is further compounded by the fact that the search space over all possible crystal structures for a given chemical composition is quite complex and non-Euclidean. This is because there is no one-to-one correspondence between the matrices (A, X) and lattices L. Given a lattice matrix L, every other matrix that is rotationally equivalent or permutation equivalent represents the same lattice. Moreover, reducing the design space from all possible structures to only stable ones changes the manifold of designs considerably.

In principle, we could always attempt to find the globally optimal crystal structure by evaluating many possible candidate structures against a simulator, but simulators for CSP are typically based on density functional theory (DFT), and generally these are extremely slow in terms of the wall-clock time. Therefore, we intend to solve this problem using only existing static datasets (OQMD [94] and MatBench [24]). that contain several (sub-optimal) crystal structures for a variety of chemical compounds along with their corresponding formation energies. With no access to the simulator, our goal is to find a globally optimal crystal structure given a new chemical compound. We describe our procedure for constructing the dataset to train on and our evaluation protocol next.

4.3.1 Datasets for Training

To robustly evaluate our method, we consider two training scenarios with different datasets: the OQMD dataset [94] and the MatBench dataset [24], both of which consist of the crystal structures and formation energies for obtained via DFT simulations. Every sample in the dataset represents a stable crystal structure $\mathbf{x}$ (i.e., a crystal at a local minimum of energy) computed via numerical relaxation [47], together with its chemical composition c and formation energy $E(\mathbf{x}, c)$. We chose these datasets because of their large size (OQMD has more than 1 million examples), and the availability of more than one stable structure per chemical, all of which are not at their global optimum.

4.3.2 Held-Out Evaluation Datasets

We evaluate our offline optimization approach in terms of its efficacy in discovering the globally optimal structure for a given chemical compound. To this end, following the protocol of Cheng et al. [17], we construct a held-out test set consisting of some compounds and the associated globally optimal crystal structures (Table 4.1) and utilize this set for evaluations.

4.3.3 Evaluation Protocol

Following the evaluation protocol in prior works [133, 17], we test our method in terms of its efficacy in recovering the globally optimal structure on 26 of the 29 compounds in [17], where the 3 remaining compounds are omitted because they cannot be simulated with GPAW to compute their ground truth energy values. These compounds are listed in Table 4.1. For each of these compounds, we run our optimization process to convergence and check the final energy of the optimized design. We compare the optimized energy against the known global minimum energy. We consider it a success if the final energy of the optimized structure produced by our approach recovers the value of the known global minimum, up to a predefined noise threshold of 20% of the ground truth minimum energy.

To stress-test our gradient-based optimization approach, we seed the optimizer with a random stable initial crystal structure. We compute this initial stable structure by running simulations in the GPAW [27] simulator, an open-source DFT simulator fully integrated into python packages for chemistry like ase and pymatgen. Concretely, we utilized the following protocol for obtaining this initial crystal structure: **(i)** for every compound, we first select the number of atoms corresponding to the optimal compound design as listed in the materials project database, **(ii)** we then randomly initialize the lattice matrix and the atom coordinates, and **(iii)** we then run the process of structure relaxation in the simulator to obtain the closest local minimum (i.e., a closeby structure that is stable).

4.4 LCOMs: Latent Conservative Objective Models for Structure Prediction

To design crystal structures with the lowest possible energy entirely from an existing dataset of only sub-optimal structures, we utilize techniques from data-driven offline model-based optimization. Directly applying these techniques [115, 136, 87] for optimizing crystals is non-trivial as these methods typically employ optimization procedures that iteratively make local changes to the design (e.g., gradient-based updates or random mutations) to optimize a "surrogate" estimate of the objective function. Such iterative procedures fall short when optimizing over non-smooth geometries and non-Euclidean manifolds like that of crystals. To alleviate this issue, we propose an approach for offline optimization that first learns a latent vector representation for a crystal structure, then performs data-driven optimization in this vector space, and finally maps back the resulting outcome to a valid crystal structure. We outline each part of this procedure below.

4.4.1 Transforming Crystal Structures to a Latent Representation

Which sort of a latent representation space is especially desirable for CSP? Since one of the central challenges in our problem is the abundance of invalid or infeasible structures in the space of all possible crystals, it is very desirable to learn latent representations that only encode valid and feasible structures. Once we learn such a space, we can directly perform offline optimization in this latent space. If we can ensure that every possible latent vector corresponds to some feasible crystal structure, then we are guaranteed to at least prune out the possibility of finding infeasible structures during the optimization process. To this end, we train a crystal diffusion variational auto-encoder (CD-VAE) on our training dataset for various chemical compositions, and then run offline optimization in the latent space of this auto-encoder. Since our training dataset only consists of stable structures, the decoder of a well-trained CD-VAE should map latent vectors to the manifold of stable crystal structures only, which would greatly benefit optimization by enabling the optimizer to move in a much smaller manifold. Below we describe the architecture and the training objective for the CD-VAE.

CD-VAE. A CD-VAE [133] is composed of three parts: a graph neural network (GNN) encoder PGNN_{ENC} that takes a crystal $\mathbf{x}$ as input and outputs a latent vector representation, an NN predictor MLP_{AGG} that outputs lattice parameters $\mathbf{x}_L$ from its encoded representation $\text{PGNN}_{\text{ENC}}(\mathbf{x}, c)$, and a GNN diffusion denoiser PGNN_{DEC} that takes a random noisy crystal $\tilde{\mathbf{x}}$ and a latent encoding $\text{PGNN}_{\text{ENC}}(\mathbf{x}, c)$ as inputs, and outputs forces to apply on the atoms coordinates $\tilde{\mathbf{x}}_X$ to build the original crystal $\mathbf{x}$ via a diffusion process. Following Xie et al. [133], the encoder PGNN_{ENC} uses a DimeNet++ [64] architecture. Likewise, the decoder PGNN_{DEC} uses a GemNet-dQ [35] architecture. During decoding, CD-VAE initializes a structure with random lattice and coordinates and utilizes Langevin dynamics [108] to gradually recover the stable structure represented by the latent vector.

To make the notation compact, we will refer to the PGNN_{ENC} as ϕ, and thus the latent representation of a crystal $\mathbf{x}$ with chemical composition c will be denoted by $\phi(\mathbf{x}, c)$. We will use the notation $\text{PGNN}_{\text{DEC}}(z)$ to denote the structure obtained after applying the denoising process with latent vector z. Akin to a variational auto-encoder [61], CD-VAE [133] is also trained to maximize the likelihood of crystral structures seen in the dataset, agnostic of the energy objective that we wish to optimize.

Training objective for the latent representation. We follow the training objective utilized by the CD-VAE [133]. The first term in this objective is the reconstruction error over lattice parameters, formally defined as:

$$\mathcal{L}_{\text{AGG}}\left(\text{MLP}_{\text{AGG}}\left(\phi(\mathbf{x}, c)\right), \mathbf{x}_L\right) = \left\|\mathbf{x}_L - \phi(\mathbf{x}, c)\right\|^2. \tag{4.3}$$

Akin to a VAE [61], we also include a loss term minimizes the KL-divergence between a normal distribution over the latent representation induced by the encoder (with mean

$\phi(\mathbf{x}, c)$ and a learned standard deviation) and a standard multi-dimensional normal distribution.

The decoder of the CD-VAE is a denoising diffusion model [54] that attempts to transform an input latent vector into the corresponding crystal structure, starting from a random structure $\tilde{\mathbf{x}}$, which is iteratively refined via the diffusion process. Succinctly, the objective for training this term is given by

$$\mathcal{L}_{\text{DEC}}\left(\tilde{\mathbf{x}}, \mathbf{x}, \phi(\mathbf{x}, c)\right) = \frac{1}{2L} \sum_{j=1}^{L} \left[\mathbb{E}_{\sigma_j} \left\| \text{PGNN}_{\text{DEC}}\left(\tilde{\mathbf{x}}, \phi(\mathbf{x}, c)\right) - \frac{d\left(\mathbf{x}_X, \tilde{\mathbf{x}}_X\right)}{\sigma_j} \right\| \right], \tag{4.4}$$

where $\{\sigma_j\}_{j=1}^{L}$ are noise schedule scalars for the diffusion process and are in a geometric sequence with common ratio greater than 1. Finally, we remark that we do not utilize terms for reconstructing types or the number of atoms (i.e., $\mathbf{x}_A$ or $\mathbf{x}_n$) because our optimization procedure only aims to optimize over other parameters of the crystal lattice so the number and types of atoms are fixed.

4.4.2 Conservative Optimization in Latent Space

Once the encoder of the CD-VAE is trained, we can then train a surrogate model, $\widehat{E}_\theta(\phi(\mathbf{x}, c), c)$ to estimate the formation energy E of a crystal structure $\mathbf{x}$ for a given chemical composition, c via standard supervised regression. Then, we can simply optimize the crystal structure to maximize the outputs of this surrogate model. However, as prior works [67, 115] note, this simple strategy often fails at finding optimized designs due to the exploitation of errors in the learned surrogate model by the optimizer. To address this issue, we extend the conservative objective models (COMs) technique from Trabucco et al. [115] for optimizing crystals in the learned latent space.

Training latent space conservative models. To prevent the optimization procedure from exploiting inaccuracies in this learned surrogate model, we apply an additional regularizer from Trabucco et al. [115], Kumar et al. [68] to robustify the surrogate model. This regularizer mines for adversarial vectors in the latent space z^+ that appear to have very low energies $\widehat{E}_\theta(z^+, c)$ under the learned surrogate model, and then explicitly pushes up the predicted energy $\widehat{E}_\theta(z^+, c)$ on such adversarial z^+. Following the COMs approach [115], we interleave the training of the learned surrogate model $\widehat{E}_\theta$ with an optimization procedure $\mathsf{Opt}(\widehat{E}_\theta, c)$ that seeks to find the aforementioned adversarial vectors z^+ that optimize the current snapshot for the surrogate model, for a given chemical composition c. After these adversarial vectors are found, the training procedure explicitly pushes up the energy output of the surrogate model on such points. To compensate for the effect of increasing the learned energy values in an unbounded manner on all latent vectors, we additionally balance the push up term by pushing down the energy values on the latent representations induced by crystal structures in

the data. This idea can be formalized into the following loss for training $\widehat{E}_\theta$:

$$\min_\theta \quad \mathbb{E}_{c,\mathbf{x}\sim\mathcal{D}}\left[\left(\widehat{E}_\theta(\phi(\mathbf{x},c),c) - E(\mathbf{x},c)\right)^2\right] - \alpha\left(\mathbb{E}_{c,\mathbf{x}\sim\mathcal{D}}\left[\mathbb{E}_{z^+\sim\mathsf{Opt}(\widehat{E}_\theta,c)}[\widehat{E}_\theta(z^+,c)] - \widehat{E}_\theta(\phi(\mathbf{x},c),c)\right]\right).$$

We will discuss the precise formulation for Opt below. Crucially, note that unlike COMs [115], which directly runs gradient descent in the input space, our approach operates in the latent space.

Optimizing in the latent space. Once a conservative surrogate model $\widehat{E}_\theta(z,c)$ is obtained using the above training procedure, we must now optimize this model to obtain the best possible structures. The optimization procedure Opt that was used to obtain adversarial latent vectors in the training objective above can then be repurposed to obtain optimized latent vectors once the latent conservative model is trained. Since the latent space z is a continuous Euclidean vector space, for any given chemical composition c, our choice of Opt is to run T rounds of gradient descent on the surrogate energy $\widehat{E}_\theta(z,c)$ with respect to the latent vector z, starting from the latent vector z_0 corresponding to a random initial crystal structure. For a given c, this procedure can be formalized as follows:

$$\begin{aligned} z_{k+1} &\leftarrow z_k - \alpha\nabla_z\widehat{E}_\theta(z,c), \\ \text{where} \quad z_0 &\sim \phi(\mathbf{x}_0,c), \quad \mathbf{x}_0 \sim \mathcal{D}. \end{aligned} \tag{4.5}$$

Once this optimization procedure is run for T steps, we pass the final latent vector z_T to the decoder of the pre-trained CD-VAE to obtain the optimized crystal structure: $\widehat{\mathbf{x}}^* = \text{PGNN}_{\text{DEC}}(z_T)$.

4.4.3 Implementation Details

For obtaining the latent space, we train a CD-VAE identically to Xie et al. [133] on our datasets, following their implementation details for the encoder and the decoder. After training the CD-VAE, we encode molecule structures from the dataset into vectors, and these vectors are then used as inputs for training the optimization model. For training LCOMs, we represent the conservative objective model $\widehat{E}_\theta(\phi(\mathbf{x},c),c)$ as a neural network with two hidden layers of size 2048 each and leaky ReLU activations. For computing z^+, we perform one gradient descent step on the vector z from input latent space. We perform 50 gradient steps and get an optimized vector in the latent space. We then decode these latent vectors into an optimized crystal structure. For reporting statistically robust results, we repeat the optimization procedure for three seeds and average over the resulting energy value.

4.5 Related Work

One widely studied optimization-based approach to CSP utilizes evolutionary algorithms [73, 80]. For instance, USPEX [39] is an algorithm that uses a variety of

heuristic strategies for iteratively evolving structures directly in the space of the crystal parameters. Each of these intermediate structures need to be evaluated against the simulator, which repeatedly involves running relaxation to the nearest stable structure. This extensive use of simulation makes such an approach computationally impractical, necessitating offline learning-based approaches like our method that do not require any simulation.

Due to the availability of large public datasets such as the materials project database and the open catalyst project [15], recent works develop learning-based approaches for solving CSPs. Another subset includes methods that use different types of evolutionary optimizers to optimize a GNN-based surrogate energy model instead of the ground-truth energy function. This includes methods based on random search [17], particle swarm optimization [21], and Bayesian optimization [83]. In contrast, our method prescribes the use a conservative surrogate model of the energy function, that takes as input a latent representation of the crystal. As we show in our experimental results, both of these aspects are crucial for effectively tackling the crystal structure prediction problem.

Another related line of prior work aims to learn generative models for graph-structured data including, but not limited to crystal structures. This includes methods that leverage variational auto-encoders [103] normalizing flows [98], generative adversarial networks [59], recurrent neural networks [44], reinforcement learning techniques [84] or a combination of auto-encoders and diffusion models, for example, the CD-VAE [133], that we build upon. While these approaches aim to model the manifold the graph-structured data and our approach utilizes the latent space learned by one such approach, CD-VAE [133], our goal of optimizing the structure is distinct from the goal of modelling the data.

Model-based optimization (MBO) refers to the problem of optimizing an unknown function by constructing a surrogate model. Bayesian optimization represents one of the most widely known classes of MBO methods [106, 105, 38], but classically MBO requires iteratively sampling new function values, which can be very expensive when evaluating a crystal structure's energy requires an expensive simulation process. More recently, offline MBO methods, sometimes referred to as data-driven optimization, have been proposed to optimize designs based entirely on previously collected *static* datasets [11, 114, 66, 115, 87]. Our work builds on these methods, and is most closely related to the COMs algorithm proposed by Trabucco et al. [115]. However, while prior offline MBO methods focus on robustifying the surrogate model while optimizing in the original design space, we integrate these approaches with latent space optimization that makes it possible to optimize over the manifold of only stable crystal structures, while still using a simple gradient descent optimizer at the core.

4.6 Experimental Evaluation

The goal of our experimental evaluation is to evaluate the efficacy of LCOMs for crystal structure prediction by answering the following questions: **(1)** Can LCOMs successfully optimize in the latent representation space?, **(2)** Do LCOMs manage to effectively recover the optimal energy structure up to a pre-defined threshold of accuracy?, and **(3)** Does LCOM drastically reduce simulation wall-clock time compared to prior methods? To answer these questions, we evaluate LCOMs against prior methods following the protocol from Section 5.3 and then perform some diagnostic experimental studies that we will discuss in this section.

Comparing LCOMs with baselines and prior methods. We compare LCOMs to three methods from prior work [17]: random search (RAS), particle-swarm optimization (PSO), and Bayesian optimization (BO). We also study two baseline methods: a method that does not run any optimization in the latent space and simply constructs a stable structure for a given chemical compound via the decoder ("CD-VAE"), and a method where the crystal is optimized with a naïve supervised learning model in the latent space of the CD-VAE via gradient descent ("Supervised learning; SL"). Note that the latter is similar to LCOMs, but the surrogate energy prediction model is not trained with any conservatism, but rather with only standard supervised regression. We evaluate these methods on the 26 chemical compounds in our evaluation dataset, and present the results in Table 4.1. During evaluation, we mark a crystal structure successful if the energy of the optimized structure is close to the energy of the best known globally optimal structure up to a certain threshold. This threshold is defined as an upper bound of 0.2 on the quantity $\left(E(\mathbf{x}, c^*) - E(\mathbf{x}, \hat{c})\right) / |E(\mathbf{x}, c^*)|$ to account for imprecision in the simulator, where c^* is the ground truth optimal crystal and $\hat{c}$ is the optimized crystal discovered by the optimization algorithm.

The results in Table 4.1 show that when trained on OQMD, our method improves significantly over naïvely optimizing in the CD-VAE latent space without conservative training (supervised learning; SL), and also that it is competitive with the prior state-of-the-art methods RAS, PSO, and BO, exceeding the performance of the PSO baseline and matching BO and RAS, without needing any simulations (we will quantify the benefits on wall-clock time soon). A similar trend also holds for the MatBench dataset in Table 4.1, indicating that LCOMs is performant for different choices of the training data. We do note that the BO and RAS approaches outperform LCOMs when trained on the MatBench dataset, but this is an artifact of the difference of our evaluation metric compared to prior work [17] on this task[1]. No optimization over the latent space and only utilizing the CD-VAE decoder (denoted as CD-VAE) to decode a structure

[1]Our evaluation protocol for determining the optimality of a structure uses an energy threshold. This is a contrast to Cheng et al. [17], which adopts a manual inspection approach to check for equality between optimized and optimal structures as their evaluation criterion. As such, comparisons between our work and that of Cheng et al. [17] should be made with an understanding of this fundamental difference in evaluation methodology.

	OQMD				Baselines		MatBench			
Compounds	RAS*	PSO	BO*	**LCOMs**	SL	CD-VAE	RAS*	PSO	BO*	**LCOMs**
LiF		✓	✓	✓			✓			✓
NaF	✓	✓	✓		✓		✓		✓	
KF	✓		✓	✓		✓	✓		✓	✓
RbF				✓	✓	✓	✓		✓	✓
CsF				✓		✓	✓		✓	✓
LiCl									✓	
NaCl	✓	✓	✓	✓	✓	✓	✓		✓	✓
KCl	✓			✓			✓		✓	✓
RbCl	✓			✓			✓	✓	✓	✓
CsCl	✓		✓	✓			✓		✓	✓
BeO		✓	✓	✓		✓			✓	✓
MgO	✓		✓				✓	✓	✓	✓
CaO	✓		✓	✓			✓	✓	✓	✓
SrO	✓		✓	✓			✓	✓	✓	✓
BaO	✓		✓				✓			✓
ZnO		✓	✓	✓				✓	✓	✓
CdO						✓	✓		✓	
BeS	✓	✓	✓	✓			✓	✓	✓	✓
MgS	✓				✓		✓			✓
CaS	✓		✓	✓			✓		✓	✓
SrS	✓			✓	✓		✓		✓	✓
BaS	✓		✓				✓	✓	✓	
ZnS	✓		✓	✓					✓	✓
CdS			✓						✓	
C	✓						✓			
Si								✓	✓	
Accuracy	17/26	6/26	16/26	16/26	5/26	6/26	20/26	8/26	22/26	19/26

Table 4.1: Evaluation of LCOMs and other prior crystal structure prediction methods in terms of the accuracy of discovering the globally optimal structure for 26 compounds. Check marks indicate successful discovery as per our criterion. Note crucially that while we utilize a threshold on optimization energy to determine success, prior work utilizes a manual inspection protocol as discussed in the footnote in Section 4.6.

for the test chemical formula does not perform well.

Does LCOM improve over prior methods in terms of wall-clock time? Our method optimizes the crystal structure in the latent space of the CD-VAE, using gradient-based optimization. One advantage of this approach is computational efficiency, since the complex graph-based component of the pipeline is only used during the encoding and decoding stages at the beginning and end of the optimization, rather than at each optimization step. Hence, we measure the wall-clock time needed to run optimization with LCOMs comparatively against other prior methods in Table 4.2. Observe that while utilizing a graph neural network (GNN-BO) reduces the wall-clock time needed by about 875× compared to DFT-PSO that queries the simulator for every design, LCOMs further reduces the wall-clock time 40× by running optimization in the latent space of a CD-VAE, which does not require running expensive message passing loops of

a graph neural network encoder but rather runs relatively faster forward passes through small MLPs. This indicates that LCOMs not only discovers are more optimal crystal structure, but it does so 40× faster than the best prior method.

	DFT-PSO	GN-BO	**LCOMs**
Optimization time per structure (seconds)	70000	80	2

Table 4.2: Comparing wall-clock time for different methods. Observe that not using a simulator reduces the wall-clock time from 70000 seconds to 80 seconds per structure, and further utilizing a latent space surrogate model in LCOMs instead of a graph neural network model cuts down the total time further by 40× to 2 seconds.

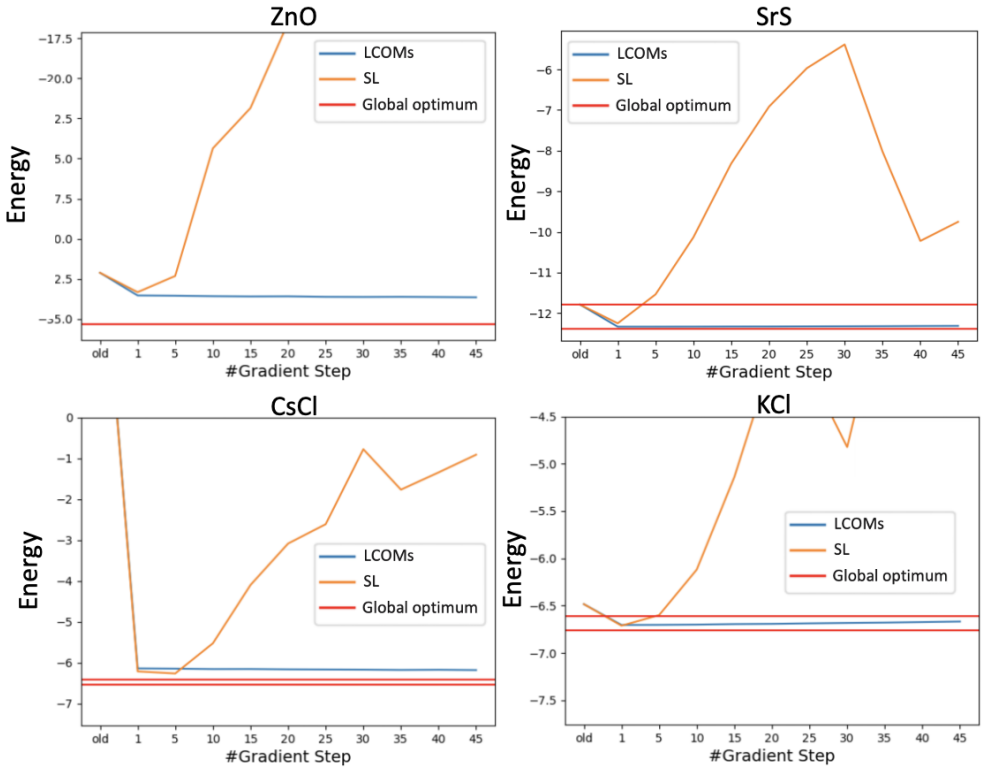


Figure 4.2: Comparing the energy of intermediate structures observed over the course of optimization with LCOMs (blue) and non-conservative models (orange). Note that while the non-conservative model gets exploited as more steps of gradient-based optimization are performed, structures discovered by conservative LCOMs attain lower energies after gradient descent, and the final structures are close to the global optimum (marked as red in the plot above).

How does conservative training influence optimization? In the next set of experiments, we aim to understand how conservative training of the surrogate model influences the behavior of gradient descent optimization in the CD-VAE latent space. If the energy model is trained naïvely (i.e., with standard supervised regression), it will make arbitrarily erroneous predictions when queried on adversarial, out-of-distribution crystals that differ too much from the training data. Some of these errors will be

under-estimation errors, and therefore, a strong optimizer would be able to exploit these errors to find points in the latent space for which the model erroneously predicts arbitrarily low energies. Empirically, we evaluate the performance of optimized crystals obtained by running 50 gradient steps of optimization on the learned surrogate models starting from an initial structure. Specifically, we compute the relative improvement in energy values after optimization, formally calculated as $(E(\mathbf{x}, c_0) - E(\mathbf{x}, \hat{c}))\,/|E(\mathbf{x}, c_0)|$, where c_0 denotes the initialization structure and $\hat{c}$ denotes the optimized structure) during optimization in Figure 4.3.

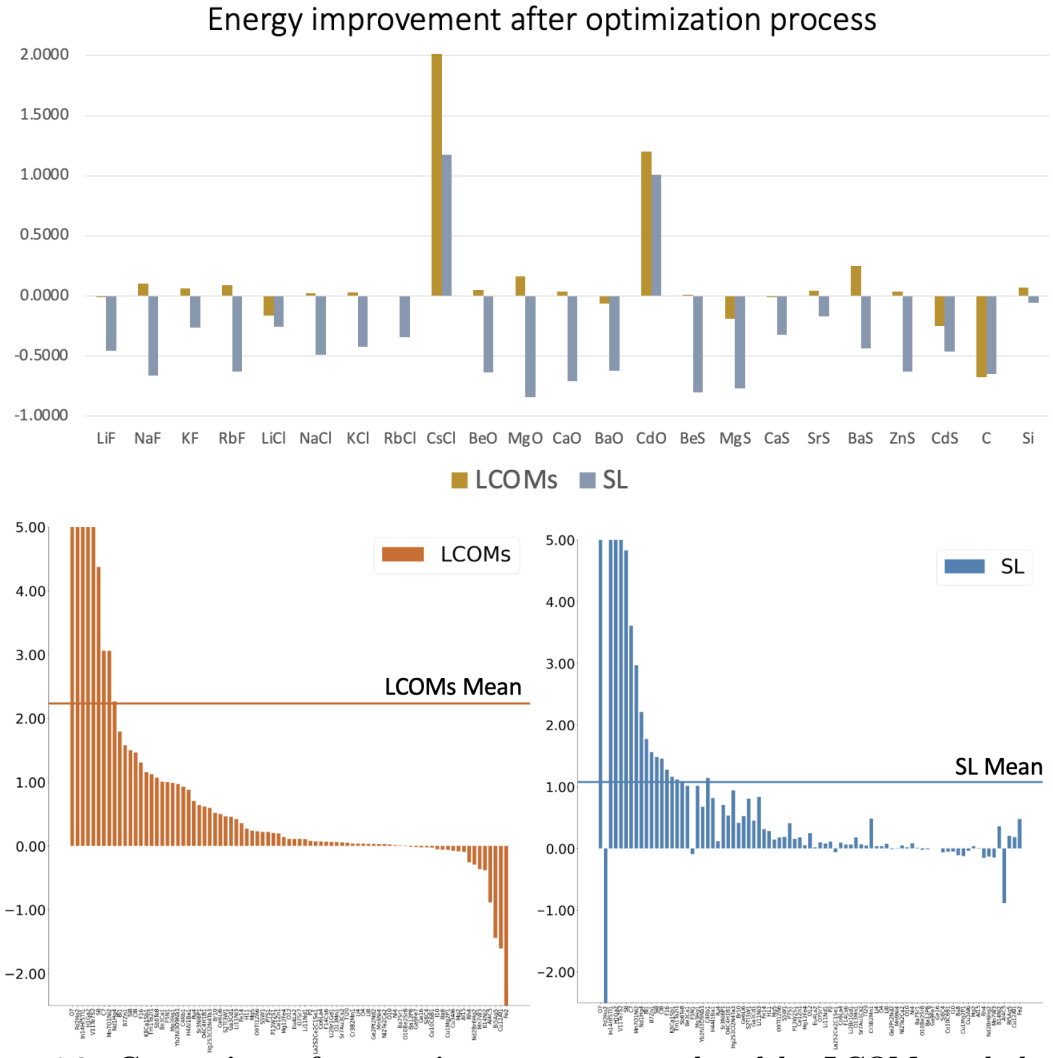


Figure 4.3: Comparison of energy improvements produced by LCOMs and the non-conservative supervised learning (SL) baseline. Top: Energy improvement for 25 compositions, when training on the OQMD dataset; **Bottom:** energy improvement over a set of 83 chemical compositions, when training on the MatBench dataset. Note that structures found by LCOMs achieve better formation energy after optimization, while the non-conservative supervised learning model actually leads to structures with worse energy values (negative improvement). The quantity on the y-axis represents the percentage of improvement (reduction) in the energy value. These results indicate that conservative training is essential for successfully instantiating a method with gradient-based latent space optimization for crystal structure prediction.

Observe that while LCOMs generally produces positive improvement, the non-conservative model leads to *negative* improvement: the optimized structures are generally worse than the random structure at initialization. This indicates that conservative training is critical for latent space optimization to work. We also perform a more fine-grained analysis, where we plot the trajectory of evolution of the energy values over each round of optimization in Figure 4.2. Observe that for the non-conservative model (orange), the energy increases over the course of optimization indicating that the optimizer is exploiting errors in the learned model. This exploitation is absent for LCOMs (in blue), indicating that conservatism is crucial for attaining good performance.

4.7 Discussion, Future Directions, and Limitations

We presented a method for offline optimization that uses the latent space of a CD-VAE to perform smooth gradient-based optimization of complex structures, with application to crystal structure prediction. Our method combines concepts from conservative objective models that robustify predictive models to make them amenable to gradient-based optimization, with generative models of graphs, which provide us with a latent space over crystal structures that overcomes the complex geometry of the design space, enabling us to use simple gradient-based optimization methods. Experiments show that our method can successfully optimize the formation energy and recover the optimal structure of a chemical compound with a good level of accuracy, comparing favorably with existing approaches, while tremendously reducing computation time. A limitation of our work and an interesting avenue for future work is to study the efficacy of LCOMs in more problems in computational chemistry. Another direction for future work is to use the best performing models to predict the optimized structure for novel chemicals and validate the predictions experimentally.

Chapter 5

Designing Cell Type-Specific Promoter Sequences via Conservative Model-Based Optimization

5.1 Introduction

Gene therapies treat diseases through the delivery and expression of therapeutic genetic cargo in disease-associated cells and tissues. Expression of the genetic cargo is controlled by its promoter sequence, a regulatory DNA sequence that is placed upstream of the coding region. To increase the effectiveness of gene therapy and reduce side-effects, the promoter needs to differentially induce expression in targeted cells while repressing expression in all other cells i.e. it needs to be cell type-specific (we will use cell type-specific promoter and differentially expressed promoter interchangably in this paper). Although the human body consists of over 400 types of cells [111], very few cell type-specific promoters have been discovered. Conventional promoter design techniques rely heavily on manual curation and involve tiling known cis-regulatory elements (CREs) or transcription factor (TF) binding motifs [74, 79, 131]. These techniques are difficult to automate and are not guaranteed to work, especially in less studied cell types.

An alternate *data-driven* paradigm for promoter design could significantly accelerate promoter discovery and improve the effectiveness of gene therapy. Specifically, if we can obtain reasonably accurate predictors for promoter-driven gene expression from data, we can simply optimize for the promoter sequence against this learned surrogate model.

Building on this insight, in this paper, we develop sequence-based learned predictors of promoter-driven expression and use them in conjunction with offline model-based optimization (MBO) algorithms to automate the design of cell-type specific promoters. While this sort of a paradigm is feasible in principle, for most cell types, very little promoter-driven expression data is available. To circumvent this challenge, we can borrow inspiration from supervised learning methods and first pre-train a model on diverse promoter-driven expression datasets, followed by fine-tuning to produce accurate models of promoter-driven expression [89]. However, despite the good in-distribution generalization of these approaches, models can perform poorly on out-of-distribution sequences, making them unsuitable for direct optimization. Moreover, to effectively use experimental validation resources, designed promoters need to be diverse so as to account for incorrect predictions.

The main contribution of this paper is a model-based optimization approach for designing promoter sequences, that we call conservative promoter design (CPD). Our approach first pre-trains a model for promoter-driven expression using large existing promoter-driven expression datasets and then fine-tunes this model using limited amount of data available for a set of target cell types. Then, it utilizes conservative optimization to optimize designs against this learned surrogate model. Our *in silico* results show that CPD is able to design highly diverse cell type-specific promoters reliably across different target cell types, significantly outperforming prior methods.

5.2 Preliminaries of Offline Model-Based Optimization

In this section we provide some background on offline model-based optimization methods. Typically, the goal in offline data-driven model-based optimization is to produce designs that maximize some objective function, using experience from a provided static dataset.

Offline model-based optimization. In the specific setting of promoter design, experimentation is quite expensive and time-consuming, hence the only viable approach is an offline approach, that can convert a dataset of historical measurements into an effective design. In the setting of offline MBO, an algorithm is provided access to a static dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ of designs $\mathbf{x}_i$ and a corresponding measurement of the objective value y_i. The algorithm consumes this dataset and produces an optimized candidate design $\mathbf{x}^*$ which is evaluated against the true objective function.

The offline MBO problem often involves learning a proxy objective f_θ mapping input sequence to measured property of interest, which we hereafter refer to as the **design model**. Optimization is then performed to find an input which maximizes this learned design model: $\mathbf{x}^* = \arg\max_{\mathbf{x}} f_\theta(\mathbf{x})$. In applying MBO to biological data, the workflow

may resemble: (1) collect data (experimentally or from publicly available sources); (2) train the model (or fine-tune a pre-trained model) via supervised learning; (4) perform optimization on the learned model with respect to its inputs, such as by *in silico* directed evolution or other classes of discrete optimization methods; and (5) evaluate the optimized designs using multiple oracle models. An **oracle model** is a model which (ideally) approximates the ground truth for evaluating new designs, following prior work [12, 114].

Distribution shift in offline MBO and conservative regularization. While we can train the design model $f_\theta(\mathbf{x})$ on a broad training dataset, the design model may still suffer from generalization failures common to supervised regression models. To computationally explore new regions of the sequence space, it is intuitive to move further away from the data distributions already experimentally explored to create more sequence diversity, but we will run into greater risk of model inaccuracy. Optimizing the promoter against the output of such a prediction model may still produce promoter sequences that are invalid and unstable, especially when trying to start the optimization from already good promoters. To alleviate this issue, we suggest learning a *conservative* model of the objective function, that is trained via a regularized supervised regression procedure following the COMs method of Trabucco et al. [115]. This conservative regularizer penalizes the value of the design model on unseen and potentially invalid promoters $\mu(\mathbf{x})$ that appear promising under the learned model $f_\theta(\cdot)$, preventing the discrete optimizer from designing promoters which appear promising under the learned design model, but do not actually perform well.

$$\min_{\theta} \quad \underbrace{\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[(f_\theta(\mathbf{x}) - y)^2\right]}_{:= \text{ supervised loss}} + \alpha \underbrace{\left(\mathbb{E}_{\mathbf{x}\sim\mu}\left[f_\theta(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[f_\theta(\mathbf{x})\right]\right)}_{:= \text{ conservative regularizer}}. \tag{5.1}$$

While explicit conservative regularization may not be needed when optimizing a promoter sequence sampled uniformly at random from the training dataset only upto a few mutations, we find in our experiments that an adequate amount of conservatism is needed if we wish to improve the high-scoring promoter sequences observed in the data.

5.3 Problem Setup

As mentioned in Section 5.1, it would be very beneficial for effective gene therapy to have a data-driven method that designs cell type-specific promoters in many settings. This work describes a COMs-based [115] method to address this need. To evaluate our promoter design method, we use it to design cell type-specific promoters for three cell lines - Jurkat, K-562 and THP-1. For any given target cell $tc \in \{\text{Jurkat, K-562, THP-1}\}$,

we define the differential expression induced by a promoter sequence $\mathbf{x}$ (i.e. its cell type-specificity) as:

$$\mathrm{DE}^{tc}(\mathbf{x}) = e^{tc}(\mathbf{x}) - \frac{1}{2}\sum_{oc \neq tc} e^{oc}(\mathbf{x}) \tag{5.2}$$

where $e^c(\mathbf{x}), c \in \{\text{Jurkat, K-562, THP-1}\}$ is the experimentally measured expression value induced by $\mathbf{x}$ in cell c. Our goal is to design sequences that maximize DE^{tc}.

The design models are trained to predict $e^c(\mathbf{x})$ in all three cell lines simultaneously using multi-task learning (MTL). We denote their predictions by $e^c_\theta(\mathbf{x})$ and the predicted differential expression DE^{tc}_θ can be computed using the individual expression predictions. Since we do not have access to experimentally measured expression values, our offline MBO method aims to design sequences that maximize DE^{tc}_θ while using the conservative regularizer during the training of the design models. As described in the previous section, using the regularizer increases the likelihood of finding sequences that also maximize DE^{tc}.

We use the data, pre-training tasks, and the general model architecture from Reddy et al. [89] to get our design model (more details about the pre-training tasks and model are in Section 5.4). They collected promoter-driven expression data in Jurkat, K-562 and THP-1 for 17,104 sequences that are 250 base pairs (bp) long. These sequences consist of promoters of naturally differentially expressed genes, sequences generated by tiling motifs enriched in differentially expressed gene promoters, and promoters of constitutive genes. Additionally, promoters are experimentally validated in batches using reporter assays with batch sizes ranging from a few hundred to many thousands or even millions, depending on the assay. Thus, we need our design method to generate multiple cell type-specific promoters for each of the three cell lines. We aim to design promoters for the assay used by Reddy et al. [89] that can measure promoter-driven expression from 15,000-20,000 promoters and thus design 5,000 promoters for each cell line.

5.4 CPD: Conservative Promoter Design

In this section we describe our approach, CPD, for designing cell type-specific promoters purely from offline data. Our workflow consists of three stages: **(1)** pre-training a base model on large existing datasets, **(2)** fine-tuning the pre-trained model on our targeted dataset with conservative regularization, **(3)** generating differentially expressed sequences by balancing optimality and diversity.

5.4.1 Pre-Training on Promoter Driven Expression Datasets

Collecting a large dataset with experimental measurements of promoter-driven expression in multiple cell types is expensive and time-consuming. Therefore, it is highly desirable to leverage existing large datasets of promoter-driven expressions to learn relevant information. Inspired by the recent success of pre-training in deep learning models [22, 50, 16, 13], we first pre-train a base model on several existing promoter datasets, following the process of Reddy et al. [89].

Promoter-driven expression datasets. Reddy et al. [89] identified that pre-training on existing large-scale promoter-driven expression datasets from massively parallel reporter assays (MPRA) leads to significantly better modelling of smaller promoter-driven expression datasets. They used data from two MPRAs for pretraining - SuRE MPRA [119] and Sharpr-MPRA [28]. SuRE MPRA measures the expression induced by 150-500bp genomic fragments from 4 individuals from 4 different populations in the K-562 and HepG2 cell lines. $\sim$ 2.4B and $\sim$ 1.2B fragments were found to be expressed in K-562 and HepG2 respectively. Most fragments have very low expression and training on all measurements is time-consuming. Thus, Reddy et al. [89] define a classification task using this data that subsets the data and bins each sequence into one of 5 expression bins. We also use this classification task for pre-training. Sharpr-MPRA is a smaller scale MPRA that measures the expression from $\sim$ 487K 145bp sequences centered at DNase I peaks in K-562 and HepG2 cells and in two different settings. Reddy et al. [89] use a preprocessed version of this data from Movva et al. [78] that builds a regression task with 12 outputs (2 replicates for expression measured in 2 settings in 2 cell lines and 4 outputs that correspond to the average expression across replicates). We use the same formulation for pre-training our models.

Model architecture and pre-training objectives. For the model architecture, we follow the pre-training procedure of Reddy et al. [89]. Specifically, we use a hybrid of 1D convolution and transformer [121] network. Before feeding the DNA sequence into the network, we first encode it into a sequence of one-hot embedding vectors. The embedding vectors are then fed into a stack of 1D convolution layers. After the convolution layers, we prepend a learnable CLS token [22] to the embedding vectors and feed them into a stack of transformer blocks, where we apply rotary positional embeddings [110] in each attention layer. We provide details of our architecture in Appendix D.2.1.

The output layer produces the probability of a sequence belonging to each of the expression bins for the SuRE MPRA-based pre-training task and directly predicts the expression values for the Sharpr-MPRA-based pre-training task. We minimize the sum of the negative log-likelihood (NLL) loss for the SuRE MPRA task and the mean squared error (MSE) loss for the Sharpr-MPRA task (since both tasks have distinct sequences, a training sequence only contributes to one of the two loss terms, the other loss is set to zero for that sequence).

5.4.2 Finetuning on Target Cell Type Dataset with Conservatism

After pre-training the model on existing datasets of promoter driven gene expression, we finetune it on the datasets we collect for the target cell types in order to form a design model. In order to have the model output the three expression levels for the target TPH-1, Jurkat and K562 cell types, we discard the output layer of the pre-trained model and attach three randomly initialized output heads, where each head consists of a multilayer perceptron (MLP) with 2 hidden layers and 512 units width. As described in Section 5.2, since we will be optimizing against the model to design promoter sequences, it is imperative to address the distribution shift problem, so we employ the COMs [115] regularization on top of our supervised fine-tuning objective. Let us denote the model prediction for expression level in cell type i as $f^i_\theta(\mathbf{x})$, then the overall fine-tuning objective can be written as:

$$\min_\theta \sum_i \left\{ \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[(f^i_\theta(\mathbf{x}) - y_i)^2 \right] + \alpha \left(\mathbb{E}_{\mathbf{x}\sim\mu} \left[\mathrm{DE}^i_\theta(x) \right] - \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\mathrm{DE}^i_\theta(x) \right] \right) \right\} \tag{5.3}$$

where $i \in \{\text{Jurkat, THP-1, K562}\}$ indicates the cell types and $\mathrm{DE}^i_\theta(x) := f^i_\theta(\mathbf{x}) - \frac{1}{2}\sum_{j\neq i} f^j_\theta(\mathbf{x})$ is the design model predicted difference in expression levels between the target cell type i and two other cell types. In contrast to the canonical formulation of COMs in Eqn 5.1, we push up and push down the difference of expression levels instead of just the expression levels since our goal here is to maximize the difference. For the unseen, overestimated promoter distribution $\mu(\mathbf{x})$, we use a gradient ascent optimizer and perform T steps of gradient ascent on $\mathrm{DE}^i_\theta(\mathbf{x})$ starting from a promoter sequence in the dataset, similar to that in COMs [115]. Since the DNA sequences are discrete, we perform the optimization in the probability simplex in the one-hot encoded space, paramterized by a softmax function. At the end of T steps of gradient ascent, we perform a hard clipping so that the resulting sequence is a valid one-hot encoding.

5.4.3 Balancing Optimality and Diversity in Promoter Design

After fine-tuning the model, we run the gradient ascent optimizer on the differential expression values predicted by the model to generate the design candidates. Specifically, starting from each promoter sequence in the dataset, we perform T steps of gradient ascent optimization to get an optimized sequence. This process gives us a large pool of design candidates. As discussed in Section 5.3, in our problem, we *simultaneously* evaluate the expression of multiple design candidates (say, K) at once. This is unlike typical offline model-based optimization problems where an algorithm must produce only a single design to be evaluated. In this section, we present an approach that

prescribes a way to select these K candidate sequences to enhance the performance of the COMs approach discussed above.

Our idea is based on the simple observation: the performance of an MBO method that is allowed to produce multiple optimized designs can be enhanced in one of two ways: **(i)** by ensuring that there exists one output design which attains a high value of the ground-truth objective, and **(ii)** by ensuring that the optimized set of designs covers as big of a volume in design space as possible, so that even if the best possible design produced by underlying MBO method is not actually optimal under the ground-truth objective, with high probability the output set still covers a good design by virtue of covering the space. That is, baking in **(i)** and **(ii)** into a strategy for selecting a subset of size K enables us to obtain the best possible candidate set given our constraints.

To instantiate this intuition into a concrete strategy, we construct a set of designs, of size K, which consists of designs which attain high objective values and cover a space as wide as possible, under a given domain-specific distance metric $D(\cdot, \cdot)$. Formally, this can be written as selecting a set of sequences $\mathcal{S}^* = \{\mathbf{x}_1^*, \mathbf{x}_2^*, \cdots, \mathbf{x}_K^*\}$ of size K such that $\mathcal{S}^*$ is the optimal solution to the following optimization problem:

$$\mathcal{S}^* := \arg\max_{\mathcal{S}} \quad \sum_{\mathbf{x} \in \mathcal{S}} \widehat{f}_\theta(\mathbf{x}) + \beta \sum_{\mathbf{x} \in \mathcal{S}, \mathbf{x}' \in \mathcal{S}} D(\mathbf{x}, \mathbf{x}'), \tag{5.4}$$

where $\widehat{f}_\theta(\mathbf{x})$ denotes a pessimistic estimate of the objective function. While in theory, we can obtain $\mathcal{S}^*$ by optimizing over the entire design space, in practice, doing so is computationally intractable, so the optimization in Equation 5.4 is usually performed from within a significantly larger set of design candidates observed by the conservative model over the course of training. To avoid the over-estimation bias in the pessimistic estimate, $\widehat{f}_\theta$ in Equation 5.4 due to correlations between the candidate sequences we wish to sub-sample from and the conservative design model, in practice, we use the mean estimate from an ensemble of "validation" models to estimate the objective function in Equation 5.4. Crucially, this ensemble is trained on a different bootstrap of the training data, ensuring no correlation between the ensemble and the larger set of design candidates produced by optimizing our conservative design model.

Theoretical motivation. We will now intuitively discuss how the aforementioned approach of balancing optimality and diversity from Equation 5.4 can be viewed under certain assumptions, as an intermediate strategy for constructing a subset $\widehat{\mathcal{S}}$ of candidate designs that attains a high performing best design. To understand how, we begin expanding the formula for the best performing design from within a subset $\mathcal{S}$ of designs. We will denote the ground-truth objective function as f^* and the learned function as $\widehat{f}$, and utilize the modelling assumption that the potentially infinite-dimensional vector of values $\widehat{f}$ is normally distributed around the ground-truth function f^* with covariance $\widehat{\Sigma}$, i.e., $\widehat{f} \sim \mathcal{N}(f^*, \widehat{\Sigma})$. Then, the performance of the subset $\mathcal{S}$ is given by:

Method	Best			90th percentile			Median			Mean		
	THP-1	Jurkat	K562	THP-1	Jurkat	K562	THP-1	Jurkat	K562	THP-1	Jurkat	K562
CPD (Ours)	**0.86**	**1.22**	**1.31**	0.59	**1.13**	**1.20**	0.44	**1.06**	**1.10**	0.45	**1.05**	**1.10**
DENs	0.71	**1.24**	1.06	**0.65**	**1.14**	0.92	**0.59**	0.99	0.71	**0.58**	0.96	0.67
Naïve grad. asc.	0.81	1.09	**1.31**	0.54	0.96	1.13	0.38	0.86	1.03	0.38	0.85	1.03
Motif tiling	0.53	1.12	1.01	0.31	0.90	0.55	-0.02	0.67	0.18	-0.12	0.61	0.21
Best from data	0.57	1.13	1.10	0.42	0.67	0.61	0.25	0.16	0.25	0.15	0.21	0.26

Table 5.1: The performance of CPD and baselines for designing promoters with differential expression levels on the three cell types. We report the best, the 90th percentile, the median and mean of the 5000 designed sequences from each method. We see that CPD robustly optimize the differential expression levels for three different target cell types, achieving good performance across different metrics. We see that CPD outperforms other methods reliably across three cell types.

$$\mathbb{E}_{\mathcal{S}\sim\text{Alg}(\widehat{f})}\left[\max_{\mathbf{x}\in\mathcal{S}} f^*(\mathbf{x})\right] := d^{*\top}\left(\widehat{h}_{\mathcal{S}}\cdot f^*\right) \tag{5.5}$$

$$:= d^{*\top}\left(\widehat{h}_{\mathcal{S}}\cdot\left(\widehat{f}+f^*-\widehat{f}\right)\right) \tag{5.6}$$

Now since by construction, $\widehat{f}\sim\mathcal{N}(f^*,\widehat{\Sigma})$, we can express $\widehat{f}=f^*+\widehat{\Sigma}\varepsilon$, where $\varepsilon\sim\mathcal{N}(0,I)$, we can replace the difference $\widehat{f}-f^*$ in as follows:

$$\mathbb{E}_{\mathcal{S}\sim\text{Alg}(\widehat{f})}\left[\max_{\mathbf{x}\in\mathcal{S}} f^*(\mathbf{x})\right] := d^{*\top}\left(\widehat{h}_{\mathcal{S}}\cdot\left(\widehat{f}-\widehat{\Sigma}\varepsilon\right)\right). \tag{5.7}$$

Since ε is a random variable that is correlated with the operator $\widehat{h}_{\mathcal{S}}$, we will need to use a uniform concentration argument over the randomness in ε to lower-bound the RHS of Equation 5.7. This lower bound implies that in order to maximize the LHS (i.e., the expected value of the highest performing sample), we must select a subset $\mathcal{S}$ that consists of candidate designs which **(i)** maximize an estimate of the value of the ground-truth function $\widehat{f}$ and **(ii)** minimize the top eigenvalue, $\lambda_{\max}(\widehat{\Sigma})$. While **(ii)** is computationally expensive, an easy heuristic approach to attain a similar benefit is to minimize the Frobenius norm of the matrix $\widehat{\Sigma}$. Since $\widehat{\Sigma}(\mathbf{x},\mathbf{y})=D(\mathbf{x},\mathbf{y})^{-1}$, we can choose to maximize pairwise distances within the subset $\mathcal{S}$, which is essentially what our strategy in Equation 5.4 attempts to achieve.

5.5 Experiments

The goal of our experiments is to understand the efficacy of our approach towards designing cell-specific promoter sequences. To this end, we quantitatively and qualitatively analyze the promoter sequences designed by our approach, while comparing them to those obtained using baseline design methods. We first discuss the prior approaches that we compare to in Section 5.5.1. Then, we quantitatively analyze the sequences

for their predicted differential expression in Section 5.5.2 and determine the effect of various design parameters on the designed sequences in 5.5.3. Finally, we qualitatively analyze the motif composition of the designed sequences in Section 5.5.4.

5.5.1 Prior Approaches and Baselines

We compare our approach, CPD, to three other baselines that we describe below:

Best in dataset. To verify that our model-based optimization approach indeed finds designs better than the best in the dataset, we consider a baseline approach that simply takes the top 5000 most differentially-expressed sequences for each target cell type based on the dataset measured expression.

Tiling TF-binding motifs enriched in existing differentially expressed sequences. We also compare to a traditional method to design cell type-specific promoters that aims to find TF-binding motifs that are enriched in some known set of differentially expressed sequences or are known to be binding motifs for differentially expressed TFs and tile these motifs, hoping that they are responsible for inducing differential expression [79, 131]. We run this method for the three chosen cell lines - Jurkat, K-562 and THP-1. We present the details of this method in Appendix D.2.4.

Sequence optimization using design models trained without conservatism. Another related prior approach for offline MBO is known as naïve gradient descent [114], where we train a model to predict the objective value and use gradient descent method to optimize the trained model with respect to the sequence. This is equivalent to COMs without the conservatism regularization. Similar to COMs, we use the softmax parameterization and perform T steps of gradient descent starting from each sequence in the dataset, as described in Section 5.4.

Deep exploration networks (DENs). We also compare to the DENs approach [71] which aims to specifically design biological sequences. To the best of our knowledge, DENs are the only offline model-based optimizers whose designed sequences have been experimentally validated. Concretely, given a prediction model, DENs train a sequence generative model to produce samples that maximize the output of the prediction model as well as maintaining a desired level of diversity for the distribution of generated samples. We provide the details of this approach in Appendix D.2.5.

5.5.2 Main Results

For each prior approach described in the previous section as well as CPD, we produce 5000 optimized designs and evaluate them according to our oracle models. We compare the final differential expression levels and present the results in Table 5.1. We observe that CPD reliably outperforms prior methods across the three different cell lines. To see how the expression levels of designed sequences differ from that of the sequences in

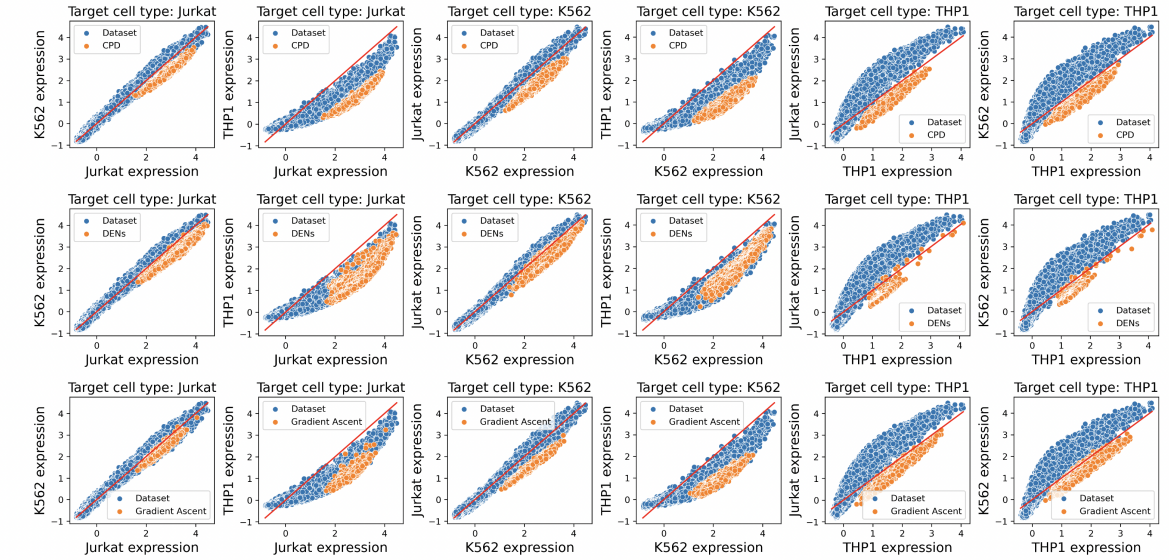


Figure 5.1: comparison of oracle predicted expression levels of designed sequences on different cell types. We visualize the expression levels on the target cell type we optimize against versus the other cell types. We see that while all methods are able to improve the differential expression levels over the average of the dataset, CPD can push the objective beyond the best of the dataset reilably across three target cell types.

the dataset, we visualize the distribution for CPD, DENs and naïve gradient ascent in Figure 5.1. We can see that across all target cell types, CPD is able to produce a optimized pool of sequences which enhances the differential expression beyond the dataset. Specifically, for K562 cell type, DENs and naïve gradient ascent are unable to produce sequences that has differential expression levels against the THP-1 cell type beyond the dataset, while CPD extend beyond the edge of the dataset. For THP-1, because the expression level is naturally low in the dataset, DENs struggles to find better sequences so the generator collapse onto a few modes, while CPD is able to produce a diverse set of sequences with high differential expression level.

Additionally, we analyze the diversity between sequences in the designed pool by computing the mean base pair entropy at every position and mean edit distance between sequences in the pool. We present the result in Table 5.2. The diversity suggests that CPD produces highly diverse sequences, matching the diversity of the best sequences in the dataset. As discussed before, DENs struggles to find diverse sequences for THP-1 target cell type.

5.5.3 Ablation Studies

In this section we provide ablation experiments to verify the efficacy of two important aspects of CPD: **(i)** the use of a conservative objective instead of a non-conservative, standard supervised regression objective, and **(ii)** balancing diversity and optimality

Method	Mean base pair entropy			Mean edit distance		
	THP-1	Jurkat	K562	THP-1	Jurkat	K562
CPD (Ours)	1.96	1.87	1.96	184.25	176.17	183.76
DENs	0.93	1.87	1.86	101.41	175.68	175.58
Naïve grad. asc.	1.96	1.92	1.88	184.07	181.00	176.69
Motif tiling	1.92	1.89	1.92	180.73	178.90	180.86
Best in data	1.97	1.94	1.94	185.06	182.74	182.30

Table 5.2: Quantifying the diversity of designed sequences with mean base pair entropy and edit distance. The results suggest that CPD is able to produce highly diverse sequences, matching the diversity of the best sequences in the dataset, while DENs struggles on THP-1 and collapse onto a few modes.

during sequence selection. We first examine the necessity of the conservative objective, and compare the results of CPD under different conservatism coefficients in Table 5.3. Our results suggest that having higher level conservatism is crucial to CPD as the higher conservatism variant almost uniformly outperforms the baseline with no conservatism. Further more, we see that with our ensemble sequence selection strategy is able to combine the best design from all α hyperparamters.

CPD α	Best			90th percentile			Median			Mean		
	THP-1	Jurkat	K562	THP-1	Jurkat	K562	THP-1	Jurkat	K562	THP-1	Jurkat	K562
0	0.81	1.09	1.31	0.54	0.96	1.13	0.38	0.86	1.03	0.38	0.85	1.03
$3e-4$	0.86	1.13	1.31	0.52	0.98	1.15	0.35	0.88	1.05	0.35	0.87	1.05
$1e-3$	0.78	1.18	1.29	0.46	1.04	1.17	0.27	0.94	1.08	0.27	0.93	1.07
$3e-3$	0.74	1.22	1.31	0.35	1.11	1.18	0.16	1.03	1.08	0.16	1.02	1.07
$1e-2$	0.58	1.21	1.28	0.27	1.10	1.15	0.09	1.02	1.04	0.10	1.02	1.03
$3e-2$	0.52	1.18	1.21	0.25	1.04	1.02	0.10	0.95	0.90	0.10	0.95	0.89
Combined	0.86	1.22	1.31	0.59	1.13	1.20	0.44	1.06	1.10	0.45	1.05	1.10

Table 5.3: Ablation study of the conservative coefficient α in CPD

We also perform an ablation study for our diversity enhanced sequence selection strategy, and compare the effects between different diversity coefficient β during sequence selection. We present the results in Table 5.4. The results suggest that CPD is not sensitive to the diversity coefficient, as CPD is able to produce diverse sequences even without explicit encouraging diversity ($\beta = 0$). This result is not surprising, since unlike a generative model, CPD starts the optimization from different staring sequences and therefore finds different local minima of the objective landscape. In comparison, the diversity driven selection is crucial for DENs as DENs sequences tend to cluster to a few local minima due to the mode collapse phenomenon in generative models. This means that utilizing our diversity metric for sequence selection is crucial for obtaining good performance when the underlying optimizer does not naturally find high entropy solutions.

Method	Mean base pair entropy			Mean edit distance		
	THP-1	Jurkat	K562	THP-1	Jurkat	K562
CPD: $\beta = 0.0$	1.96	1.87	1.96	184.25	176.17	183.76
CPD: $\beta = 0.1$	1.96	1.87	1.96	184.25	176.17	183.77
CPD: $\beta = 0.3$	1.96	1.87	1.96	184.27	176.18	183.77
CPD: $\beta = 1.0$	1.96	1.87	1.96	184.30	176.46	183.90
CPD: $\beta = 3.0$	1.96	1.88	1.96	184.41	177.05	184.14
CPD: $\beta = 10.0$	1.97	1.90	1.97	184.80	179.01	184.78
DENs: $\beta = 0.0$	0.93	1.78	1.54	101.41	169.04	151.44
DENs: $\beta = 0.1$	0.93	1.78	1.54	101.41	169.04	151.67
DENs: $\beta = 0.3$	0.93	1.79	1.55	101.41	169.32	152.33
DENs: $\beta = 1.0$	0.93	1.79	1.58	101.41	169.83	154.52
DENs: $\beta = 3.0$	0.93	1.82	1.67	101.41	172.03	161.17
DENs: $\beta = 10.0$	0.93	1.87	1.86	101.41	175.68	175.58

Table 5.4: Quantifying the diversity of designed sequences with mean base pair entropy and edit distance. We see that CPD is able to produce diverse sequence without the need to artificially encourage diversity during sequence selection.

5.5.4 Motif composition of designed sequences

Ultimately, the efficacy of a promoter design method would be dictated by whether or not the designed promoter enables differential expression in real experiments. As we did not have access to experimental validation resources, we seek to verify the potential biological validity of the designed sequences by analyzing whether they contain known TF-binding motifs. We use GimmeMotifs [14] to identify instances of the clustered TF-binding motifs defined by Vierstra et al. [122] in the sequences designed by the best performing methods, COMs, DENs, and naïve gradient ascent. When identifying motifs, we use a 0.01 false-positive rate cut-off. The most enriched motifs in these sequences are shown in Figure 5.2. It is clear that the designed sequences from all of the methods contain many known TF-binding motifs. More interestingly, there are many ZBTB-binding motifs in the sequences, especially in those designed for Jurkat cells. The ZBTB family of TFs are known to play a role in the development, differentiation and functioning of T cells [18], which Jurkat cells are derived from. These results suggest that the designed sequences contain many known TF-binding motifs, some of which are known to affect cell-type specific functions, which provides strong support for the biological validity of the designed sequences.

5.6 Conclusion

In this paper, we develop a data-driven approach to tackle the problem of cell-type specific promoter sequence design. By combining pre-training on diverse promoter-driven expression datasets with conservative offline model-based optimization techniques, our approach, CPD, is able to design novel and effective promoter sequences that enhance differential expression of different targets, by simply training on existing promoter

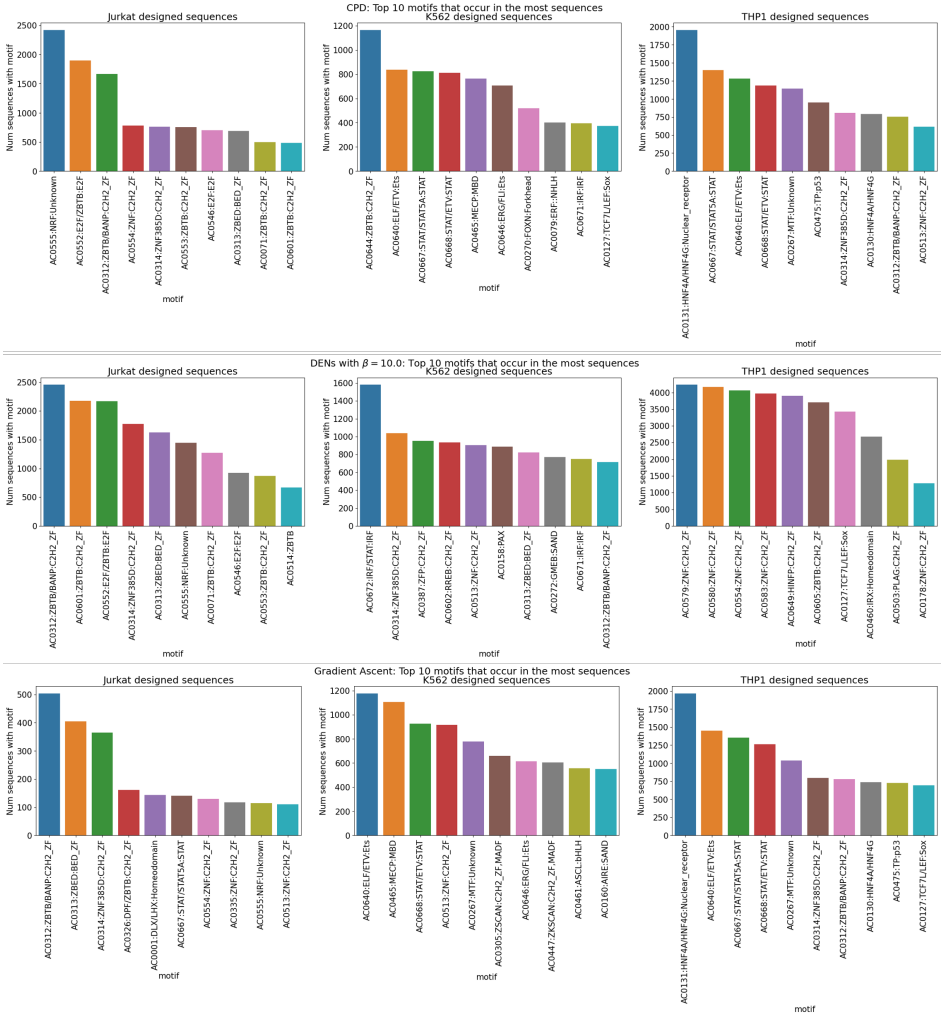


Figure 5.2: Motifs most enriched in sequences designed using various methods and the number of sequences in which they occur.

experimental data, without requiring any active "in-the-loop" experimentation. Our experiments show that CPD can produce promoters that not only enhance differential expression as predicted by oracle evaluation models but are also diverse enough, outperforming prior approaches. We also find that both conservatism and diversity are crucial and the promoter sequences designed by our method satisfy biological validity criteria.

While our results are promising, there are still some limitations of this work and quite a few avenues for future work. First, we note that our evaluation protocol is limited as it does not involve actual experimentation. Even though our ablations study a variety of properties of the designed sequences and finds positive results, we believe that actual experimentation to synthesize the proposed design candidates will be crucial in demonstrating the efficacy of this method. Second, we believe that running our offline

design approach in conjunction with a few rounds of in-the-loop experimentation will likely produce much better sequences, that attain significant practical improvements. Finally, combining advances in self-supervised pre-training and generative modeling for sequences together with our conservative approach will likely yield better results and hence, is an interesting avenue for future work.

Chapter 6

Conclusion

In this thesis, we study the problem of offline model-based optimization (MBO) which is ubiquitous in a wide variety of scientific and engineering domains. We begin by giving it a formal definition and identify the core challenges associated with its real-world instances. With these core challenges in mind, we select a suite of realistic tasks from a wide variety of disciplines to build Design-Bench, a benchmark for offline MBO with a standardized evaluation protocol. Using our benchmark, we evaluate and fairly compare prior methods in offline MBO, and discover that no single method outperforms others in all tasks, indicating there is still a large room for improvement in offline MBO methods.

When evaluating prior methods in Design-Bench, we discover that many existing methods make use of generative models or explicit density estimation to capture the dataset distribution, which severely limit their performance across different tasks. Generative models or density models can be difficult to train and often requires tuning for each modality of data. To address this limitation, we propose conservative objective models (COMs), a simple but effective offline MBO method that does not require training a generative or density model. Instead of directly capturing the dataset distribution, COMs rely on conservative regularization of the prediction model, which makes it easy and stable to train. We demonstrate that COMs outperforms prior methods in a variety of tasks in Design-Bench.

To verify that COMs works not only on benchmarks but also in real-world problems, we present two applications of COMs in the final part of the thesis. We propose latent conservative models (LCOMs) for the problem of crystal structure prediction in computational chemistry, and apply COMs to the problem of differentially expressive promoter DNA sequence optimization in synthetic biology. Through these two applications, we demonstrate that our offline MBO methods can be successfully applied in real-world problems in various domains.

Throughout our study of real-world offline MBO problems, we have also identified some

important limitations in conservative offline MBO methods. Throughout this work, one major assumption we make in offline MBO is that it is easy to learn a prediction model for objective function via supervised learning. While this is true for many common modalities of data where researchers have already developed good prediction models, supervised prediction are still open problems in many specialized domains, and it is not clear how to apply offline MBO methods when supervised prediction models cannot be reliably trained. Another important challenge is the integration of domain knowledge into learned models. While deep learning models can be accurate with sufficient training data, in many scientific and engineering problems, the data is fairly limited and practitioners often rely on domain knowledge to make good decisions. Due to the black-box nature of neural networks, sometimes it can be a challenge to integrate domain knowledge into these learn model.

Like many other research programs, our journey in offline MBO also raises many questions for future research. We conclude this thesis by discussing some promising directions we've identified in offline MBO:

- **Self-supervised pre-training in offline MBO.** In many real-world problems of offline MBO, collecting labeled data is often very expensive as it involves physical experiments. However, in many settings, it is easy to collect a large amount of unlabeled data. For example, in synthetic biology, while it is expensive to collect a protein sequence dataset with certain property measured, it is very easy to collect a dataset of unlabeled protein sequences. Recently, researchers in computer vision and natural language process have introduced a wide variety of self-supervised learning methods [22, 13, 16, 50], which can learn expressive embeddings that enables sample efficient finetuning for downstream tasks. Finding effective methods of self-supervised pre-training in offline MBO has the potential of making model training a lot more data-efficient, therefore making it much easier to apply offline MBO methods in small data domains.
- **Theoretical analysis of compositionality in offline MBO.** In Chapter 2, we introduce the concept of compositionality, and how compositionality makes it possible to find designs better than the best one in the dataset. While the concept it intuitive, it would be highly desirable to provide some theoretical treatments of this subject, where one provides a formally definition and theoretical guarantees for the possibilities of improving beyond the datset.
- **Software toolkit for offline MBO.** While our method, COMs, significantly reduces the tuning required compare to prior methods, applications of offline MBO still requires a significant amount of machine learning background and practical experience, which limits its applications in many disciplines where practitioners are not familiar with machine learning. Therefore, it is highly desirable to build an easy-to-use software toolkit for offline MBO to facilitate the applications of offline MBO methods in more domains.

Even though the exact benchmarks or methods we propose in this thesis might not prove to be useful eventually, we hope that the work we present in this thesis can bring in some insights into the problem and incite future research in methodology and applications of offline MBO.

Bibliography

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021. 17

[2] 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 *Conference on Robot Learning (CoRL)*, 2019. 11

[3] Christof Angermueller, David Belanger, Andreea Gane, Zelda Mariet, David Dohan, Kevin Murphy, Lucy Colwell, and D Sculley. Population-based black-box optimization for biological sequence design. *arXiv preprint arXiv:2006.03227*, 2020. 30, 103, 106

[4] Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HklxbgBKvr`. 30, 103

[5] Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HklxbgBKvr`. 83

[6] Christof Angermüller, David Belanger, Andreea Gane, Zelda Mariet, David Dohan, Kevin Murphy, Lucy Colwell, and D. Sculley. Population-based black-box optimization for biological sequence design. 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*, pages 324–334. PMLR, 2020. URL `http://proceedings.mlr.press/v119/angermueller20a.html`. 83, 90, 91

[7] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: Programmable bayesian optimization in pytorch. *CoRR*, abs/1910.06403, 2019. URL `http://arxiv.org/abs/1910.06403`. 16, 95

[8] 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, et al. Survey of variation in human transcription factors reveals prevalent dna binding changes. *Science*, 351(6280):1450–1454, 2016. 11, 32, 83, 86, 102, 105

[9] Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496. IEEE, 2016. 20

[10] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 10, 11

[11] David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019. URL `http://proceedings.mlr.press/v97/brookes19a.html`. 46

[12] David H Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. *arXiv preprint arXiv:1901.10060*, 2019. 2, 5, 7, 9, 10, 15, 16, 17, 20, 22, 25, 30, 31, 32, 35, 54, 90, 91, 94, 102, 103

[13] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 56, 67

[14] Niklas Bruse and Simon J van Heeringen. Gimmemotifs: an analysis framework for transcription factor motif analysis. *BioRxiv*, page 474403, 2018. 63

[15] Lowik Chanussot, Abhishek Das, Siddharth Goyal, Thibaut Lavril, Muhammed Shuaibi, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, et al. Open catalyst 2020 (oc20) dataset and community challenges. *Acs Catalysis*, 11(10):6059–6072, 2021. 46

[16] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 56, 67

[17] Guanjian Cheng, Xin-Gao Gong, and Wan-Jian Yin. Crystal structure prediction by combining graph network and optimization algorithm. *Nature Communications*, 13(1):1492, 2022. 38, 41, 42, 46, 47, 109, 110

[18] Zhong-Yan Cheng, Ting-Ting He, Xiao-Ming Gao, Ying Zhao, and Jun Wang. Zbtb transcription factors: key regulators of the development, differentiation and effector function of t cells. *Frontiers in Immunology*, 12:713294, 2021. 63

[19] H Chermette. Density functional theory: a powerful tool for theoretical studies in coordination chemistry. *Coordination chemistry reviews*, 178:699–721, 1998. 38

[20] Sayak Ray Chowdhury and Aditya Gopalan. Bayesian optimization under heavy-tailed payoffs. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 13790–13801, 2019. URL `http://papers.nips.cc/paper/9531-bayesian-optimization-under-heavy-tailed-payoffs`. 14

[21] Maurice Clerc. *Particle swarm optimization*, volume 93. John Wiley & Sons, 2010. 46

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 56, 67, 111

[23] Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. Show your work: Improved reporting of experimental results. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2185–2194. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1224. URL `https://doi.org/10.18653/v1/D19-1224`. 93

[24] Alexander Dunn, Qi Wang, Alex Ganose, Daniel Dopp, and Anubhav Jain. Benchmarking materials property prediction methods: the matbench test set and automatminer reference algorithm. *npj Computational Materials*, 6(1):138, 2020. 41

[25] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. 10

[26] J Enkovaara, C Rostgaard, J J Mortensen, J Chen, M Dułak, L Ferrighi, J Gavnholt, C Glinsvad, V Haikola, H A Hansen, H H Kristoffersen, M Kuisma, A H Larsen, L Lehtovaara, M Ljungberg, O Lopez-Acevedo, P G Moses, J Ojanen, T Olsen, V Petzold, N A Romero, J Stausholm-Møller, M Strange, G A Tritsaris, M Vanin, M Walter, B Hammer, H Häkkinen, G K H Madsen, R M Nieminen, J K Nørskov, M Puska, T T Rantala, J Schiøtz, K S Thygesen, and K W Jacobsen. Electronic structure calculations with gpaw: a real-space implementation of the projector augmented-wave method. *Journal of Physics: Condensed Matter*, 22(25):253202, jun 2010. doi: 10.1088/0953-8984/22/25/253202. URL `https://dx.doi.org/10.1088/0953-8984/22/25/253202`. 109

[27] Jussi Enkovaara, Nichols A Romero, Sameer Shende, and Jens J Mortensen. Gpaw-massively parallel electronic structure calculations with python-based software. *Procedia Computer Science*, 4:17–25, 2011. 42

[28] Jason Ernst, Alexandre Melnikov, Xiaolan Zhang, Li Wang, Peter Rogov, Tarjei S Mikkelsen, and Manolis Kellis. Genome-scale high-resolution mapping of activating and repressive nucleotides in regulatory regions. *Nature biotechnology*, 34(11): 1180–1190, 2016. 56, 112

[29] Clara Fannjiang and Jennifer Listgarten. Autofocused oracles for model-based design. *arXiv preprint arXiv:2006.08052*, 2020. 2, 5, 9, 10, 11, 15, 16, 17, 22, 30, 31, 32, 84, 93, 104

[30] 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*. OpenReview.net, 2021. URL `https://openreview.net/forum?id=FmMKSO4e8JK`. 30

[31] Justin Fu and Sergey Levine. Offline model-based optimization via normalized maximum likelihood estimation. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=FmMKSO4e8JK`. 2, 5

[32] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020. 7, 27

[33] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018. 30

[34] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural processes. *CoRR*, abs/1807.01622, 2018. URL `http://arxiv.org/abs/1807.01622`. 30

[35] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. *Advances in Neural Information Processing Systems*, 34:6790–6802, 2021. 38, 43

[36] Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012. 1, 9, 12, 13, 83, 86

[37] Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012. 13, 20

[38] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Found. Trends Mach. Learn.*, 8 (5-6):359–483, November 2015. ISSN 1935-8237. doi: 10.1561/2200000049. URL `http://dx.doi.org/10.1561/2200000049`. 46

[39] Colin W Glass, Artem R Oganov, and Nikolaus Hansen. Uspex—evolutionary crystal structure prediction. *Computer physics communications*, 175(11-12): 713–720, 2006. 45

[40] Rafael Gómez-Bombarelli, David Duvenaud, José Miguel Hernández-Lobato, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. In *ACS central science*, 2018. 21, 22

[41] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. NIPS'14, 2014. 9, 30, 32

[42] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 21, 23, 30

[43] Charles E Grant, Timothy L Bailey, and William Stafford Noble. Fimo: scanning for occurrences of a given motif. *Bioinformatics*, 27(7):1017–1018, 2011. 113

[44] Francesca Grisoni, Michael Moret, Robin Lingwood, and Gisbert Schneider. Bidirectional molecule generation with recurrent neural networks. *Journal of chemical information and modeling*, 2020. 46

[45] T. Haarnoja, Aurick Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018. 85, 105

[46] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018. 12

[47] Jürgen Hafner. Ab-initio simulations of materials using vasp: Density-functional theory and beyond. *Journal of computational chemistry*, 29(13):2044–2078, 2008. 41

[48] Kam Hamidieh. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154: 346 – 354, 2018. ISSN 0927-0256. doi: https://doi.org/10.1016/j.commatsci.2018.07.052. URL `http://www.sciencedirect.com/science/article/pii/S0927025618304877`. 4, 9, 11, 84, 87, 104, 106

[49] Nikolaus Hansen. The CMA evolution strategy: A comparing review. In José An-

tonio Lozano, Pedro Larrañaga, Iñaki Inza, and Endika Bengoetxea, editors, *Towards a New Evolutionary Computation - Advances in the Estimation of Distribution Algorithms*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer, 2006. doi: 10.1007/3-540-32494-1_4. URL https://doi.org/10.1007/3-540-32494-1_4. 15, 31

[50] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022. 56, 67

[51] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 111, 113

[52] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018. 86, 104

[53] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 10

[54] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 44

[55] Warren Hoburg and Pieter Abbeel. Geometric programming for aircraft design optimization. volume 52, 04 2012. ISBN 978-1-60086-937-2. doi: 10.2514/6.2012-1680. 20

[56] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018. 28, 101

[57] Scott M. Jordan, Yash Chandak, Daniel Cohen, Mengxue Zhang, and Philip S. Thomas. Evaluating the performance of reinforcement learning algorithms. *CoRR*, abs/2006.16958, 2020. URL https://arxiv.org/abs/2006.16958. 93

[58] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkE6PjC9KX. 30

[59] Sungwon Kim, Juhwan Noh, Geun Ho Gu, Alan Aspuru-Guzik, and Yousung Jung. Generative adversarial networks for crystal structure prediction. *ACS central science*, 6(8):1412–1420, 2020. 46

[60] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`. 27

[61] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 9, 16, 43

[62] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL `http://arxiv.org/abs/1312.6114`. cite arxiv:1312.6114. 30, 32

[63] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Proceedings of the 31st international conference on neural information processing systems*, pages 972–981, 2017. 10

[64] Johannes Klicpera, Shankari Giri, Johannes T Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *arXiv preprint arXiv:2011.14115*, 2020. 38, 43

[65] Tamara G Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM review*, 45(3):385–482, 2003. 8

[66] Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. *NeurIPS*, 2019. 2, 5, 6, 8, 9, 13, 15, 16, 17, 20, 21, 22, 25, 30, 31, 32, 46, 94, 102

[67] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020. 3, 21, 24, 28, 44, 102

[68] Aviral Kumar, Amir Yazdanbakhsh, Milad Hashemi, Kevin Swersky, and Sergey Levine. Data-driven offline optimization for architecting hardware accelerators. *arXiv preprint arXiv:2110.11346*, 2021. 44

[69] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 21

[70] Thomas Liao, Grant Wang, Brian Yang, Rene Lee, Kristofer Pister, Sergey Levine, and Roberto Calandra. Data-efficient learning of morphology and controller for a microrobot. In *2019 IEEE International Conference on Robotics and Automation*, 2019. URL `https://arxiv.org/abs/1905.01334`. 1, 20

[71] Johannes Linder, Nicholas Bogard, Alexander B Rosenberg, and Georg Seelig. A generative neural network for maximizing fitness and diversity of synthetic dna and protein sequences. *Cell systems*, 11(1):49–62, 2020. 60, 115

[72] Daniel James Lizotte. *Practical bayesian optimization*. University of Alberta, 2008. 8

[73] David C Lonie and Eva Zurek. Xtalopt: An open-source evolutionary algorithm for crystal structure prediction. *Computer Physics Communications*, 182(2): 372–387, 2011. 45

[74] Carol H Miao, Kazuo Ohashi, Gijsbert A Patijn, Leonard Meuse, Xin Ye, Arthur R Thompson, and Mark A Kay. Inclusion of the hepatic locus control region, an intron, and untranslated region increases and stabilizes hepatic factor ix gene expression in vivo but not in vitro. *Molecular Therapy*, 1(6):522–532, 2000. 52

[75] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020. 30

[76] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014. URL `http://arxiv.org/abs/1411.1784`. cite arxiv:1411.1784. 9

[77] J. J. Mortensen, L. B. Hansen, and K. W. Jacobsen. Real-space grid implementation of the projector augmented wave method. *Phys. Rev. B*, 71:035109, Jan 2005. doi: 10.1103/PhysRevB.71.035109. URL `https://link.aps.org/doi/10.1103/PhysRevB.71.035109`. 109

[78] Rajiv Movva, Peyton Greenside, Georgi K Marinov, Surag Nair, Avanti Shrikumar, and Anshul Kundaje. Deciphering regulatory dna sequences and noncoding genetic variants using neural network models of massively parallel reporter assays. *PLoS One*, 14(6):e0218073, 2019. 56

[79] Lior Nissim, Ming-Ru Wu, Erez Pery, Adina Binder-Nissim, Hiroshi I Suzuki, Doron Stupp, Claudia Wehrspaun, Yuval Tabach, Phillip A Sharp, and Timothy K Lu. Synthetic rna-based immunomodulatory gene circuits for cancer immunotherapy. *Cell*, 171(5):1138–1150, 2017. 52, 60

[80] Artem R Oganov, Andriy O Lyakhov, and Mario Valle. How evolutionary crystal structure prediction works and why. *Accounts of chemical research*, 44(3):227–237, 2011. 45

[81] Saibal K Pal, CS Rai, and Amrit Pal Singh. Comparative study of firefly algorithm and particle swarm optimization for noisy non-linear optimization problems. *International Journal of intelligent systems and applications*, 4(10):50, 2012. 8

[82] Robert G. Parr. Density-functional theory of atoms and molecules. 1989. 109

[83] Martin Pelikan, David E Goldberg, Erick Cantú-Paz, et al. Boa: The bayesian

optimization algorithm. In *Proceedings of the genetic and evolutionary computation conference GECCO-99*, volume 1, pages 525–532. Citeseer, 1999. 46

[84] Tiago Pereira, Maryam Abbasi, Bernardete Ribeiro, and Joel P. Arrais. Diversity oriented deep reinforcement learning for targeted molecule generation. *Journal of Cheminformatics*, 13, 2020. 46

[85] Valerio Perrone, Rodolphe Jenatton, Matthias Seeger, and Cedric Archambeau. Multiple adaptive bayesian linear regression for scalable bayesian optimization with warm start. *arXiv preprint arXiv:1712.02902*, 2017. 8

[86] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, 2007. 30

[87] Han Qi, Yi Su, Aviral Kumar, and Sergey Levine. Data-driven offline decision-making via invariant representation learning. *arXiv preprint arXiv:2211.11349*, 2022. 42, 46

[88] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John F. Canny, Pieter Abbeel, and Yun S. Song. Evaluating protein transfer learning with TAPE. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 9686–9698, 2019. URL `http://papers.nips.cc/paper/9163-evaluating-protein-transfer-learning-with-tape`. 90, 106

[89] Aniketh Janardhan Reddy, Michael H Herschl, Sathvik Kolli, Amy X Lu, Xinyang Geng, Aviral Kumar, Patrick D Hsu, Sergey Levine, and Nilah M Ioannidis. Pretraining strategies for effective promoter-driven gene expression prediction. *bioRxiv*, pages 2023–02, 2023. 53, 55, 56, 112, 113, 114

[90] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 114

[91] Reuven Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operations Research*, 99:89–112, 1996. 30

[92] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-carlo Simulation (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2004. ISBN 038721240X. 30

[93] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013. 8

[94] James E Saal, Scott Kirklin, Muratahan Aykol, Bryce Meredig, and Christopher Wolverton. Materials design and discovery with high-throughput density functional theory: the open quantum materials database (oqmd). *Jom*, 65:1501–1509, 2013. 41

[95] Paul J Sample, Ban Wang, David W Reid, Vlad Presnyak, Iain J McFadyen, David R Morris, and Georg Seelig. Human 5‘ utr design and variant effect prediction from a massively parallel translation assay. *Nature biotechnology*, 37 (7):803–809, 2019. 90, 91, 103, 106

[96] Shibani Santurkar, Andrew Ilyas, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Image synthesis with a single (robust) classifier. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 1260–1271, 2019. 30

[97] 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(7603):397–401, May 2016. ISSN 1476-4687. doi: 10.1038/nature17995. URL `https://doi.org/10.1038/nature17995`. 1, 20, 32, 37, 90, 103

[98] Victor Garcia Satorras, Emiel Hoogeboom, Fabian B. Fuchs, Ingmar Posner, and Max Welling. E(n) equivariant normalizing flows for molecule generation in 3d. *ArXiv*, abs/2105.09016, 2021. 46

[99] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL `http://arxiv.org/abs/1707.06347`. 11, 86, 104

[100] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015. 8

[101] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104:148–175, 2016. 30

[102] Wen-Jun Shen, Hau-San Wong, Quan-Wu Xiao, Xin Guo, and Stephen Smale. Introduction to the peptide binding problem of computational immunology: New results. *Foundations of Computational Mathematics*, 14(5):951–984, Oct 2014. ISSN 1615-3383. doi: 10.1007/s10208-013-9173-9. URL https://doi.org/10.1007/s10208-013-9173-9. 4

[103] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. *ArXiv*, abs/1802.03480, 2018. 46

[104] Prabhu Teja Sivaprasad, Florian Mai, Thijs Vogels, Martin Jaggi, and François Fleuret. On the tunability of optimizers in deep learning. *CoRR*, abs/1910.11758, 2019. URL http://arxiv.org/abs/1910.11758. 93

[105] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, 2012. URL http://dl.acm.org/citation.cfm?id=2999325.2999464. 5, 8, 20, 21, 30, 46

[106] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, 2015. 30, 46

[107] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180, 2015. 8

[108] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/3001ef257407d5a371a96dcd947c7d93-Paper.pdf. 43

[109] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 111

[110] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021. 56, 112

[111] Tabula Sapiens Consortium. The tabula sapiens: A multiple-organ, single-cell transcriptomic atlas of humans. *Science*, 376(6594):eabl4896, 2022. 52

[112] Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-confidence off-policy evaluation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015. 27

[113] 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*, pages 5026–5033. IEEE, 2012. 12

[114] Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-bench: Benchmarks for data-driven offline model-based optimization, 2021. URL `https://github.com/brandontrabucco/design-bench`. 30, 31, 32, 35, 46, 54, 60, 97, 99, 102

[115] Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pages 10358–10368. PMLR, 2021. 38, 39, 42, 44, 45, 46, 54, 57, 108, 110

[116] Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In Marina Meila and Tong Zhang, editors, *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*, pages 10358–10368. PMLR, 2021. URL `http://proceedings.mlr.press/v139/trabucco21a.html`. 5, 16, 96

[117] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=rkZvSe-RZ`. 30

[118] Joris van Arensbergen, Vincent D FitzPatrick, Marcel de Haas, Ludo Pagie, Jasper Sluimer, Harmen J Bussemaker, and Bas van Steensel. Genome-wide mapping of autonomous promoter activity in human cells. *Nature biotechnology*, 35(2):145–153, 2017. 112

[119] Joris van Arensbergen, Ludo Pagie, Vincent D FitzPatrick, Marcel de Haas, Marijke P Baltissen, Federico Comoglio, Robin H van der Weide, Hans Teunissen, Urmo Võsa, Lude Franke, et al. High-throughput identification of human snps affecting regulatory element activity. *Nature genetics*, 51(7):1160–1169, 2019. 56, 112

[120] Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer, 1987. 8

[121] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones,

Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 56, 111

[122] Jeff Vierstra, John Lazar, Richard Sandstrom, Jessica Halow, Kristen Lee, Daniel Bates, Morgan Diegel, Douglas Dunn, Fidencio Neri, Eric Haugen, et al. Global reference mapping of human transcription factor footprints. *Nature*, 583(7818): 729–736, 2020. 63, 113

[123] Aron Walsh, Shiyou Chen, Su-Huai Wei, and Xin-Gao Gong. Kesterite thin-film solar cells: Advances in materials modelling of cu2znsns4. *Advanced Energy Materials*, 2(4):400–409, 2012. 38

[124] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988. 12

[125] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2): 65–85, 1994. 8

[126] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006. 4

[127] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, May 1992. 30

[128] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL `https://doi.org/10.1007/BF00992696`. 15, 31

[129] James T. Wilson, Riccardo Moriconi, Frank Hutter, and Marc Peter Deisenroth. The reparameterization trick for acquisition functions. *CoRR*, abs/1712.00424, 2017. URL `http://arxiv.org/abs/1712.00424`. 16, 32

[130] Scott M Woodley and Richard Catlow. Crystal structure prediction from first principles. *Nature materials*, 7(12):937–946, 2008. 38

[131] Ming-Ru Wu, Lior Nissim, Doron Stupp, Erez Pery, Adina Binder-Nissim, Karen Weisinger, Casper Enghuus, Sebastian R Palacios, Melissa Humphrey, Zhizhuo Zhang, et al. A high-throughput screening and computation platform for identifying synthetic promoters with enhanced cell-state specificity (specs). *Nature communications*, 10(1):1–10, 2019. 52, 60

[132] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 111

[133] Tian Xie, Xiang Fu, Octavian-Eugen Ganea, Regina Barzilay, and Tommi Jaakkola. Crystal diffusion variational autoencoder for periodic material generation. *arXiv preprint arXiv:2110.06197*, 2021. 38, 42, 43, 45, 46, 110

[134] Tomoki Yamashita, Hiroyoshi Momida, and Tamio Oguchi. Crystal structure predictions of naxc6o6 for sodium-ion batteries: First-principles calculations with an evolutionary algorithm. *Electrochimica Acta*, 195:1–8, 2016. 38

[135] Xin-She Yang and Adam Slowik. Firefly algorithm. In *Swarm Intelligence Algorithms*, pages 163–174. CRC Press, 2020. 8

[136] Sihyun Yu, Sungsoo Ahn, Le Song, and Jinwoo Shin. Roma: Robust model adaptation for offline model-based optimization. *Advances in Neural Information Processing Systems*, 34, 2021. 42

[137] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 10

[138] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. 2017. URL `https://arxiv.org/abs/1611.01578`. 20

Appendix A

Appendix for Design-Bench

A.1 Data Collection

In this section, we detail the data collection steps used for creating each of the tasks in design-bench. We answer **(1)** where is the data from, and **(2)** what pre-processing steps are used?

A.1.1 TF Bind 8 and TF Bind 10

The TF Bind 8 and TF Bind 10 tasks are derivativesof the transcription factor binding activity survey performed by [8], where the binding activity scores of every possible DNA sequence was measured with a variety of human transcription factors. We filter the dataset by selecting a particular transcription factor `SIX6_REF_R1`, and defining an optimization problem where the goal is to synthesize a length 8 DNA sequence with high binding activity with human transcription factor `SIX6_REF_R1`. This particular transcription factor for TF Bind 8 was recently used for optimization in [5, 6]. TF Bind 8 is a fully characterized dataset containing 65792 samples, representing every possible length 8 combination of nucleotides $\mathbf{x}_{\text{TFBind8}} \in \{0, 1\}^{8\times 4}$. The training set given to offline MBO algorithms is restricted to the bottom 50%, which results in a visible training set of 32898 samples.

A.1.2 ChEMBL

The ChEMBL task is a derivative of a much larger dataset that is derived from ChEMBL [36], a large database of chemicals and their properties. The datawas

originally collected by performing physical experiments on a large number of molecules, and measuring a chemical property in the presence of a target assay. We have processed the ChEMBL database—available at `https://www.ebi.ac.uk/chembl/g/#browse/activities`—into collections of smaller datasets mapping particular molecules to measured values, determined by a target assay that accompanies each set. We choose the assay specified by `ASSAY_CHEMBL_ID` = `CHEMBL3885882` and select the standard type of `MCHC` as the measurement to maximize with offline model-based optimization. The resulting dataset has 1093 samples in total. This assay is chosen for its high validation rank correlation, namely 0.7141, when fitting a random forest regression model to map molecules to `MCHC` values. The majority of other assays in ChEMBL produce a validation rank correlation below 0.5. We preprocess the dataset by converting each molecule into a SMILES string using RDKit, and then apply the DeepChem `SmilesTokenizer` to convert each SMILES string into a sequence of integer tokens. We then remove all molecules whose SLIMES sequence is longer than a maximum of 31 tokens with the vocabulary has 591 elements, $\mathbf{x}_{\text{ChEMBL}} \in \{0,1\}^{31\times 591}$. When evaluating MBO methods, we remove the top 50% of molecules sorted by their `MCHC` value to increase task difficulty.

A.1.3 Superconductor

The Superconductor task is inspired by recent work [29] that applies offline MBO to optimize the properties of superconducting materials for high critical temperature. The data we provide in our benchmark is real-world superconductivity data originally collected by [48], and subsequently made available to the public at `https://archive.ics.uci.edu/ml/datasets/Superconductivty+Data#`. The original dataset consists of superconductors featurized into vectors containing measured physically properties like the number of chemical elements present, or the mean atomic mass of such elements. One issue with the original dataset that was used in [29] is that the numerical representation of the superconducting materials did not lend itself to recovering a physically realizable material that could be synthesized in a lab after performing model-based optimization. In order to create an *invertible* input specification, we deviate from prior work and encode superconductors as vectors whose components represent the number of atoms of specific chemical elements present in the superconducting material—a serialization of the chemical formula of each superconductor. The result is a real-valued design space with 86 components $\mathbf{x}_{\text{Superconductor}} \in \mathcal{R}^{86}$. The full dataset used to learn approximate oracles for evaluating MBO methods has 21263 samples, but we restrict this number to 17010 (the 80th percentile) for the training set of offline MBO methods to increase difficulty.

A.1.4 Ant & D'Kitty Morphology

Both morphology tasks are collected by us, and share methodology. The goal of these tasks is to design the morphology of a quadrupedal robot—an ant or a D'Kitty—such that the agent is able to crawl quickly in a particular direction. In order to collect data for this environment, we create variants of the MuJoCo Ant and the ROBEL D'Kitty agents that have parametric morphologies. The goal is to determine a mapping from the morphology of the agent to the average return of the agent using a controller optimized for that morphology. In order to facilitate fast optimization, we pre-compute a morphology conditioned neural network controller using SAC [45] that has been trained to perform optimally on a wide range of morphologies. For both the Ant and the D'Kitty, we train the controllers for more than ten million environment steps, and a maximum episode length of 200, with all other settings as default. These morphology conditioned controllers are trained on Gaussian distributions of morphologies. The Gaussian distributions are obtained by adding Gaussian noise with standard deviation 0.03 for Ant and 0.01 for D'Kitty the design-space range to the default morphologies. After obtaining trained morphology-conditioned controllers, we create a dataset of morphologies for model-based optimization by sampling initialization points randomly, and then using CMA-ES to optimize for morphologies that attain high reward using the morphology-conditioned controllers. To obtain initialization points, we add Gaussian random noise to the default morphology for the Ant with standard deviation 0.075 and D'Kitty with standard deviation 0.1, and then apply CMA-ES with standard deviation 0.02. We ran CMA-ES for 250 iterations and then restarted, until 25000 morphologies were collected, resulting in 25009 samples for both the Ant and D'Kitty. The design space for Ant morphologies is $\mathbf{x}_{\text{Ant}} \in \mathcal{R}^{60}$, whereas for D'Kitty morphologies is $\mathbf{x}_{\text{D'Kitty}} \in \mathcal{R}^{56}$. We sremove the top 40% of samples when training offline MBO algorithms.

A.1.5 NAS

The data for the NAS task is collected by us. The goal of this task is to search for a good neural network architecture to optimize the test accuracy on the CIFAR10 dataset. The architecture search space is a 64-dimensional discrete variable with 5 categories for each dimension. We collect the dataset by randomly sample architecture designs in the search space, and train them on the CIFAR10 dataset. We sample 2440 total designs, and select the bottom performing 70% to be our training set. This gives us 1771 samples in total, with the test accuracy ranging from 59.3% to 63.8%.

A.1.6 Hopper Controller

The goal of this task is to design a set of weights for as neural network policy, in order to achieve high expected return when evaluating that policy. The data collected for

Hopper Controller was taken by training a three layer neural network policy with 64 hidden units and 5126 total weights on the Hopper-v2 MuJoCo task using Proximal Policy Optimization [99]. Specifically, we use the default parameters for PPO provided in stable baselines [52]. The dataset we provide with this benchmark has 3200 unique weights. In order to collect this many, we run 32 experimental trials of PPO, where we train for one million steps, and save the weights of the policy every 10,000 environment steps. The policy weights are represented originally as a list of tensors. We first traverse this list and flatten each of the tensors, and we then concatenate each of these flattened tensors into a single training example $x_{\text{Hopper}} \in \mathcal{R}^{5126}$. The result is an optimization problem over neural network weights. After collecting these weights, we perform no additional pre-processing steps. In order to collect objective score values we perform a single rollout for each x using the Hopper-v2 MuJoCo environment. The horizon length for training and evaluation is limited to 1000 simulation time steps.

A.2 Oracle Functions

We detail oracle functions for evaluating ground truth scores for each of the tasks in design-bench. A common thread among these is that the oracle, if trained, is fit to a larger static dataset containing higher performing designs than observed by a downstream MBO algorithm.

A.2.1 TF Bind 8 and TF Bind 10

TF Bind 8 and TF Bind 10 are a fully characterized discrete offline MBO tasks, which means that all possible designs have been evaluated [8] and are contained in the full hidden datasets. The oracles are therefore implemented simply as a lookup table that returns the score corresponding to a particular DNA sequence from the dataset. By restricting the size of the training set visible to an offline MBO algorithm, it is possible for the algorithm to propose a design that achieves a higher score than any other DNA sequence visible to the algorithm during training.

A.2.2 ChEMBL

We tested several models as candidate oracle functions for ChEMBL [36], including a Gaussian Process, Random Forest, CNN, and Transformer regression models. We ultimately chose the Random Forest model in scikit-learn due to its quick inference and relatively high performance compared with neural network alternatives, achieving a spearman's rank correlation coefficient of 0.7141 with a held-out validation set. These models were trained on the entire hidden ChEMBL dataset for `ASSAY_CHEMBL_ID` = `CHEMBL3885882` with standard type `MCHC` encoded into SMILES and tokenized. Hyper-parameters for the random forest oracle are provided in the official github release of

design-bench.

A.2.3 Superconductor

The Superconductor oracle function is also a random forest regression model. The model we use it the model described by [48]. We borrow the hyperparameters described by them, and we use the RandomForestRegressor provided in scikit-learn. Similar to the setup for the previous set of tasks, this oracle is trained on the entire hidden dataset of superconductors. The random forest has a rank correlation of 0.9155 with a held-out validation set.

A.2.4 Ant & D'Kitty Morphology

The Ant & D'Kitty Morphology tasks in design-bench use an exact oracle function, using the MuJoCo simulator. For both morphology tasks, the simulator performs a rollout and returns the sum of rewards at every timestep in that rollout. Each task is accompanied by a pre-trained morphology-conditioned policy. To perform evaluation, a morphology is passed to the Ant or D'Kitty MuJoCo environments respectively, and a dynamic-morphology agent is initialized inside these environments. These simulations can be time consuming to run, and so we limit the rollout length to 100 steps. The morphology conditioned policies were trained using the reinforcement learning algorithm SAC for 10 million steps for each task, and are ReLU networks with two hidden layers of size 64.

A.2.5 NAS

The NAS task in the design bench uses an exact oracle, where we train the proposed architecture on CIFAR10 and then test it on the test set. To perform the evaluation, we construct the proposed architecture using PyTorch, and train it for 20 epochs using batch size 256 and then compute the test accuracy on the test set.

A.2.6 Hopper Controller

Unlike the previously described tasks, Hopper Controller implements an exact oracle function. For Hopper Controller the oracle takes the form of a single rollout using the Hopper-v2 MuJoCo environment. The designs for Hopper Controller are neural network weights, and during evaluation, a policy with those weights is instantiated—in this case that policy is a three layer neural network with 11 input units, two layers with 64 hidden units, and a final layer with 3 output units. The intermediate activations between layers are hyperbolic tangents. After building a policy, the Hopper-v2 environment is reset and the reward for 1000 time-steps is summed. That summed reward constitutes

the score returned by the Hopper Controller oracle. The limit of performance is the maximum return that an agent can achieve in Hopper-v2 over 1000 steps.

A.3 Experimental Details

In this section we present additional details for the experiments, including the score normalization process and 50th percentile performance.

A.3.1 Objective Normalization

In order to report performance on the same order of magnitude for each offline model-based optimization task in Design-Bench, we normalize the performance reported in Table 2.2 by calculating the minimum objective value y_{min} and the the maximum objective value y_{max} in the full unobserved dataset associated with each offline model-based optimization problem. *Crucially*, note that this is not the same as normalizing with respect to the best and worst samples in the training dataset used by the offline MBO algorithm, but rather a bigger dataset of designs and objective values. We then report performance by calculating what fraction of the distance between y_{min} and y_{max} is attained by a particular offline MBO baseline.

$$y_{\text{normalized}}(y) = \frac{y - y_{\text{min}}}{y_{\text{max}} - y_{\text{min}}} \tag{A.1}$$

The final performance $y_{\text{normalized}}$ is the normalized performance of an offline MBO method that achieved an unprocessed objective value of y. The result is larger than one when the offline MBO method finds a solution more performance than all solutions in the full unobserved dataset associated with the corresponding task. The result is less than zero when the offline MBO method finds a solution attaining less performance than all samples in the full unobserved dataset.

A.3.2 50th Percentile Experiment Results

In this section, we present the 50th percentile performance of the runs presented in Table 2.2 of Chapter 2. Similar to the 100th percentile performance reported in the main text, performance is calculated by evaluating solutions to each task found by an optimization method, subtracting the minimum objective value present in the corresponding task dataset, and dividing by the range of objective values present in the corresponding task dataset. The result is a performance of greater than one if optimization converges to a solution with a higher objective value that the best observed design in the corresponding task dataset.

	TF Bind 8	TF Bind 10	ChEMBL	NAS	Superconductor	Ant Morphology	D'Kitty Morphology
Auto. CbAS	0.419 ± 0.007	0.461 ± 0.007	-1.823 ± 0.000	0.217 ± 0.005	0.131 ± 0.010	0.364 ± 0.014	0.736 ± 0.025
CbAS	0.428 ± 0.010	0.463 ± 0.007	-1.807 ± 0.004	0.292 ± 0.027	0.111 ± 0.017	0.384 ± 0.016	0.753 ± 0.008
BO-qEI	0.439 ± 0.000	0.467 ± 0.000	-1.774 ± 0.020	0.544 ± 0.099	0.300 ± 0.015	0.567 ± 0.000	0.883 ± 0.000
CMA-ES	0.537 ± 0.014	0.484 ± 0.014	-1.763 ± 0.019	0.591 ± 0.102	0.379 ± 0.003	-0.045 ± 0.004	0.684 ± 0.016
Gradient Ascent	0.609 ± 0.019	0.474 ± 0.005	-1.772 ± 0.018	0.433 ± 0.000	0.476 ± 0.022	0.134 ± 0.018	0.509 ± 0.200
Grad. Min	0.645 ± 0.030	0.470 ± 0.002	-1.769 ± 0.014	0.433 ± 0.000	0.471 ± 0.016	0.185 ± 0.008	0.746 ± 0.034
Grad. Mean	0.616 ± 0.023	0.471 ± 0.004	-1.757 ± 0.010	0.433 ± 0.000	0.469 ± 0.022	0.187 ± 0.009	0.748 ± 0.024
MINs	0.421 ± 0.015	0.468 ± 0.006	-1.745 ± 0.000	0.433 ± 0.000	0.336 ± 0.016	0.618 ± 0.040	0.887 ± 0.004
REINFORCE	0.462 ± 0.021	0.475 ± 0.008	-1.805 ± 0.003	-1.895 ± 0.000	0.463 ± 0.016	0.138 ± 0.032	0.356 ± 0.131
COMs	0.497 ± 0.038	0.465 ± 0.008	0.633 ± 0.000	0.287 ± 0.173	0.386 ± 0.018	0.519 ± 0.026	0.885 ± 0.003

Table A.1: 50th percentile evaluations for baselines on every task. Results are averaged over 8 trials, and the ± indicates the standard deviation of the reported performance. This table corresponds to the normalized performance, using the normalization methodology described in Appendix A.3.1

.

A.3.3 Unnormalized Experimental Results

In this section, we present the raw 100th percentile performance of the runs presented in Table 2.2 of Chapter 2. These values, presented in Table A.2, represent the mean raw objective values and the standard deviation of the objective values attained by various offline MBO methods.

	TF Bind 8	TF Bind 10	ChEMBL	NAS	Superconductor	Ant Morphology	D'Kitty Morphology
Auto. CbAS	0.910 ± 0.044	0.655 ± 0.178	42467.285 ± 0.000	64.530 ± 0.764	77.910 ± 8.361	474.888 ± 44.424	226.156 ± 7.043
CbAS	0.927 ± 0.051	0.738 ± 0.239	46681.988 ± 4987.456	66.360 ± 0.820	93.078 ± 12.695	469.499 ± 30.570	209.412 ± 9.593
BO-qEI	0.798 ± 0.083	0.742 ± 0.150	30069.684 ± 3187.300	70.447 ± 0.606	74.322 ± 6.347	413.084 ± 0.000	213.816 ± 0.000
CMA-ES	0.953 ± 0.022	0.811 ± 0.090	31607.031 ± 1578.222	69.475 ± 0.821	86.072 ± 4.508	799.394 ± 715.702	4.290 ± 1.505
Gradient Ascent	0.977 ± 0.025	0.762 ± 0.155	32514.541 ± 2612.903	63.770 ± 0.000	95.789 ± 4.436	-100.265 ± 22.118	187.206 ± 27.274
Grad. Min	0.984 ± 0.012	0.729 ± 0.126	32617.006 ± 370.390	63.770 ± 0.000	93.590 ± 1.719	80.853 ± 62.308	205.639 ± 13.427
Grad. Mean	0.986 ± 0.012	0.714 ± 0.071	33715.059 ± 1136.034	63.770 ± 0.000	92.265 ± 3.206	48.064 ± 78.555	209.355 ± 13.928
MINs	0.905 ± 0.052	0.599 ± 0.082	42732.578 ± 5126.862	66.709 ± 0.471	86.702 ± 4.171	505.515 ± 34.934	273.479 ± 14.184
REINFORCE	0.948 ± 0.028	0.786 ± 0.137	41448.012 ± 3220.380	39.720 ± 0.000	88.996 ± 2.389	-127.440 ± 30.831	-194.540 ± 238.857
COMs	0.945 ± 0.033	0.649 ± 0.153	391827.500 ± 2273.631	64.041 ± 1.431	81.238 ± 6.170	535.125 ± 16.064	278.344 ± 17.727

Table A.2: Unnormalized 100th percentile unnormalized evaluations for baselines on every task. Results are averaged over 8 trials, and the ± indicates the standard deviation of the reported performance. This table corresponds to the unnormalized performance.

A.3.4 Computation Resources

The amount of computation resources required to produce the experiments for design-bench is relatively modest except for the NAS tasks. We ran our experiments on a single server with 2 Intel Xeon E5-2698 v4 CPUs and 8 Nvidia Tesla V100 GPUs. All our experiments can be completed within 96 hours on this single machine.

A.4 Additional MBO Tasks That Were Discarded From Our Benchmark

The main benchmark consists of eight offline MBO tasks, four of which have discrete design-spaces, and four of which have contiguous design-spaces. In addition to the provided tasks, we also experimented with two other candidate MBO tasks from prior

work [6, 12], but chose to not include them in the final benchmark due to lack indistinguishable results across all methods, suggesting that these tasks may not be suitable for devising better algorithms.

A.4.1 GFP

GFP uses the oracle function derived from Rao et al. [88]. This oracle is a transformer regression model with 4 attention blocks and a hidden size of 64. The Transformer is fit to the entire hidden GFP dataset, making it possible to sample a protein design that achieves a higher score than any other protein visible to an offline MBO algorithm. Our Transformer has a Spearman's rank correlation coefficient of 0.8497 with a held-out validation set derived from the GFP dataset.

The GFP task provided is a derivative of the GFP dataset [97]. The dataset we use in practice is that provided by Brookes et al. [12] at the url `https://github.com/dhbrookes/CbAS/tree/master/data`. We process the dataset such that a single training example consists of a protein represented as a tensor $\mathbf{x}_{\text{GFP}} \in \{0, 1\}^{237 \times 20}$. This tensor is a sequence of 237 one-hot vectors corresponding to which amino acid is present in that location in the protein. We use the dataset format of [12] with no additional processing. The data was originally collected by performing laboratory experiments constructing proteins similar to the Aequorea victoria green fluorescent protein and measuring fluorescence. We employ the full dataset of 56086 proteins when learning approximate oracles for evaluating offline MBO methods, but restrict the training set given to offline MBO algorithms to 5000 samples drawn from between the 50th percentile and 60th percentile of proteins in the GFP dataset, sorted by fluorescence values. This subsampling procedure is consistent with prior work [12].

A.4.2 UTR

UTR uses a Transformer as the oracle function, which differs from the CNN that was originally used by [6]. Our reasoning for making this change is that the Transformer is a newer and possibly higher capacity model that may be less prone to mistakes than the shallower CNN model proposed by Sample et al. [95]. This Transformer has 4 attention blocks and a hidden size of 64. The Transformer is fit to the entire hidden UTR dataset, making it possible to sample a DNA sequence that achieves a higher score than any other sequence visible to an offline MBO algorithm. The resulting model has a spearman's rank correlation of 0.6424 with a held-out validation set.

The UTR task is derived from work by Sample et al. [95] who trained a CNN model to predict the expressive level of a particular gene from a corresponding 5'UTR sequence. Our use of the UTR task for model-based optimization follows Angermüller et al. [6], where the goal is to design a length 50 DNA sequence to maximize expression level. We follow the methodology set by Sample et al. [95] to sort all length 50 DNA

sequences in the unprocessed UTR dataset by total reads, and then select the top 280,000 DNA sequences with the most total reads. The result is a dataset containing 280,000 samples of length 50 DNA sequences $\mathbf{x}_{\text{UTR}} \in \{0,1\}^{50\times 4}$ and corresponding ribosome loads. When training offline MBO algorithms, we subsequently eliminate the top 50% of sequences ranked by their ribosome load, resulting in a visible dataset with only 140,000 samples.

A.4.3 Additional Experimental Results

We report the normalized performance of all baselines on the three additional MBO tasks that were not chosen for inclusion in the benchmark. Note that for GFP [12] and UTR [6, 95] performance of offline MBO method is not not distinguishable, and we consider this an indication each task is not suitable for benchmarking offline MBO methods. We encourage future revisions of these tasks.

	GFP	UTR
Auto. CbAS	0.865 ± 0.000	0.691 ± 0.012
CbAS	0.865 ± 0.000	0.694 ± 0.010
BO-qEI	0.254 ± 0.352	0.684 ± 0.000
CMA-ES	0.054 ± 0.002	0.707 ± 0.014
Grad.	0.864 ± 0.001	0.695 ± 0.013
Grad. Min	0.864 ± 0.000	0.696 ± 0.009
Grad. Mean	0.864 ± 0.000	0.693 ± 0.010
MINs	0.865 ± 0.001	0.697 ± 0.010
REINFORCE	0.865 ± 0.000	0.688 ± 0.010
COMs	0.864 ± 0.000	0.699 ± 0.011

Table A.3: Normalized 100th percentile normalized evaluations for baselines on unused tasks. Each entry reports the empirical mean and empirical standard deviation over 8 independent trials.

A.5 Normalization Of Inputs and Outputs Is Important for Gradient Ascent

An important component for the good performance of the gradient ascent baseline is the normalization of design space. We found that the identical gradient-ascent baseline performed a factor 1.4x worse on Hopper Controller, when optimizing in the space of unnormalized designs and objective values, as seen in Figure A.1. This indicates that normalization is key in obtaining good performance with a naïve gradient ascent baseline. For continuous design-space tasks, we normalize both the designs, and the scores to have unit Gaussian statistics. For discrete design-space tasks, we first map designs to real-valued logits of a categorical distribution before performing this normalization. See the official code for how this mapping is performed. This is a necessary part of the optimization workflow because scores vary by several orders of magnitude in the

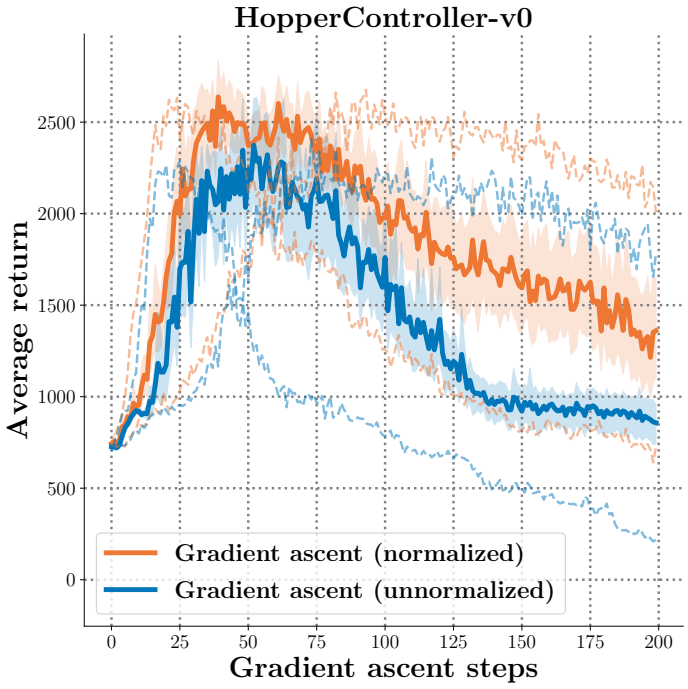


Figure A.1: Comparison of unnormalized vs normalized gradient ascent in the HopperController-v0 task.

dataset, for example, 0.91 for TF Bind 8 and as high as 799.394 for Ant Morphology. The specific normalization equation is given below.

$$\tilde{\mathbf{x}}_{i,j} = \frac{\mathbf{x}_{i,j} - \mu(\mathbf{x}_{,j})}{\sigma(\mathbf{x}_{,j})} \quad : \quad \mathbf{x} \in \mathbb{R}^{N \times D} \tag{A.2}$$

We also normalize the objective values in a similar fashion to have unit Gaussian statistics. The result in a new set of designs $\tilde{\mathbf{x}}$ and objective values $\tilde{y}$ that is optimized over

$$\tilde{y}_{i,j} = \frac{y_{i,j} - \mu(y_{,j})}{\sigma(y_{,j})} \quad : \quad y \in \mathbb{R}^{N \times 1} \tag{A.3}$$

The gradient ascent procedure is performed in the space of these normalized designs. Suppose T steps of gradient ascent have been taken, and a final normalized solution $\tilde{\mathbf{x}}_T^*$ is found. This solution is de-normalized using the following transformation.

$$(\mathbf{x}_T^*)_{ij} = (\tilde{\mathbf{x}}_T^*)_{ij} \cdot \sigma(\mathbf{x}_{,j}) + \mu(\mathbf{x}_{,j}) \tag{A.4}$$

This normalization strategy is heavily inspired by data whitening, which is known to reduce the variance of machine learning algorithms that learn discriminative mappings on that data. The learned model of the objective function is one such discriminative model, and normalization likely improves the consistency of Gradient Ascent across independent experimental trials.

A.6 Hyperparameter Selection Workflow

Hyperparameter tuning under a restricted computational budget is emerging as an import research domain in optimization [104, 23, 57]. Care must be taken when tuning each of the prescribed algorithms so that only offline information about the task is used for hyperparameter selection. Formally, this means that the hyperparameters, $\mathcal{H}$, are conditionally independent of the particular value of the performance metric $\mathcal{M}$, given the offline task dataset $\mathcal{D}$. Examples of hyperparameter selection strategies that violate this requirement might, for example, perform a grid search over $\mathcal{H}$ and take the set that maximizes the performance metric, but this is not offline. An example of a tuning strategy that is fully offline is tuning the parameters of a learned model such that is is a good fit for the task dataset $\mathcal{D}$. One can choose $\mathcal{H}$ that minimizes a validation loss, such as negative log likelihood. A detailed record of hyperparameters can be found in the experiment scripts located alongside our reference implementations: .

We now present guidelines for hyperparameter selection (i.e. ***workflow***) for methods evaluated in the benchmark. These are general principles that can be used to tune the hyperparameters of these methods on a new task in an offline fashion. While we only present workflow details for methods we benchmark in Section 4.6, we expect that these general strategies will allow users to devise analogous schemes for tuning hyperparameters of new offline MBO methods with shared components.

A.6.1 Strategy For Autofocused CbAS

The main tunable components of Autofocused methods [29] are the learned objective function, and the generative model fit to the data distribution. When training the learned objective function, tracking a validation performance metric like rank correlation is helpful to ensure that the resulting learned model is able to generalize beyond its training dataset. This tracking is especially important for Autofocused methods because re-fitting the learned objective model during importance can lead to divergence if the importance weights generated by Autofocusing are very large or very small in magnitude. The algorithm is tuned well if, for example, the validation rank correlation stays above a positive threshold, such as a threshold of 0.9.

The second component of Autofocused methods is the fit of the generative model used for sampling designs. The algorithm has the best chance of success if the generative model can generalize beyond the dataset in which it was trained. This can be monitored by holding out a validation set and tracking a metric such as negative log likelihood on this held-out set. In the case when the generative model is not an exact likelihood-based generative model—for example, a VAE—other validation metrics can be used that measure the fit of the generative model on a validation set. The generative model is especially impacted by the importance sampling procedure used by Estimation of Distribution Algorithms (EDAs), and tracking the effective sample size of the

importance weights can help diagnose when the generative model is failing to generalize to a validation set.

A.6.2 Strategy For CbAS

The main tunable components of CbAS methods [12] are the learned objective function, and the generative model fit to the data distribution. While the learned objective function is not affected by the importance sampling weights generated by CbAS, the same tuning strategy described in section A.6.1 that focuses on generalization to a validation set is effective. Generative model tuning can also follow an identical strategy to that described in section A.6.1, which focuses on the ability for the generative model to represent samples outside of its training set. In the case of a β-VAE, which is used with CbAS in this work, the main parameter for controlling this generalization ability is the β parameter. We found that β is task specific, and must be found in order for the CbAS optimizer using β-VAE to generate samples that are in the same distribution as its validation set. This value can be tuned in practice using a validation metric like that in section A.6.1.

A.6.3 Strategy For MINs

The main tunable components of MINs [66] are the learned objective function, and the generative model fit to the data distribution. The learned objective function is typically trained using a maximum likelihood objective, and the validation log-likelihood (or regression error) can be directly tracked. The learned objective function should train until a minimum validation loss is reached, which ensured that the model will generalize as well as possible beyond its training set. Since only the static task dataset is used for this—it may be split into train/validation sets—this tuning strategy is fully offline.

The generative model for MINs is an inverse mapping $\mathbf{x} = f^{-1}(y, \mathbf{z})$, conditioned on the objective value y. Training conditional generative models is considerable less stable than unconditional generative models, so in addition to monitoring the fit of a validation set recommended in section A.6.1, it is also necessary to track the extent of the dependence of the generative model's predictions on the objective value y. This can be evaluated in practice by comparing the distribution of x from the conditional generative model $p(\mathbf{x}|y)$ to an unconditional generative model $p(\mathbf{x})$ with an identical initialization, or by comparing if $p(\mathbf{x}|y)$ is independent of y by querying the inverse model for different values of y and visualizing the similarity in the predictions of $\mathbf{x}$. One metric for more formally studying the extent of the dependence of $\mathbf{x}$ on $\mathbf{z}$ is the mutual information $I(\mathbf{x}; \mathbf{z})$. The conditional generative model has an appropriate fit if for some positive threshold c we have that $I(\mathbf{x}; \mathbf{z}) > c$.

A.6.4 Strategy For Gradient Ascent

The main tunable components of Gradient Ascent MBO methods are the learned objective function, and the parameters for gradient ascent. The learned objective function is typically trained using a maximum likelihood objective under a Gaussian distribution, and the methodology for obtaining a high-performing learned objective function is identical to that in section A.6.3. The second aspect of gradient ascent MBO algorithms are the parameters of the gradient-based optimizer for the designs—such as its learning rate, and the number of gradient steps it performs. The learning rate should be small enough that the gradient steps taken increase the prediction of the learned objective function—if the learning rate is too large, gradient steps may not follow the path of steepest ascent of the objective function. The number of gradient steps is more difficult to tune. The strategy we used is a fixed number of steps, and an offline criterion to select this parameter is future work.

A.6.5 Strategy For REINFORCE

The main tunable components of REINFORCE-based MBO methods are the learned objective function, and the parameters for the policy gradient estimator. The learned objective function is typically trained using a maximum likelihood objective, and the methodology for obtaining a high-performing learned objective function is identical to that in section A.6.3. The remaining parameters to tune are specific to REINFORCE. The distribution of the policy should be carefully selected to be able to model the distribution of designs. For continuous MBO tasks, a Gaussian distribution is appropriate, and for discrete MBO tasks, a categorical distribution is appropriate. In addition, the learning rate, and optimizer should be selected so that policy updates improve the model-predicted score.

A.6.6 Strategy For Bayesian Optimization

The main tunable components of Bayesian Optimization MBO methods [7] are the learned objective function, and the parameters for the bayesian optimization loop. The learned objective function is typically trained using a maximum likelihood objective, and the methodology for obtaining a high-performing learned objective function is identical to that in section A.6.3. For a detailed review of the strengths and weaknesses of various Bayesian Optimization strategies and their hyperparameters, we refer the reader to the BoTorch documentation, available at the BoTorch website `https://botorch.org/docs/overview`. In this work we employ a Gaussian Process as the model, and the quasi-Monte Carlo Expected Improvement acquisition function, which has the advantage of scaling up to our high-dimensional optimization problems.

A.6.7 Strategy For Covariance Matrix Adaptation (CMA-ES)

The main tunable components of Covariance Matrix Adaptation MBO methods are the learned objective function, and the parameters for the evolution strategy. The learned objective function is typically trained using a maximum likelihood objective, and the methodology for obtaining a high-performing learned objective function is identical to that in Subsection A.6.3. For a detailed review of the strengths and weaknesses of various Bayesian Optimization strategies and their hyperparameters, we refer the reader to an open-source implementation of CMA-ES and its corresponding documentation `https://github.com/CMA-ES/pycma`. In this work we employ the default settings for CMA-ES reported in this open source implementation, with $\sigma = 0.5$.

A.6.8 Strategy For Conservative Objective Models (COMs)

Conservative Objective Models has three main tunable parameters, and we refer the reader to the original paper for a full experimental description [116]. The first parameter for COMs is the degree to which the objective model is allowed to overestimate the objective value for off-manifold designs. This parameter can be implemented as a constraint with threshold τ, or as a penalty with weight α. This parameter is chosen to be as high as possible, permitting high validation performance. When either τ or α imposes too much conservatism, this regularizes the objective model, and may lead the model to poorly fit the dataset $\mathcal{D}$. This parameter is uniformly chosen to be 2 for all discrete tasks and 0.5 for all continuous tasks. The second tunable parameter of COMs is the number of gradient ascent steps to perform when optimizing $\mathbf{x}$, and is uniformly chosen to be 50. The final parameter is the learning rate used when optimizing $\mathbf{x}$, which is uniformly chosen to be $2\sqrt{d}$ for all discrete tasks and $0.05\sqrt{d}$ for all continuous tasks, where d is the cardinality of the design space.

Appendix B

Appendix for Conservative Objective Models

B.1 Method Details

In this section we provide additional information about our method **conservative objective models (COMs)**. In this section, we provide a 50th percentile evaluation of COMs compared to other methods and discuss additional details for COMs including including hyperparameters. Finally, we discuss how the benchmarking tasks are curated.

B.1.1 Additional Results

In addition to reporting performance using the mean 100th percentile objective value, as in Table 3.1, we additionally provide a table measuring the mean 50th percentile objective value in Table B.1. This follows the convention for evaluation standardized by [114] when benchmarking model-based optimization algorithms. The 50th percentile results in Table B.1 confirm that COMs again is optimal in **4/7** tasks, the most of any method we tested, and attains a normalized average performance of **0.590**, the greatest normalized average performance of all baselines we tested.

B.1.2 Implementation details

In addition to various considerations from Sections 3.3.3 and 3.3.4, one important implementation detail of COMs is to normalize the inputs ($\mathbf{x}$) and outputs (y values)

	GFP	TF Bind 8	UTR	Norm. avg. perf.	# Optimal
$\mathcal{D}$ (**best**)	0.789	0.439	0.593		
Auto. CbAS	0.848 ± 0.007	0.419 ± 0.007	0.576 ± 0.011	0.441	0 / 7
CbAS	0.852 ± 0.004	0.428 ± 0.010	0.572 ± 0.023	0.444	0 / 7
MINs	0.820 ± 0.018	0.421 ± 0.015	0.585 ± 0.007	0.574	3 / 7
BO-qEI	0.246 ± 0.341	0.439 ± 0.000	0.571 ± 0.000	0.478	1 / 7
CMA-ES	0.047 ± 0.000	0.537 ± 0.014	**0.612 ± 0.014**	0.311	1 / 7
Grad.	0.838 ± 0.004	0.609 ± 0.019	0.593 ± 0.006	0.464	1 / 7
Grad. Min	0.837 ± 0.001	**0.645 ± 0.030**	0.598 ± 0.005	0.529	2 / 7
Grad. Mean	0.838 ± 0.002	**0.616 ± 0.023**	**0.601 ± 0.003**	0.528	3 / 7
REINFORCE	0.844 ± 0.003	0.462 ± 0.021	0.568 ± 0.017	0.395	1 / 7
COMs (Ours)	**0.864 ± 0.000**	0.497 ± 0.038	**0.608 ± 0.012**	**0.590**	**4 / 7**

	Superconductor	Ant Morphology	D'Kitty Morphology	Hopper Controller	
$\mathcal{D}$ (**best**)	0.399	0.565	0.884	1.0	
Auto. CbAS	0.131 ± 0.010	0.364 ± 0.014	0.736 ± 0.025	0.019 ± 0.008	
CbAS	0.111 ± 0.017	0.384 ± 0.016	0.753 ± 0.008	0.015 ± 0.002	
MINs	0.336 ± 0.016	**0.618 ± 0.040**	**0.887 ± 0.004**	**0.352 ± 0.058**	
BO-qEI	0.300 ± 0.015	0.567 ± 0.000	**0.883 ± 0.000**	0.343 ± 0.010	
CMA-ES	0.379 ± 0.003	-0.045 ± 0.004	0.684 ± 0.016	-0.033 ± 0.005	
Grad.	**0.476 ± 0.022**	0.134 ± 0.018	0.509 ± 0.200	0.092 ± 0.084	
Grad. Min	**0.471 ± 0.016**	0.185 ± 0.008	0.746 ± 0.034	0.222 ± 0.065	
Grad. Mean	**0.469 ± 0.022**	0.187 ± 0.009	0.748 ± 0.024	0.243 ± 0.064	
REINFORCE	**0.463 ± 0.016**	0.138 ± 0.032	0.356 ± 0.131	-0.064 ± 0.003	
COMs (Ours)	0.386 ± 0.018	0.519 ± 0.026	**0.885 ± 0.003**	**0.375 ± 0.003**	

Table B.1: Comparative evaluation of COMs against prior methods in terms of the mean 50th-percentile score and its standard deviation over 8 trials. Tasks include Superconductor, Hopper-Controller, AntMorphology, and DKittyMorphology, which have a continuous design input space and GFP, TFBind8 and UTR with a discrete design input space.

for training the conservative model, $\hat{f}_\theta(\mathbf{x})$. Our motivation for using normalization was simple: Since the input and output ranges and modalities of various tasks we evaluated on in Table 3.1 is very different from each other, in order to be able to use a *uniform* set of hyperparameters for COMs, it is necessary to normalize both the inputs $\mathbf{x}$ and outputs y to a standard range. Following standard normalization practices, we normalized $\mathbf{x}$ and y such that the resulting first and second moments match those of a unit Gaussian distribution. In practice, this means collecting all objective values from the training dataset into a vector $Y \in \mathbb{R}^{N\times 1}$, evaluating the sample mean $\hat{\mu} = \text{mean}(Y)$ and sample standard deviation $\hat{\sigma} = \text{std}(Y - \hat{\mu})$. A similar procedure is used for calculating the sample mean and sample standard deviation of $\mathbf{x}$. The objective values and inputs are then normalized by subtracting their sample mean and dividing by their sample standard deviation $y \leftarrow (y - \hat{\mu})/\hat{\sigma}$, except where doing so would divide by zero. This normalization allows COMs to use the uniform set of hyperparameters, which we mention explicitly, in Table B.2.

Hyperparameter	Discrete	Continuous
Number of epochs to train $\hat{f}_\theta$	50	50
T (Number of gradient ascent steps using Equation 3.5)	50	50
Number of steps used to generate adversarial samples $\mu(\mathbf{x})$ in Equation 3.6	50	50
α learning rate (used to optimize Equation 3.6 via dual gradient descent)	0.01	0.01
τ in Equation 3.6	2.0	0.5
η in Equation 3.5	$2.0\sqrt{d}$	$0.05\sqrt{d}$

Table B.2: Hyperparameters for COMs. All hyperparameters are kept constant across all discrete tasks and continuous tasks respectively in COMs. The variable d indicates the cardinality of a single design $\mathbf{x}$ in the training set of the model. Scaling the learning rate by a factor proportional to $\sqrt{d}$ follows the implementation of the Gradient ascent baseline from Trabucco et al. [114]

B.1.3 Benchmarking Details

In order to promote reproducibility, we additionally provide the task identifiers and keyword arguments used with the *design-bench* Trabucco et al. [114] package. These arguments are passed to the `design_bench.make` function call in order to build a model-based optimization Task object in Python. Note that in addition to specifying the name of the task dataset (such as GFP), one must also specify the desired oracle function (such as a Transformer). In Table B.3 we detail the specific combination of task datasets and oracle functions used in this work. Additionally, when an approximate oracle is used, commonly because an exact simulator or closed form equation for the ground truth y values is not available, there is a train-test discrepancy, where the predictions of the approximate oracle may not be a perfect reflection of the ground truth y values contained in the original model-based optimization dataset. This discrepancy is further explored by Trabucco et al. [114]; however, we find that UTR is particular susceptible to such discrepency, and so we choose to relabel the y values contained in the MBO dataset with the predictions of the CNN oracle. See Appendix B.4 for more information.

Task	Design-Bench ID	Relabel
GFP	`GFP-Transformer-v0`	False
TF Bind 8	`TFBind8-Exact-v0`	False
UTR	`UTR-ResNet-v0`	True
Superconductor	`Superconductor-RandomForest-v0`	False
Ant Morphology	`AntMorphology-Exact-v0`	False
D'Kitty Morphology	`DKittyMorphology-Exact-v0`	False
Hopper Controller	`HopperController-Exact-v0`	False

Table B.3: Design-Bench task identifiers. This table contains the necessary arguments to pass to the `design_bench.make` function call. More information is available at `https://github.com/brandontrabucco/design-baselines`, and documentation for Design-Bench is available at `https://github.com/brandontrabucco/design-bench`

B.2 Proof of Theorem 3.4.1

In this section, we provide a proof for Theorem 3.4.1 and show that the conservative training by performing gradient descent on θ with respect to the objective in Equation 3.8 (restated below in a more convenient form as Equation B.1) indeed obtains a conservative model of the actual objective function. Note that $\overline{\mathcal{D}}(\mathbf{x}'|\mathbf{x})$ denotes a smoothed Dirac-delta distribution centered at $\mathbf{x}$, which can be obtained by adding random noise to a given $\mathbf{x}$.

$$\mathcal{L}(\theta; \mu, \overline{\mathcal{D}}) := \alpha \left(\mathbb{E}_{\mathbf{x}_0 \sim \overline{\mathcal{D}}, \mathbf{x}_T \sim \mu(\mathbf{x}_T|\mathbf{x}_0)} \left[\hat{f}_\theta(\mathbf{x}_T) \right] - \mathbb{E}_{\mathbf{x} \sim \overline{\mathcal{D}}, \mathbf{x}_T \sim \overline{\mathcal{D}}(\mathbf{x}_T|\mathbf{x}_0)} \left[\hat{f}_\theta(\mathbf{x}_T) \right] \right) \tag{B.1}$$

$$+ \underbrace{\frac{1}{2} \mathbb{E}_{\mathbf{x}_0 \sim \overline{\mathcal{D}}, (\mathbf{x}, y) \sim \overline{\mathcal{D}}(\mathbf{x}|\mathbf{x}_0)} \left[\left(\hat{f}_\theta(\mathbf{x}) - y \right)^2 \right]}_{:= (\wedge)}. \tag{B.2}$$

We now restate a formal version of Theorem 3.4.1, and then provide a proof. We make an additional assumption that the neural tangent kernel, $\mathbf{G}_f^k(\mathbf{x}, \mathbf{x}')$, is semi-positive definite.

Theorem B.2.1 (Formal version of Theorem 3.4.1)**.** *Assume that $\hat{f}_\theta(\mathbf{x})$ is trained by performing gradient descent on θ with respect to the objective $\mathcal{L}(\theta; \mu, \overline{\mathcal{D}})$ in Equation B.1 with a learning rate η. The parameters in step k of gradient descent are denoted by θ^k, and let the corresponding conservative model be denoted as $\hat{f}_\theta^k$. Let $\mathbf{G}$, μ, $\widehat{L}$, L, $\overline{\mathcal{D}}$ be defined as discussed above. Then, under assumptions listed above, $\forall\ \mathbf{x} \in \mathcal{D}, \mathbf{x}'' \in \mathcal{X}$, the conservative model at iteration $k+1$ of training satisfies:*

$$\hat{f}_\theta^{k+1}(\mathbf{x}'') := \max \Big\{ \hat{f}_\theta^{k+1}(\mathbf{x}) - \widehat{L} ||\mathbf{x}'' - \mathbf{x}||_2, \tilde{f}_\theta^{k+1}(\mathbf{x}'') - \\ \eta\alpha \mathbb{E}_{\mathbf{x} \sim \overline{\mathcal{D}}, \mathbf{x}' \sim \mu}[\mathbf{G}_f^k(\mathbf{x}'', \mathbf{x}')] + \eta\alpha \mathbb{E}_{\mathbf{x} \sim \overline{\mathcal{D}}, \mathbf{x}' \sim \overline{\mathcal{D}}}[\mathbf{G}_f^k(\mathbf{x}'', \mathbf{x}')] \Big\},$$

where $\tilde{f}_\theta^{k+1}(\mathbf{x}'')$ is the resulting $(k+1)$-th iterate of $\hat{f}_\theta$ if conservative training were not used. Thus, if α is sufficiently large, the expected value of the asymptotic function, $\hat{f}_\theta := \lim_{k \to \infty} \hat{f}_\theta^k$, on inputs $\mathbf{x}_T$ found by the optimizer, lower-bounds the value of the true function $f(\mathbf{x}_T)$:

$$\mathbb{E}_{\mathbf{x}_0 \sim \mathcal{D}, \mathbf{x}_T \sim \mu(\mathbf{x}_T|\mathbf{x}_0)}[\hat{f}_\theta(\mathbf{x}_T)] \leq \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{D}, \mathbf{x}_T \sim \mu(\mathbf{x}_T|\mathbf{x}_0)}[f(\mathbf{x})].$$

Proof. For proving the first part of the theorem, we first derive the expression for the gradient of $\mathcal{L}(\theta; \mu, \overline{\mathcal{D}})$ with respect to θ, and denote the y-value for a given $\mathbf{x}$ as a deterministic function $y(\mathbf{x})$. Our proof can directly be extended to a non-deterministic $y(\mathbf{x})$ with an additional integral over y values, but we stick to deterministic $y(\mathbf{x})$ for

simplicity.

$$\nabla_\theta \mathcal{L}(\theta; \mu, \overline{\mathcal{D}}) = \alpha \int \left(\overline{\mathcal{D}}(\mathbf{x}_0)\mu(\mathbf{x}|\mathbf{x}_0) - \overline{\mathcal{D}}(\mathbf{x}_0)\overline{\mathcal{D}}(\mathbf{x}|\mathbf{x}_0)\right) \nabla_\theta \hat{f}_\theta(\mathbf{x}) \; d\mathbf{x}_0 d\mathbf{x} + \int \overline{\mathcal{D}}(\mathbf{x}_0)\overline{\mathcal{D}}(\mathbf{x}|\mathbf{x}_0)(f_\theta(\mathbf{x}) - y(\mathbf{x}))\nabla_\theta \hat{f}_\theta(\mathbf{x}) \; d\mathbf{x} d\mathbf{x}_0.$$

At any iteration k of gradient descent, the next parameter iterate θ^{k+1} are obtained via, $\theta^{k+1} = \theta^k - \eta \nabla_\theta \mathcal{L}(\theta; \mu, \overline{\mathcal{D}})$. Using this relation, and making an approximate linearization assumption on the non-linear function $\hat{f}_\theta^k$ for a small learning rate $\eta << 1$ under the assumption of the neural tangent kernel (NTK) [56] regime, which models the behavior of deep neural networks in the infinite-width limit, we obtain the expression for the next function value: $\hat{f}_\theta^{k+1}(\mathbf{x}'')$:

$$\begin{aligned}
&\hat{f}_\theta^{k+1}(\mathbf{x}'') \approx \hat{f}_\theta^k(\mathbf{x}'') + (\theta^{k+1} - \theta^k)^T \nabla_\theta \hat{f}_\theta^k(\mathbf{x}'') \\
&= \underbrace{\hat{f}_\theta^k(\mathbf{x}'') + \eta \mathbb{E}_{\mathbf{x} \sim \overline{\mathcal{D}}, \mathbf{x}' \sim \overline{\mathcal{D}}}[\Big(y(\mathbf{x}') - \hat{f}_\theta^k(\mathbf{x}')\Big) \mathbf{G}_f^k(\mathbf{x}'', \mathbf{x}')]}_{:=(*)} \\
&- \underbrace{\Big(\eta\alpha \mathbb{E}_{\mathbf{x} \sim \overline{\mathcal{D}}, \mathbf{x}' \sim \mu}[\mathbf{G}_f^k(\mathbf{x}'', \mathbf{x}')] - \eta\alpha \mathbb{E}_{\mathbf{x} \sim \overline{\mathcal{D}}, \mathbf{x}' \sim \overline{\mathcal{D}}}[\mathbf{G}_f^k(\mathbf{x}'', \mathbf{x}')]\Big)}_{:=\Delta(\mathbf{x}'')},
\end{aligned}$$

where the expression marked as $(*)$ denotes the $(k+1)$-th iterate of the function, under gradient descent on just the mean-squared error $(f(\mathbf{x}) - y)^2$ term, marked as $(\wedge)$ in Equation B.1. Noting that the theorem statement denotes the term $(*)$ as $\tilde{f}_\theta^{k+1}(\mathbf{x}'')$, we obtain our first desired result. To obtain the first argument of the max in the theorem statement, note that if the function $\hat{f}_\theta^{k+1}$ is $\widehat{L}$-Lipschitz, the value at $\mathbf{x}''$ cannot be smaller than $\hat{f}_\theta^{k+1}(\mathbf{x}) - \widehat{L}||\mathbf{x} - \mathbf{x}''||_2$, and hence the maximum over the two terms.

For proving the second part of the theorem statement, observe that if we can show that in expectation over $\mathbf{x}'' \sim \mu(\mathbf{x}_T); \mu(\mathbf{x}_T) := \int_{\mathbf{x}_0} \overline{\mathcal{D}}(\mathbf{x}_0)\mu(\mathbf{x}_T|\mathbf{x}_0) \; d\mathbf{x}_0$, the quantity $\Delta(\mathbf{x}'')$ is positive, then our argument is complete since we have shown that each step of gradient descent on θ reduces the value of $\mathbb{E}_{\mathbf{x}_0 \sim \overline{\mathcal{D}}, \mathbf{x}_T \sim \mu(\mathbf{x}_T|\mathbf{x}_0)}[\hat{f}_\theta^k(\mathbf{x}_T)]$ by a positive quantity by virtue of training with Equation B.1 as compared to only training θ with standard squared error $(\wedge)$. Thus, if $\mathbb{E}_{\mathbf{x}_0 \sim \overline{\mathcal{D}}, \mathbf{x}_T \sim \mu(\mathbf{x}_T|\mathbf{x}_0)}[\Delta^k(\mathbf{x}_T)]$ is positive for all gradient descent steps k, we obtain the desired lower-bound condition as $k \to \infty$. As an additional detail, note that we assumed $\widehat{L} >> L$ (i.e. the Lipschitz constant of $\hat{f}_\theta(\mathbf{x})$ is sufficiently larger than that of $f(\mathbf{x})$). This condition handles the boundary case when the predictions $\hat{f}_\theta^{k+1}(\mathbf{x}')$ get lower-bounded under the first argument of max in the first part of Theorem B.2.1 due to the Lipschitz condition: $\hat{f}_\theta^{k+1}(\mathbf{x}) - \widehat{L}||\mathbf{x}' - \mathbf{x}||_2$.

Finally, we fill in the missing piece that show $\mathbb{E}_{\mathbf{x}_0 \sim \overline{\mathcal{D}}, \mathbf{x}_T \sim \mu(\mathbf{x}_T|\mathbf{x}_0)}[\Delta^k(\mathbf{x}_T)]$ is positive for each k. Under the assumption that the neural tangent kernel $\mathbf{G}^k(\mathbf{x}, \mathbf{x}')$ is semi-positive

definite for all k, we can express:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x}_0 \sim \overline{\mathcal{D}}, \mathbf{x}_T \sim \mu(\mathbf{x}_T|\mathbf{x}_0)}[\Delta^k(\mathbf{x}_T)] \\
&:= \eta\alpha \int_{\mathbf{x},\mathbf{x}_0,\mathbf{x}',\mathbf{x}_T} \left[\overline{\mathcal{D}}(\mathbf{x})\mu(\mathbf{x}'|\mathbf{x}) - \overline{\mathcal{D}}(\mathbf{x})\overline{\mathcal{D}}(\mathbf{x}'|\mathbf{x})\right] \overline{\mathcal{D}}(\mathbf{x}_0)\mu(\mathbf{x}_T|\mathbf{x}_0)\mathbf{G}_f^k(\mathbf{x}', \mathbf{x}_T) \\
&= \eta\alpha \int_{\mathbf{x}_0} \overline{\mathcal{D}}(\mathbf{x}_0) \int_{\mathbf{x}} \overline{\mathcal{D}}(\mathbf{x}) \int_{\mathbf{x}',\mathbf{x}_T} [\mu(\mathbf{x}'|\mathbf{x}) - \overline{\mathcal{D}}(\mathbf{x}'|\mathbf{x})]\mu(\mathbf{x}_T|\mathbf{x}_0)\mathbf{G}_f^k(\mathbf{x}', \mathbf{x}_T)
\end{aligned}$$

By now writing the above in matrix form, we note that the RHS of the above equation has the same structure as the second term in the RHS of Equation 14 in Kumar et al. [67], and furthermore since $\mathbf{G}_f^k$ is positive semi-definite, it satisfies the required conditions for Equation 14 and Theorem D.1 from Kumar et al. [67] to be applicable. Thus, exactly following the proof of Theorem D.1 in Kumar et al. [67] for the linear function approximation case in reinforcement learning, with the following substitutions: $P_{\mathbf{F}} := \mathbf{G}_f^k(\cdot, \mathbf{x}_T)$ (i.e., a column of the kernel Gram-matrix for a fixed value of the second argument) and $a = \mathbf{x}_T$, $s = \mathbf{x}_0$, we can show that $\mathbb{E}_{\mathbf{x}_0 \sim \overline{\mathcal{D}}, \mathbf{x}_T \sim \mu(\mathbf{x}_T|\mathbf{x}_0)}[\Delta^k(\mathbf{x}_T)] \geq 0$, thus finishing our argument. □

B.3 Network Details

In each of our experiments, we train a neural network $\hat{f}_\theta$ to approximate the ground truth score function of an offline MBO task, where θ represents the weights of the model. Distinct from prior methods based on generative models [66, 12] we are able to utilize the same neural network architecture for representing the learned model, $\hat{f}_\theta(\mathbf{x})$ across all MBO tasks. This architecture is a three-layer neural network with two hidden layers of size 2048, followed by Leaky ReLU activation functions with a leak of 0.3. Each neural network $\hat{f}_\theta$ has an output layer that predicts a single scalar objective value y, which is used for regression. Specifically, $\hat{f}_\theta$ is trained to minimize the mean squared error of observed objective values, using the default parameters of the Adam optimizer as discussed in Section 3.3.4.

B.4 Data Collection

In this section, we detail the data collection steps used for creating each of the tasks from [114], used for benchmarking COMs. We answer **(1)** where is the data from, and **(2)** what pre-processing steps are used?

B.4.1 TF Bind 8

The TF Bind 8 task is a derivative of the transcription factor binding activity survey performed by Barrera et al. [8], where the binding activity scores of every possible length

eight DNA sequence was measured with a variety of human transcription factors. We filter the dataset by selecting a particular transcription factor `SIX6_REF_R1`, and defining an optimization problem where the goal is to synthesize a length 8 DNA sequence with high binding activity with human transcription factor `SIX6_REF_R1`. This particular transcription factor for TF Bind 8 was recently used for optimization in Angermueller et al. [4, 3]. TF Bind 8 is a fully characterized dataset containing 65792 samples, representing every possible length 8 combination of nucleotides $\mathbf{x}_{\text{TFBind8}} \in \{0, 1\}^{8\times 4}$. The training set given to offline MBO algorithms is restricted to the bottom 50%, which results in a visible training set of 32898 samples.

B.4.2 GFP

The GFP task provided is a derivative of the GFP dataset [97]. The dataset we use in practice is that provided by Brookes et al. [12] at the url `https://github.com/dhbrookes/CbAS/tree/master/data`. We process the dataset such that a single training example consists of a protein represented as a tensor $\mathbf{x}_{\text{GFP}} \in \{0, 1\}^{237\times 20}$. This tensor is a sequence of 237 one-hot vectors corresponding to which amino acid is present in that location in the protein. We use the dataset format of [12] with no additional processing. The data was originally collected by performing laboratory experiments constructing proteins similar to the Aequorea victoria green fluorescent protein and measuring fluorescence. We employ the full dataset of 56086 proteins when learning approximate oracles for evaluating offline MBO methods, but restrict the training set given to offline MBO algorithms to 5000 samples drawn from between the 50th percentile and 60th percentile of proteins in the GFP dataset, sorted by fluorescence values. This subsampling procedure is consistent with the procedure used by prior work [12].

B.4.3 UTR

The UTR task is derived from work by Sample et al. [95] who trained a CNN model to predict the expressive level of a particular gene from a corresponding 5‘UTR sequence. Our use of the UTR task for model-based optimization follows Angermueller et al. [3], where the goal is to design a length 50 DNA sequence to maximize expression level. We follow the methodology set by Sample et al. [95] to sort all length 50 DNA sequences in the unprocessed UTR dataset by total reads, and then select the top 280,000 DNA sequences with the most total reads. The result is a dataset containing 280,000 samples of length 50 DNA sequences $\mathbf{x}_{\text{UTR}} \in \{0, 1\}^{50\times 4}$ and corresponding ribosome loads. When training offline MBO algorithms, we subsequently eliminate the top 50% of sequences ranked by their ribosome load, resulting in a visible dataset with only 140,000 samples.

B.4.4 Superconductor

The Superconductor task is inspired by recent work [29] that applies offline MBO to optimize the properties of superconducting materials for high critical temperature. The data we provide in our benchmark is real-world superconductivity data originally collected by [48], and subsequently made available to the public at `https://archive.ics.uci.edu/ml/datasets/Superconductivty+Data#`. The original dataset consists of superconductors featurized into vectors containing measured physically properties like the number of chemical elements present, or the mean atomic mass of such elements. One issue with the original dataset that was used in [29] is that the numerical representation of the superconducting materials did not lend itself to recovering a physically realizable material that could be synthesized in a lab after performing model-based optimization. In order to create an *invertible* input specification, we deviate from prior work and encode superconductors as vectors whose components represent the number of atoms of specific chemical elements present in the superconducting material—a serialization of the chemical formula of each superconductor. The result is a real-valued design space with 86 components $\mathbf{x}_{\text{Superconductor}} \in \mathcal{R}^{86}$. The full dataset used to learn approximate oracles for evaluating MBO methods has 21263 samples, but we restrict this number to 17010 (the 80th percentile) for the training set of offline MBO methods to increase difficulty.

B.4.5 Hopper Controller

The goal of the Hopper Controller task is to design a set of weights for a neural network policy that achieves high expected return. The data collected for HopperController was taken by training a three layer neural network policy with 64 hidden units and 5126 total weights on the Hopper-v2 MuJoCo task using Proximal Policy Optimization [99]. Specifically, we use the default parameters for PPO provided in stable baselines [52]. The dataset we provide with this benchmark has 3200 unique weights. In order to collect this many, we run 32 experimental trials of PPO, where we train for one million steps, and save the weights of the policy every 10,000 environment steps. The policy weights are represented originally as a list of tensors. We first traverse this list and flatten each of the tensors, and we then concatenate each of these flattened tensors into a single training example $\mathbf{x}_{\text{Hopper}} \in \mathcal{R}^{5126}$. The result is an optimization problem over neural network weights. After collecting these weights, we perform no additional pre-processing steps. In order to collect scores we perform a single rollout for each x using the Hopper-v2 MuJoCo environment. The horizon length for training and evaluation is limited to 1000 simulation time steps, which is standard practice for this MuJoCo environment.

B.4.6 Ant & D'Kitty Morphology

Both morphology tasks share methodology. The goal of these tasks is to design the morphology of a quadrupedal robot—an ant or a D'Kitty—such that the agent is able to crawl quickly in a particular direction. In order to collect data for this environment, we create variants of the MuJoCo Ant and the ROBEL D'Kitty agents that have parametric morphologies. The goal is to determine a mapping from the morphology of the agent to the average return of a pre-trained morphology conditioned agent. We implement this by pre-training a morphology conditioned neural network policy using SAC [45]. For both the Ant and the D'Kitty, we train the agents for more than ten million environment steps, and a maximum episode length of 200, with all other settings as default. These agents are pre-trained on Gaussian distributions of morphologies. The Gaussian distributions are obtained by adding Gaussian noise with standard deviation 0.03 for Ant and 0.01 for D'Kitty the design-space range to the default morphologies.

After obtaining trained morphology-conditioned policies, we create a dataset of morphologies for model-based optimization by sampling initialization points randomly, and then using CMA-ES to optimize for morphologies that attain high reward using the pretrained morphology-conditioned policy. To obtain initialization points, we add Gaussian random noise to the default morphology for the Ant with standard deviation 0.075 and D'Kitty with standard deviation 0.1, and then apply CMA-ES with standard deviation 0.02. We ran CMA-ES for 250 iterations and then restart, until a minimum of 25000 morphologies were collected, resulting in a final dataset size of 25009 for both the Ant and D'Kitty. The design space for Ant Morphologies is $\mathbf{x}_{\text{Ant}} \in \mathcal{R}^{60}$, whereas for D'Kitty morphologies is $\mathbf{x}_{\text{D'Kitty}} \in \mathcal{R}^{56}$. We subsample the dataset to its 40th percentile when training offline MBO algorithms, resulting in 10004 samples.

B.5 Oracle Functions

We detail oracle functions for evaluating ground truth scores for each task. A common thread is that the oracle, if trained, is fit to a larger static dataset containing higher performing designs than observed by a downstream MBO algorithm.

B.5.1 TF Bind 8

TF Bind 8 is a fully characterized discrete offline MBO task, which means that all possible designs have been evaluated [8] and are contained in the full hidden TF Bind 8 dataset. The oracle for TF Bind 8 is therefore implemented as a lookup table that returns the score corresponding to a particular length 8 DNA sequence from the dataset. By restricting the size of the training set visible to an offline MBO algorithm, it is possible for the algorithm to propose a design that achieves a higher score than any

other DNA sequence visible to the offline MBO algorithm during training.

B.5.2 GFP

GFP uses a simplified Transformer to the TAPE Transformer proposed by Rao et al. [88]. The Transformer used has 4 attention blocks and a hidden size of 64. The Transformer is fit to the entire hidden GFP dataset, making it possible to sample a protein design that achieves a higher score than any other protein visible to an offline MBO algorithm. The model has a Spearman's rank correlation coefficient of 0.8497 with a held-out validation set derived from the GFP dataset.

B.5.3 UTR

UTR uses a CNN, which differs from the CNN that was originally used by [3] in that it has residual connections. Our reasoning for making this change is that ResNet is a newer and possibly higher capacity model that may be less prone to mistakes than the shallower CNN model proposed by Sample et al. [95]. The chosen CNN has 2 residual blocks with 2 convolution layer each, and a hidden size of 120. The CNN is fit to the entire hidden UTR dataset, making it possible to sample a DNA sequence that achieves a higher score than any other sequence visible to an offline MBO algorithm. The resulting CNN has a spearman's rank correlation coefficient of 0.8617 with a held-out validation set.

B.5.4 Superconductor

The Superconductor oracle function is also a random forest regression model. The model we use it the model described by [48]. We borrow the hyperparameters described by them, and we use the RandomForestRegressor provided in scikit-learn. Similar to the setup for the previous set of tasks, this oracle is trained on the entire hidden dataset of superconductors. The random forest has a spearman's rank correlation coefficient with a held-out validation set of 0.9155.

B.5.5 HopperController

HopperController and the remaining tasks implement an exact oracle function. For HopperController the oracle takes the form of a single rollout using the Hopper-v2 MuJoCo environment. The designs for HopperController are neural network weights, and during evaluation, a policy with those weights is instantiated—in this case that policy is a three layer neural network with 11 input units, two layers with 64 hidden units, and a final layer with 3 output units. The intermediate activations between layers are hyperbolic tangents. After building a policy, the Hopper-v2 environment is

reset and the reward for 1000 time-steps is summed. That summed reward constitutes the score returned by the Hopper Controller oracle. The limit of performance is the maximum return that an agent can achieve in Hopper-v2 over 1000 steps.

B.5.6 Ant & D'Kitty Morphology

The final two tasks in design-bench use an exact oracle function, using the MuJoCo simulator. For both morphology tasks, the simulator performs a rollout and returns the sum of rewards at every timestep in that rollout. Each task is accompanied by a pre-trained morphology-conditioned policy. To perform evaluation, a morphology is passed to the Ant or D'Kitty MuJoCo environments respectively, and a dynamic-morphology agent is initialized inside these environments. These simulations can be time consuming to run, and so we limit the rollout length to 100 steps. The morphology conditioned policies were trained using Soft Actor Critic for 10 million steps for each task, and are ReLU networks with two hidden layers of size 64.

Appendix C

Appendix for Crystal Structure Design

C.1 Additional Ablation Study

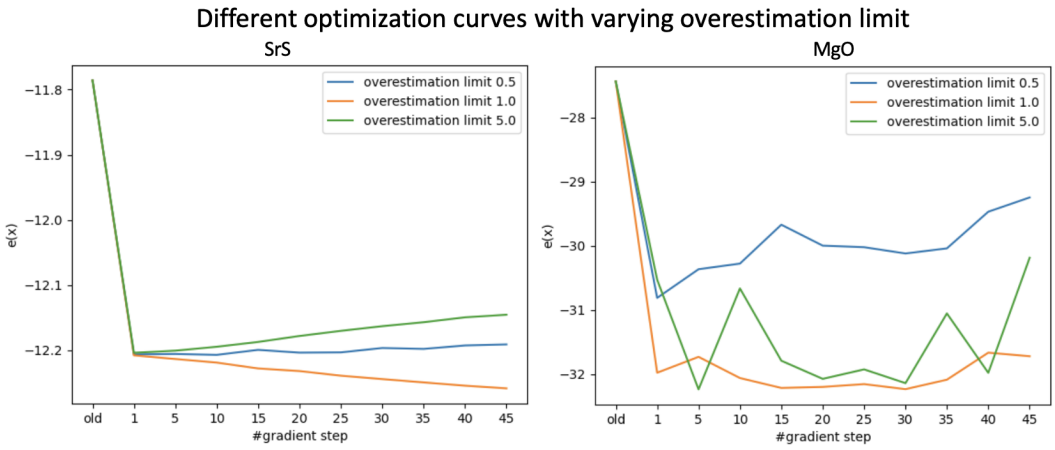


Figure C.1: Ablation study. The overestimation threshold, τ, is the factor controlling the level of conservatism imposed by LCOMs. The above plot shows the performance of the crystal structures found by LCOMs by varying this threshold for two sample compounds: SrS and MgO.

Since our method builds on existing conservative optimization algorithms, one of the main hyper parameters of our method is the coefficient α controlling the strength of the conservatism regularizer. In the practical instantiation of COMs [115], this hyperparameter is replaced by its Lagrangian dual Equation 6 in Trabucco et al. [115]), and the corresponding hyperparameter in the practical algorithm is τ, the threshold of

allowed over-estimation on adversarial examples (in our case, adversarial latent vectors). A smaller τ enforces a stricter upper bound on the allowed amount of distribution shift, whereas a larger τ does not penalize distributional shift. As a result, energies of produced designs would be close to the energy in the dataset when the coefficient τ is small, but also get exploited when τ is too large. An intermediate value of τ is expected to likely lead to the most favorable results.

As shown in Figure C.1, an intermediate value of τ (e.g., 1.0 in this case) leads to the best results as more gradient steps are performed to optimize the crystal structure. As expected, while a very small $\tau = 0.5$ plateaus in the case of MgO, a very large value $\tau = 5.0$ starts to get exploited for both the sample compounds, MgO and SrS. These results align with our hypothesis.

C.2 Details of Our Simulator

DFT simulators, underpinned by Density Functional Theory (DFT) [82], serve as crucial computational tools for approximating the solution of the Schrödinger equation for a given system of particles. Particularly, in our study, these particles constitute the chemical structure of a crystal. By providing an approximate solution of the underlying differential equation, DFT simulators enable the calculation of system dynamics, including critical properties such as total energy, and facilitate the simulation of system relaxation to a stable, energy-minimal configuration.

DFT represents a class of computational algorithms rather than a single operation method, which justifies the availability of multiple DFT simulators. Examples of these simulators include licensed platforms like VASP, and open-source ones like GPAW [77, 26]. We leveraged the operational flexibility inherent to DFT in this work by using GPAW to create random stable structures as initial points for the optimization process. Its accessibility as an open-source tool, and ease of integration with Python, made it the preferred choice.

However, GPAW does have limitations, most notably the absence of pseudo-potentials for all chemical elements, essential for approximating the potential experienced by valence electrons in atoms. This limitation hindered the simulation of some structures used for evaluation, as highlighted by Cheng et al. [17]. Consequently, our evaluation was limited to 25 out of the original 29 compounds discussed in this prior work that informed our evaluation procedure.

C.3 Experiment Details

In this section, we detail the hyperparameters and configurations employed in our experiments to facilitate reproducibility of the results. Please note that for competing

models, we rely on results reported in the original work instead of replicating the experiments. For comprehensive information regarding these models, we refer the reader to the work of [17].

Encoding and Decoding Following the method in [133], we firstly train an variational encoder to transform crystal structure to CD-VAE latent space, with the same training protocol in [133]. We use a batch size of 256 here when training the encoder.

Hyperparameters In LCOMs, we follow most of the hyper-parameters in the implementation of COMs method [115]. The number of epochs to train the model $\widehat{E}_\theta(\phi(\mathbf{x}, c), c)$ is 50 and the number of gradient descent steps used in Equation 4.5 is 50. The number of steps used to generate optimized results in latent space is 10 for model trained with OQMD dataset and 40 for model trained with MatBench dataset. Please note that this number is picked by evaluating the distance between optimized groups and training dataset. The training batch size is 128 and the learning rate for model training is 0.00003. The model structure is followed by the one in [115]. The overestimation limit τ in Equation 6 of [115] is picked as 1.0.

Appendix D

Appendix for Promoter Design

D.1 Additional Results

In this section we provide some additional results for the designed sequences from CPD and baseline methods. We present histograms of the differential expression levels of CPD, DENs and naïve gradient ascent across three target cell types in Figure D.1. We see that CPD is able to produce a large number of sequences beyond the best sequences in the dataset, outperforming the baseline methods.

D.2 Hyperparameters and Experiment Details

In this section we provide the details for the training setup and hyperparameters of CPD and various baselines. CPD is able to produce sequences with high differential expression levels, outpeforming DENs and naïve gradient ascent.

D.2.1 Details for Pre-training

In this section, we provide the details about the pre-trained model. Our model architecture consists of a 1D convolutional network and a Transformer network. Taking the one-hot encoding of the DNA sequence as input, the convolutional network consists of 3 convolution layers, with 256, 512 and 1024 units respectively. After each convolution layer, we apply GeLU activation [51], GroupNorm [132] and dropout [109] with probability 0.1. After the convolutional network, we concatenated the output with a learnable CLS embedding [22], and feed the concatenated sequence into a Transformer [121] network with 5 blocks. Each Transformer block has 1024 hidden dimensions and 8

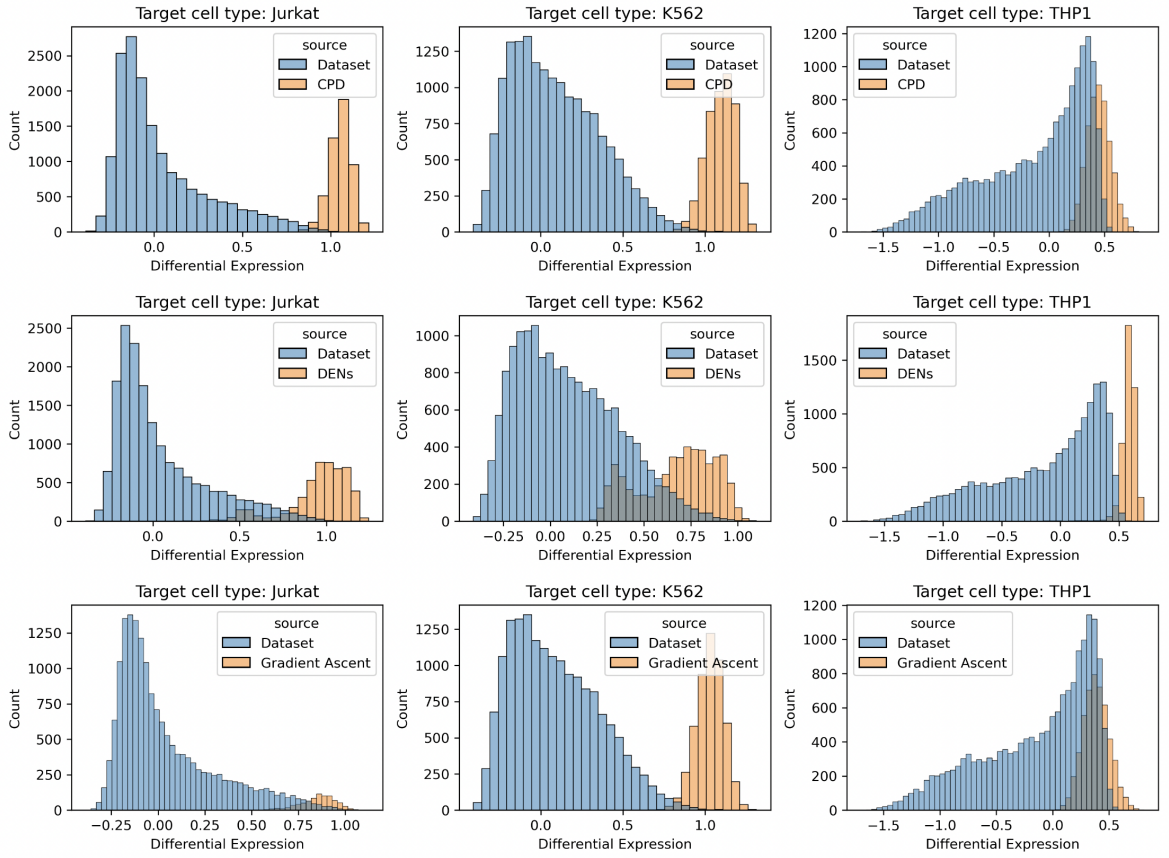


Figure D.1: Histogram for the oracle predicted differential expression level. We see that while all methods are able to improve the differential expression levels over the average of the dataset, CPD can push the objective beyond the best of the dataset reilably across three target cell types.

attention heads. We apply the RoPE position embedding [110] at each attention layer.

Following the pre-training process of Reddy et al. [89], we pre-train our model using the Sharpr-MPRA [28] and SuRE MPRA [118, 119]. During training, we use a batch size of 448, AdamW optimizer wtih learning rate 1e-4 and weight decay 3e-3. We train the model for 20000 steps in total, with a cosine learning rate decay schedule that decays to 0. The best checkpoint was selected using validation loss throughput the training process.

D.2.2 Details for Ensemble Oracle Model

In this section, we provide the details about the ensemble models we used for sequence selection and oracle evaluation. The sequence selection and evaluation oracle are two sets of ensemble models trained independently on random partitions of the dataset. For each model of the ensemble, we take the pre-trained model and apply three MLP heads

on top of the last layer embedding to predict the expression levels of the three cell types. Within the ensemble set, we vary the depth (2, 4 and 8 layers), hidden dimensions (215, 1024 and 2048) and activation functions (tanh, GeLU, ReLU and SiLU [51]) of the MLP heads. This gives us 36 ensembles models of different architectures in total in a set.

The ensemble models are finetuned with AdamW optimizer with batch size 512, learning rate 5e-5 and weight decay 3e-3. We finetune the model for 250 steps in total, with a cosine learning rate decay schedule that decays to 0.

D.2.3 Details for CPD

For CPD, similar to one of the ensemble models, we apply three heads with 2 hidden layers, 512 hidden dimensions, and GeLU activation to the last layer embedding of our pre-trained model. For the conservatism loss, we produce the adversarial example with gradient ascent optimizer in the one-hot encoding space, parameterized by a softmax. We perform 60 steps of gradient ascent using the Adam Optimizer with learning rate 0.5, and clip the result to a valid one-hot encoding. We train the model using the same hyperparamters as the ensemble models described in Appendix D.2.2.

During sequence generation, we start from each sequence in the dataset and optimize the sequence for 60 steps with the same gradient ascent optimizer we use during training time. We take the final sequence as the optimization result.

D.2.4 Details for Motif Tiling

In this section we provide the details for the motif tiling method for optimizing promoters. First, to identify motifs that might contribute to differential expression, we use FIMO [43] with default settings to detect instances of clustered TF-binding motifs defined by Vierstra et al. [122] [1] in the sequences assayed by Reddy et al. [89] and retain detected motif occurrences with q-value ¡ 0.01. Let's now consider designing a cell type-specific promoter for Jurkat. For every motif, we run 2 pairwise t-tests to determine if its presence leads to higher expression in Jurkat compared to K-562 or THP-1. Motifs that have positive effect sizes in both t-tests (i.e. leads to higher expression in Jurkat compared to both K-562 and THP-1) with q-values ¡ 0.05 are retained as those that could be contributing towards differential expression in Jurkat. Then, these motifs are used to design two sets of sequences - one set of sequences is designed by inserting the same motif into a background sequence as many times as possible while separating the motifs by 10bp and the second set is designed by randomly sampling motifs from the list of retained motifs and inserting as many of them as possible into a background sequence while separating the motifs by 10bp. While generating each sequence, the

[1] https://resources.altius.org/~jvierstra/projects/motif-clustering-v2.0beta/

background sequence is a randomly chosen sequence from those assayed by Reddy et al. [89] that exhibits a differential expression of at least 2. Inserted motif sequences are sampled from the motif's position weight matrix (PWM). The same process is repeated for K-562 and THP-1.

We discover 7, 45, and 10 motifs that may be causing differential expression in Jurkat, K-562 and THP-1 respectively. Thus, we get 7, 45, and 10 sequences in Jurkat, K-562 and THP-1 respectively by tiling the same motif repeatedly. Then, for each cell line, we design 5,000 sequences by randomly sampling motifs. Therefore, we get a total of 5007, 5045, and 5010 sequences for Jurkat, K-562 and THP-1 respectively using this design method.

D.2.5 Details for DENs

DENs are generative models that are trained to output diverse sequences that maximize a design model's predictions. The generator takes random noise as input and transforms it into a sequence PWM. We use a UNet-style [90] generator that first transforms the noise vector into a sequence PWM-sized matrix (i.e. of size (250, 4)). Then, it applies 6 downsizing convolutional layers followed by 5 upsizing convolutional layers. A final convolutional layer then pools information across the final set of filters' outputs to produce the sequence PWM. The PWM is used to sample sequences that are fed to the design model to get its predictions and a fitness-based loss that trains the DEN to output high-fitness sequences is computed. Additionally, to explicitly increase the diversity of the generated sequences, in every training step, random noise vectors are input to the DEN in pairs to get two sequence PWMs per pair. Then, a diversity-based loss is computed that incentivizes the sequences generated using the two different noise vectors to be distinct from each other, both in sequence and design model embedding space. An entropy-based loss is also minimized to reduce the entropy of the PWM output by the DEN at every position.

Thus, when training a DEN to generate cell type-specific promoters for a target cell $i \in \{\text{Jurkat, K-562, THP-1}\}$, the training objective we use is:

$$\min_{\phi} \quad \underbrace{-\left[\sum_{\substack{j=1,\\ s_j \sim g_\phi(u_1)}}^{2} \mathrm{DE}_\theta^i(s_j) + \sum_{\substack{j=1,\\ q_j \sim g_\phi(u_2)}}^{2} \mathrm{DE}_\theta^i(q_j)\right]}_{:= \text{ fitness loss}}$$

$$+ \beta_{\text{diversity}} \underbrace{\max\left[-0.3 + \max_{\sigma \in [0,10]} \frac{1}{N-\sigma}\left[\frac{1}{2} \sum_{\substack{j=1 \\ s_j \sim g_\phi(u_1) \\ q_j \sim g_\phi(u_2)}}^{2} \sum_{k=\sigma}^{N} s_{j,k}.q_{j,k-\sigma}\right], 0\right]}_{:= \text{ sequence-based diversity loss}}$$

$$+ \beta_{\text{diversity}} \underbrace{\max\left[-0.3 + \frac{1}{2} \sum_{\substack{j=1 \\ s_j \sim g_\phi(u_1) \\ q_j \sim g_\phi(u_2)}}^{2} \frac{R_\theta(s_j).R_\theta(q_j)}{||R_\theta(s_j)||.||R_\theta(q_j)||}, 0\right]}_{:= \text{ embedding-based diversity loss}}$$

$$+ \beta_{\text{entropy}} \underbrace{\max\left[1.8 - \frac{1}{N} \sum_{k=1}^{N} \left[\log_2 4 - \sum -g_\phi(u_1)_k \log_2\left(g_\phi(u_1)_k + 10^{-8}\right)\right], 0\right]}_{:= \text{ entropy loss}}$$

where ϕ is the set of trainable parameters of DEN g_ϕ which outputs $N = 250$ base pairs long sequence PWMs by taking u_1 or u_2 - 200-dimensional random noise vectors sampled from the uniform distribution over [-1, 1], as inputs. From the sequence PWMs output by g_ϕ, we sample two one-hot encoded sequences per noise vector denoted by s_j and q_j. These sequences are then input to the trained design model f_θ that predicts expression induced in each of the three cell types. Then, the predicted differential expression in the target cell i induced by a sequence $\mathbf{x}$ is given by $\mathrm{DE}_\theta^i(x) := f_\theta^i(\mathbf{x}) - \frac{1}{2}\sum_{j \neq i} f_\theta^j(\mathbf{x})$. The fitness loss maximizes this predicted differential expression. The other loss terms increase sequence diversity and reduce entropy in the sequence PWM. Here, $s_{j,k}$ is the one-hot encoded base pair at position k in s_j (similarly for $q_{j,k}$), $R_\theta(s_j)$ is an embedding for s_j extracted from the design model f_θ, $g_\phi(u_1)_k$ is the probability distribution over base pairs in the sequence PWM $g_\phi(u_1)$ at position k. Finally, the coefficients $\beta_{\text{diversity}}$ and β_{entropy} are used to weight the diversity and entropy losses relative to the fitness loss and to one another. They can be varied to regulate the diversity vs. fitness trade-off. We refer readers to the original work by Linder et al. [71] that proposed DENs for more details on the method. We tune the various hyperparameters reflected in the training objective by observing the overall quality of the generated sequences.