Conservative Objective Models for Effective Offline Model-Based Optimization

Brandon Trabucco [1] **Aviral Kumar** [1] **Xinyang Geng** [1] **Sergey Levine** [1]

Abstract

In this paper, we aim to solve data-driven model-based optimization (MBO) problems, where the goal is to find a design input that maximizes an unknown objective function provided access to only a static dataset of inputs and their corresponding objective values. Such data-driven optimization procedures are the only practical methods in many real-world domains where active data collection is expensive (e.g., when optimizing over proteins) or dangerous (e.g., when optimizing over aircraft designs, actively evaluating malformed aircraft designs is unsafe). Typical methods for MBO that optimize the input against a learned model of the unknown score function are affected by erroneous overestimation in the learned model caused due to distributional shift, that drives the optimizer to low-scoring or invalid inputs. To overcome this, we propose *conservative objective models* (COMs), a method that learns a model of the objective function which lower bounds the actual value of the ground-truth objective on out-of-distribution inputs and uses it for optimization. In practice, COMs outperform a number existing methods on a wide range of MBO problems, including optimizing controller parameters, robot morphologies, and superconducting materials.

1. Introduction

Black-box model-based optimization (MBO) problems are ubiquitous in a wide range of domains, such as protein (Brookes et al., 2019) or molecule design (Gaulton et al., 2012), designing controllers (Berkenkamp et al., 2016) or robot morphologies (Liao et al., 2019), optimizing neural network designs (Zoph & Le, 2017) and aircraft design (Hoburg & Abbeel, 2012). 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 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 a tight interaction loop between optimization and *active* data collection (Snoek et al., 2012). 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 favorable alternative approach for a broad range of MBO problems is to develop *data-driven* methods that can obtain optimized designs by training highly generalizable and expressive deep neural network models on previously collected datasets of inputs ($\mathbf{x}$) and their corresponding objective values (y), without access to the true function or any form of active data collection (Kumar & Levine, 2019). Moreover, in a number of these practical domains, such as protein (Sarkisyan et al., 2016) or molecule design (Gaulton et al., 2012), plenty of prior data already exists and can be utilized for completely offline model-based optimization.

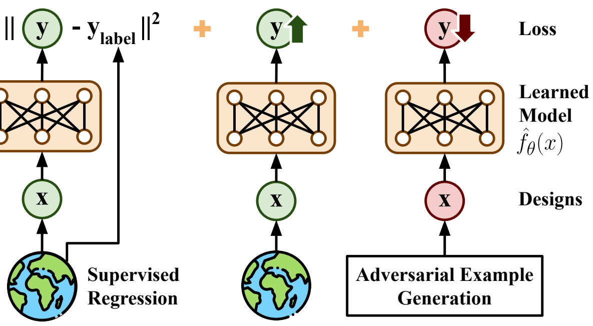

Figure 1. **Overview of COMs.** Our method trains a model of the objective function by training a neural net with supervised regression on the training data augmented two additional loss terms to obtian conservative predictions. These additional terms aim to maximize the predictions of the neural net 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.

[1]Department of Electrical Engineering and Computer Sciences, University of California Berkeley.. Correspondence to: Brandon Trabucco <btrabucco@berkeley.edu>, Aviral Kumar <aviralk@berkeley.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

Typical approaches for addressing MBO problems learn a model of the unknown objective function $\hat{f}$ that maps an input $\mathbf{x}$ (Snoek et al., 2012) (or a representation of the input (Gómez-Bombarelli et al., 2018)) to its objective value $\hat{f}(\mathbf{x})$ via supervised regression, and then 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 drive the optimization procedure towards off-manifold, invalid, and low-scoring inputs (Kumar & Levine, 2019), as these will falsely appear optimistic under the learned model. Since the offline MBO problem statement does not allow for any active data collection, it is unable to recover from poor solutions which it erroneously predicts high values.

If we can instead learn a *conservative* model of the objective function that does not overestimate the objective value on off-manifold out-of-distribution inputs, optimizing against this conservative model would produce the best solutions for which we are *confident* in the value, thus avoiding going off-manifold or far away from the training data In this paper, we propose a method to learn such *conservative objective models* (COMs), and then optimize the design against this conservative model using a trust-region constrained gradient-ascent procedure. Building on recent advances in offline reinforcement learning (Levine et al., 2020; Kumar et al., 2020), the key idea behind our method is to train the learned objective function model with an additional objective that minimizes its predictions on out-of-distribution inputs, thus ensuring it does not overestimate the objective value. We show that our approach mitigates overestimation near the manifold of the dataset in theory, and empirically, we find that this leads to good performance across a range of offline model-based optimization tasks.

The primary contribution of this paper 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: they only require training a model for the objective function, as opposed to other recent approaches that also train generative models either for generating optimized designs or for estimating the support of the data distribution. 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 six complex MBO tasks that span a wide range of real-world tasks including molecule substructure design, neural network parameter optimization, and superconducting material design. On some tasks, COMs outperform the *best* existing method *on that task* by a factor of **1.3-2x**.

2. Preliminaries

The goal in data-driven, offline model-based optimization (Kumar & Levine, 2019) is to find best possible solution, $\mathbf{x}^*$, to optimization problems of the form

$$\mathbf{x}^* \leftarrow \arg\max_{\mathbf{x}} \ f(\mathbf{x}), \tag{1}$$

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 (Gómez-Bombarelli et al., 2018; Brookes et al., 2019; Kumar & Levine, 2019; Fannjiang & Listgarten, 2020), 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 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 1 is gradient descent on the learned function, as given by

$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha \nabla_x \hat{f}_\theta(\mathbf{x})|_{x=\mathbf{x}_k}, \ \text{ for } \ k \in [1, T], \quad \mathbf{x}^\star = \mathbf{x}_T. \tag{2}$$

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. 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 2) to produce out-of-distribution and low-scoring inputs. We first discuss a procedure for learning such conservative estimates and then explain how these conservative models can be used for MBO.

3.1. Learning conservative objective models (COMs)

The key idea behind our approach is to augment the training of a objective model, $\hat{f}_\theta(\mathbf{x})$, typically trained using

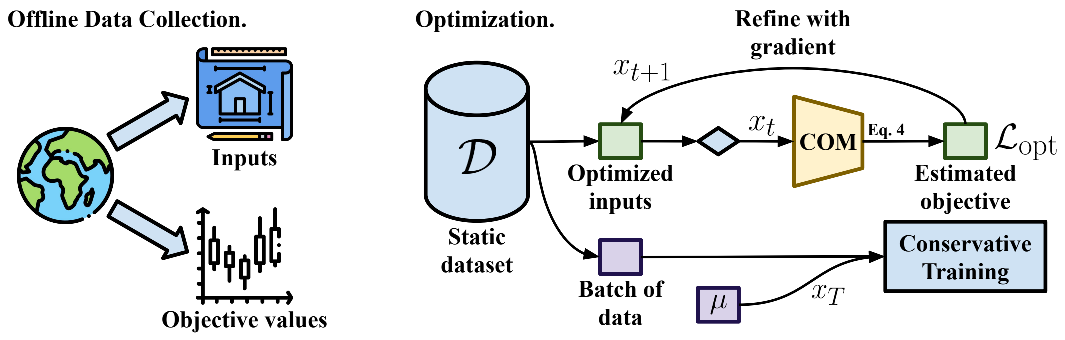

Figure 2. **Training and optimization using COMs.** The section on the left indicates that each task provides a static dataset that is collected offline without ayn MBO algorithm in-the-loop. The section on the right shows how a conservative objective model is used to produce promising optimized designs using gradient ascent, and how these designs are inputs to a conservative regularizer.

supervised regression, with a regularizer that minimizes the expected value of this function under an automatically chosen adversarial distribution $\mu(\mathbf{x})$. We will cover the construction of this adversarial distribution $\mu(\mathbf{x})$ formally, but intuitively it represents out-of-distribution inputs that are likely to be obtained during optimization and hence their values should be pushed down. While simply minimizing the function values under this adversarial distribution should effectively reduce the learned objective value at these inputs, our approach also additionally maximizes the expected value of this function under $\hat{D}(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{D}} \delta_{\mathbf{x}=\mathbf{x}_i}$, the empirical distribution of the inputs $\mathbf{x}$ in the dataset $\mathcal{D}$. This maximization term prevents excessive underestimation of the learned objective function that can arise *uniformly* over the entire input space when implementing COMs in practice with deep neural network function approximators. In Section 4, we will show that 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 design is inspired by recent work in offline RL (Kumar et al., 2020), where a similar approach is used to learn conservative value functions, though it has never been applied to MBO to the best of our knowledge. We will elaborate on this connection in Section 5. We will discuss the choice of $\mu(\mathbf{x})$ in the next paragraph. Formally, our training objective is given by the following equation, where α is a parameter that trades off conservatism for regression:

$$\hat{f}^\star_\theta \leftarrow \arg\min_{\theta \in \Theta} \ \alpha \left(\mathbb{E}_{\mathbf{x} \sim \mu(\mathbf{x})} \left[\hat{f}_\theta(\mathbf{x}) \right] - \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[\hat{f}_\theta(\mathbf{x}) \right] \right) + \frac{1}{2} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\left(\hat{f}_\theta(\mathbf{x}) - y \right)^2 \right], \tag{3}$$

The value of α and the choice of distribution $\mu(\mathbf{x})$ play a crucial role in determining the utility of this bound. If $\mu(\mathbf{x})$ is not chosen carefully, then the learned function $\hat{f}^\star_\theta$ may not be a lower-bound on the inputs $\mathbf{x}$ of interest, which is the design actually found by the optimizer. Similarly, if the chosen α is very small, then the resulting $\hat{f}^\star_\theta(\mathbf{x})$ 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.

Choosing $\mu(\mathbf{x})$. Our choice of $\mu(\mathbf{x})$ specifically focuses on points that the optimizer is likely to encounter while optimizing the input. While there are a number of ways to find such points, the approach we employ is to simply sample a starting point $\mathbf{x}_0$ from the dataset $\mathcal{D}$, and then perform several steps of gradient ascent on $\hat{f}_\theta$ starting from this point. Since the choice of $\mu(\mathbf{x})$ depends on θ, we reformulate the optimization over θ as an *adversarial* optimization problem, as shown below:

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

3.2. Using COMs for offline model-based optimization

Once we have a trained conservative model from Equation 3, we must use this learned model for finding the best possible input, $\mathbf{x}^\star$. Prior works (Kumar & Levine, 2019; Brookes et al., 2019) use $\hat{f}^\star_\theta$ 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. 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 $\hat{f}^\star_\theta$. However, since our method only prevents overestimation in a given region (specifically, for $\mathbf{x}$ values drawn from $\mu(\mathbf{x})$), we must make sure that the optimizer doesn't "outrun" this region, since conventional gradient ascent can move very far in even a few steps when gradients are large.

Algorithm 1 COM: Training Conservative Models

```
1: Initialize $\hat{f}_\theta$. Pick $\eta, \alpha$ 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 $\mathbf{x}_0$:
           $\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \nabla_\mathbf{x} \hat{f}_\theta(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_t}$;  $\mu(\mathbf{x}) = \sum_{\mathbf{x}_0 \in \mathcal{D}} \delta_{\mathbf{x}=\mathbf{x}_T(\mathbf{x}_0)}$.
5:     Minimize $\mathcal{L}(\theta; \alpha)$ with respect to $\theta$.
       $\mathcal{L}(\theta;\alpha) = \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{D}}(\hat{f}_\theta(\mathbf{x}_0) - y)^2 - \alpha \mathbb{E}_{\mathbf{x}_0}[\hat{f}_\theta(\mathbf{x}_0)] + \alpha \mathbb{E}_{\mu(\mathbf{x})}[\hat{f}_\theta(\mathbf{x})]$
       $\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}(\theta; \alpha)$
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 trust-region 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 $\mathcal{L}_{\text{opt}}(\mathbf{x}) := \hat{f}^\star_\theta(\mathbf{x}) - \beta \hat{f}^\star_\theta(\mathbf{x} + \eta \nabla_\mathbf{x} \hat{f}^\star_\theta(\mathbf{x}))$.
3: Return the solution $\mathbf{x}^\star = \mathbf{x}_T$.
```

We therefore incorporate a "trust-region" constraint into our optimizer. 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):

$$\forall\, t \in [T], \mathbf{x}_0 \in \mathcal{D}; \quad \mathbf{x}_{t+1} = \mathbf{x}_t + \eta \nabla_\mathbf{x} \mathcal{L}_{\text{opt}}(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_t}$$
$$\text{where } \mathcal{L}_{\text{opt}}(\mathbf{x}) := \hat{f}^\star_\theta(\mathbf{x}) - \beta \hat{f}^\star_\theta(\mathbf{x} + \eta \nabla_\mathbf{x} \hat{f}^\star_\theta(\mathbf{x})). \quad (4)$$

Equation 4 ensures that the value of the learned function $\hat{f}_\theta(\mathbf{x}_{t+1})$ is larger than the value at its previous iterate $\mathbf{x}_t$ and it additionally penalizes the value at a hallucinated, lookahead gradient-ascent iterate beyond $\mathbf{x}_{t+1}$ weighted by β. This β-weighted penalty ensures that the function $\hat{f}_\theta(\mathbf{x})$ is relatively constant near the obtained $\mathbf{x}_{t+1}$, and is expected to decrease the rate at which naïve gradient ascent can outrun the region defined by $\mu(\mathbf{x})$ in Equation 3, that the conservative model was trained on.

Equation 4 is theoretically equivalent (upto second order) to standard gradient ascent on learned $\hat{f}^*(\mathbf{x})$, with an additional regularizer on the norm of the gradient $\nabla_\mathbf{x} \hat{f}^*(\mathbf{x})$ (to observe this note the Taylor expansion of $\mathcal{L}_{\text{opt}}(\mathbf{x})$ in Equation 4). Thus, it implements a "slow" version of gradient ascent such that $\mathbf{x}^*$ doesn't outrun the region during optimization. A detailed discussion of this constraint along with how it is implemented in practice is provided in Appendix A.

3.3. Additional Design Decisions for COMs

Next we discuss other design decisions that appear in COMs training (Equation 3) or when optimizing the input against a learned conservative model (Equation 4).

Choosing α. The hyperparameter α in Equation 3 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 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:

$$\hat{f}^\star_\theta \leftarrow \arg\min_{\theta \in \Theta} \; \frac{1}{2} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\left(\hat{f}_\theta(\mathbf{x}) - y \right)^2 \right]$$
$$\text{s.t.} \; \left(\mathbb{E}_{\mathbf{x}\sim\mu(\mathbf{x})} \left[\hat{f}_\theta(\mathbf{x}) \right] - \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\hat{f}_\theta(\mathbf{x}) \right] \right) \leq \tau. \quad (5)$$

While Equation 5 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 objective values are normalized before use in Equation 5), and therefore a single choice works well across a range of tasks. We find that a single value of $\tau = 0.05$ is effective on every task and empirically ablate the choice of τ in Appendix A.

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 4) 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. On all of our tasks, we choose T to be a large value equal to 450 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 6.2, Figure 3. Of course, there are many other possible ways of selecting T, including ideas inspired from offline model-selection methods in offline reinforcement learning (Thomas et al., 2015), but our simple procedure, which is also popular in offline RL (Fu et al., 2020), is sufficient to obtain good optimization performance.

Choice of $\mu(\mathbf{x})$. By default, we compute $\mu(\mathbf{x})$ via multiple steps of gradient ascent on the learned $\hat{f}(\mathbf{x})$ during training, which is different from the procedure for computing $\mathbf{x}^*$ in Equation 4, we find that a version of COMs that also utilizes the trust-region procedure in Equation 4 for computing $\mu(\mathbf{x})$ in Equation 3 also performs similarly to this default version of COMs. We present empirical evidence for this observation in Appendix A.

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 4 (Line 5). Finally, at the end of training, we run the trust-region 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. Elaborate implementation details for our method can be found in Appendix A. 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 C. In order to train this conservative objective model, we use the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 10^{-3}. Empirically, we found larger values of η to produce inputs $\mathbf{x}_T$ that do not maximize the values of $\hat{f}^\star_\theta(\mathbf{x}_T)$, and 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 one gradient ascent step starting from a given design in the dataset, $\mathbf{x}_0 \in \mathcal{D}$. While it appears limiting at first, we show in Appendix A that this is sufficient to ensure the conservatism constraint in Equation 5. During optimization, we utilized the trust-region gradient-ascent optimizer with $\beta = 0.9$. As we will also show in our experiments (Section 6.2), this value of β universally produces stable optimization behavior. Finally, in order to choose the time step T in Equation 4 that is supposed to provide us with the final solution $\mathbf{x}^\star = \mathbf{x}_T$, we pick a large and universal time step of $T = 450$.

4. Theoretical Analysis of COMs

We will now theoretically analyze conservative objective models, and show that the conservative training procedure (Equation 3) indeed learns a conservative model of the objective function. To do so, we will show that under Equation 3, 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 shall denote $\overline{\mathcal{D}}(\mathbf{x})$ as the smoothed density of $\mathbf{x}$ in the dataset $\mathcal{D}$ (see Appendix B for a formal definition). We will express Equation 3 in an equivalent form that factorizes the distribution $\mu(\mathbf{x})$ as $\mu(\mathbf{x}) = \sum_{\mathbf{x}_0 \sim \mathcal{D}} \overline{\mathcal{D}}(\mathbf{x}_0)\mu(\mathbf{x}_T|\mathbf{x}_0)$:

$$\min_{\theta \in \Theta} \alpha \left(\mathbb{E}_{\mathbf{x}_0 \sim \overline{\mathcal{D}}, \mathbf{x}_T \sim \mu(\mathbf{x}_T|\mathbf{x}_0)} \left[\hat{f}_\theta(\mathbf{x}_T) \right] - \mathbb{E}_{\mathbf{x} \sim \overline{\mathcal{D}}} \left[\hat{f}_\theta(\mathbf{x}) \right] \right) + \frac{1}{2} \mathbb{E}_{(\mathbf{x},y) \sim \overline{\mathcal{D}}} \left[\left(\hat{f}_\theta(\mathbf{x}) - y \right)^2 \right]. \quad (6)$$

While $\mu(\mathbf{x}_T|\mathbf{x}_0)$ is a Dirac-delta distribution in practice (Section 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) (Jacot et al., 2018) 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 (Kumar et al., 2020) to prove our theoretical result in Theorem 1, shown below:

Proposition 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 6 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 1 including a complete formal statement can be found in Appendix B. The intuition behind the proof is that inducing conservatism in the function $\hat{f}_\theta$ at each gradient step of optimizing Equation 6 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, this bound with an offset to account for the fact that the bound in Proposition 1 is computed in expectation over $\mathbf{x}_T$ while we want to bound the point $\mathbf{x}^*$, will hold for the point $\mathbf{x}^\star$, and therefore the estimated value of this point will not overestimate its true value. This 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, $\mathbf{x}_0 \in \mathcal{D}$.

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 (Rubinstein & Kroese, 2004) and other methods derived from the REINFORCE trick (Williams, 1992b; Rubinstein, 1996), reward-weighted regression (Peters & Schaal, 2007), and Gaussian processes (Snoek et al., 2015; Shahriari et al., 2016; Snoek et al., 2012) 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 (Snoek et al., 2015; 2012), latent variable models (Kim et al., 2019; Garnelo et al., 2018b;a), and ensembles of learned score models (Angermueller et al., 2020a;b; Mirhoseini et al., 2020). These methods still require actively querying the true function $f(\mathbf{x})$. Further, as shown by (Brookes et al., 2019; Fannjiang & Listgarten, 2020; Kumar & Levine, 2019), 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 (Kingma & Welling, 2013; Goodfellow et al., 2014a) that models the manifold of inputs. (Brookes et al., 2019; Fannjiang & Listgarten, 2020) use a variational autoencoder (Kingma & Welling, 2013) to model the space of $\mathbf{x}$ and use it alongside a learned objective function. (Kumar & Levine, 2019) 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 6), and as a result these generative models often need to be tuned for each domain (Trabucco et al., 2021). 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.

Adversarial examples. As discussed in Section 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 (Goodfellow et al., 2014b), 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 (Tramèr et al., 2018) of the learned function, and there is evidence that robust objective models help mitigate over estimation (Santurkar et al., 2019), 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.

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 offline MBO tasks based on problems in physical sciences, neural network design, and robotics, proposed in the design-bench benchmark (Trabucco et al., 2021). 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 does the trust-region optimizer improve the stability of optimizing COMs?, **(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 by using visualizations for our analysis. Code for reproducing our experimental results is available at `https://github.com/brandontrabucco/design-baselines`.

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 (Brookes et al., 2019), MINs (Kumar & Levine,

	Superconductor-v0	**HopperController-v0**	**AntMorphology-v0**	**DKittyMorphology-v0**
$\mathcal{D}$ **(best)**	73.90	1361.6	108.5	215.9
MINs	80.23 ± 10.67	746.1 ± 636.8	388.5 ± 9.085	352.9 ± 38.65
CbAS	72.17 ± 8.652	547.1 ± 423.9	393.0 ± 3.750	369.1 ± 60.65
Autofocus	77.07 ± 11.11	443.8 ± 142.9	386.9 ± 10.58	376.3 ± 47.47
Grad. Ascent	89.64 ± 9.201	1050.8 ± 284.5	399.9 ± 4.941	**390.7 ± 49.24**
REINFORCE	86.58 ± 4.270	553.0 ± 296.0	**446.3 ± 8.669**	334.5 ± 38.22
CMA-ES	98.59 ± 5.271	676.5 ± 344.5	**921.3 ± 1383.**	-0.9 ± 2.350
BO-qEI	89.73 ± 0.000	759.4 ± 142.3	348.6 ± 4.229	265.5 ± 0.000
COMs (ours)	**114.47 ± 6.400**	**2149.5 ± 400.3**	**442.4 ± 9.651**	333.4 ± 32.16

Table 1. **Comparative evaluation of COMs** against prior methods in terms of the mean 100th-percentile score and its standard deviation over 16 trials. Tasks include Superconductor-v0, HopperController-v0, AntMorphology-v0, and DkittyMorphology-v0, which is the set of continuous tasks provided by (Trabucco et al., 2021). COMs perform strictly better on high-dimensional tasks, obtaining about **2x** gains on HopperController-v0, and compelling gains on Superconductor-v0 and AntMorphology-v0 tasks. In addition, COMs is the only method that 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.

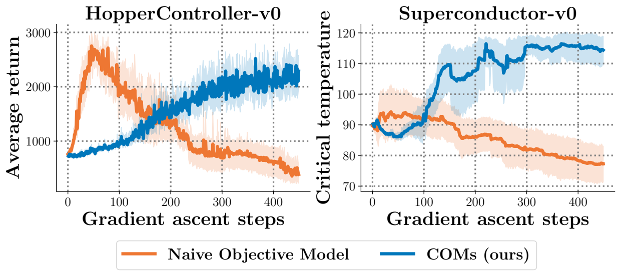

Figure 3. **Stability of COMs versus naïve gradient ascent.** The x-axis shows the number of gradient ascent steps taken on the design $\mathbf{x}^*$, and the y-axis shows the 100th percentile of the ground truth task objective function evaluated at every gradient step, which is used only for analysis 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.

2019), REINFORCE (Williams, 1992a), CMA-ES (Hansen, 2006) and autofocused CbAS (Fannjiang & Listgarten, 2020), that augments CbAS with a re-weighted objective model, and BO-qEI, Bayesian Optimization with the quasi-expected improvement acquisition function (Wilson et al., 2017). CbAS and MINs train generative models such as VAEs (Kingma & Welling, 2013) and GANs (Goodfellow et al., 2014a), 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 in all experiments and for all tasks. We also compare to a naïve gradient ascent baseline that simply learns a non-conservative model of the objective and optimizes against it via gradient ascent.

Our evaluation protocol follows prior work (Brookes et al., 2019; Trabucco et al., 2021): 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 100th percentile ground truth objective values on this set of samples, $\max(\mathbf{x}_1^\star, \cdots, \mathbf{x}_N^\star)$, as well as the 50th percentile objective value, averaged over 16 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.

Our results for different domains are shown in Table 1: **(A)** Superconductor-v0 (Fannjiang & Listgarten, 2020), where the goal is to optimize over 81-dimensional superconductor designs to maximize the critical temperature using 16953 points, **(B)** HopperController-v0 (Kumar & Levine, 2019), 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)** DKittyMorphology-v0, where the goal is to design the 60 and 56-dimensional morphology, respectively, of robots to maximize policy performance using datasets of size 12300 and 9546, respectively. Results for baseline methods are based on numbers reported by Trabucco et al. (2021). Additional details for the setup of these tasks is provided in Appendix Section D. On three out of four tasks, COMs attain the best results, in some cases (e.g. (B) HopperController) attaining the performance of over *twice* 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. While COMs performs within a standard deviation margin of CbAS and MINs on DKittyMorphology-v0, it substantially outperforms prior methods on Superconductor-v0 and the high-dimensional HopperController-v0 tasks, by a factor of **1.3-2x**. A naïve objective model without the conservative term, which is prone to falling off-the-manifold of valid inputs, struggles in especially high-dimensional tasks. 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

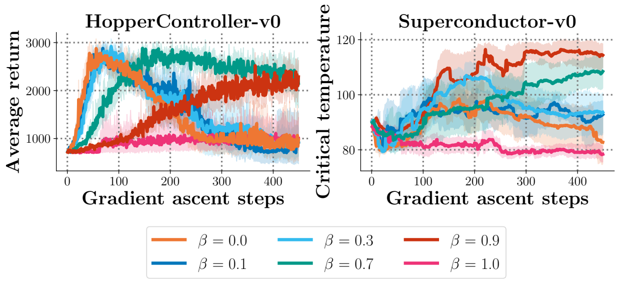

Figure 4. **Ablation of stability and universality of β.** In each of the two plots, we instantiate COMs on the HopperController-v0 and Superconductor-v0 tasks, and vary β that controls the strength of the trust-region (Equation 4). The x-axis denotes the number of gradient ascent steps taken on the 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 β results in improved stability of COMs, and *universally*, $\beta = 0.9$ results in consistent high performance.

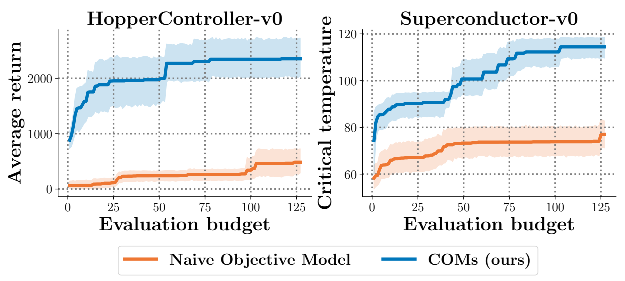

Figure 5. **Ablation of consistency of COMs by visualizing sensitivity to the post-optimization evaluation budget.** How does the performance of COMs and naïve gradient ascent vary as the evaluation budget is reduced? In our standard evaluation, we allow each offline MBO algorithm a "budget" of 128 evaluations 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-v0, COMs is nearly invariant to budgets down to size 55. This indicates that COMs consistently produce optimized inputs $\mathbf{x}^\star$ that attain high values under the true function.

a parametric model $y = \hat{f}_\theta(x)$ of the objective function with a regularizer, without any need for training a generative model, which may be harder in practice to tune.

6.2. Ablation Studies

In this section, we perform an ablative experimental analysis of COMs to answer questions posed at the beginning of Section 6. First, we evaluate the efficacy of 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 the value of β (Equation 4) that controls the strength of the trust-region constraint and show that a single β is uniformly optimal on a wide-variety of MBO tasks over input spaces with very distinct properties. Finally, 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. 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.

Trust-region gradient-ascent (Equation 4) improves stability and COMs are robust to β. Next we evaluate the stability of COMs to the various values of the hyperparameter β that appears in the trust-region gradient-ascent optimization (Equation 4). Note that this β was chosen universally across tasks. In Figure 4, we evaluate the sensitivity of the performance of COMs with respect to β on two tasks: HopperController-v0, and Superconductor-v0. Note that in both cases, COMs becomes increasingly stable and performance improves as β approaches an upper limit of 1, and appears optimal in both tasks for $\beta = 0.9$. While COMs do not completely eliminate this hyperparameter, our empirical results in Table 1 across a wide-array of input types (*both* the weights of a neural network in HopperController and the morphology of a robot in AntMorphology; 60 dimensions vs 5126 dimensional input spaces; etc.) with $\beta = 0.9$ suggest this value is universal and can be fixed to a constant.

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 (Trabucco et al., 2021; Brookes et al., 2019), 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 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 5, COMs not only perform well at larger values of N, but are also effective with smaller budgets, reaching near-optimal performance on HopperController-v0 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.

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 the most high-dimensional cases, COMs improves on the next best method by a factor of **1.3-2x**. 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 once good solutions are found.

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 reviewers and all other members from RAIL at UC Berkeley for their suggestions, feedback and support. This work was supported by National Science Foundation and the DARPA Assured Autonomy Program, and compute support from Google and Intel.

References

Angermueller, C., Belanger, D., Gane, A., Mariet, Z., Dohan, D., Murphy, K., Colwell, L., and Sculley, D. Population-based black-box optimization for biological sequence design. *arXiv preprint arXiv:2006.03227*, 2020a.

Angermueller, C., Dohan, D., Belanger, D., Deshpande, R., Murphy, K., and Colwell, L. Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=HklxbgBKvr.

Berkenkamp, F., Schoellig, A. P., and Krause, A. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 491–496. IEEE, 2016.

Brookes, D., Park, H., and Listgarten, J. 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.

Fannjiang, C. and Listgarten, J. Autofocused oracles for model-based design. *arXiv preprint arXiv:2006.08052*, 2020.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. M. A. Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018a.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S. M. A., and Teh, Y. W. Neural processes. *CoRR*, abs/1807.01622, 2018b. URL http://arxiv.org/abs/1807.01622.

Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., Light, Y., McGlinchey, S., Michalovich, D., Al-Lazikani, B., et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.

Gómez-Bombarelli, R., Duvenaud, D., Hernández-Lobato, J. M., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. In *ACS central science*, 2018.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. NIPS'14, 2014a.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b.

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

Hamidieh, K. 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.

Hansen, N. The CMA evolution strategy: A comparing review. In Lozano, J. A., Larrañaga, P., Inza, I., and Bengoetxea, E. (eds.), *Towards a New Evolutionary Computation - Advances in the Estimation of Distribution Algorithms*, volume 192 of *Studies in Fuzziness and Soft Computing*, pp. 75–102. Springer, 2006. doi: 10.1007/3-540-32494-1_4. URL https://doi.org/10.1007/3-540-32494-1_4.

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

Hoburg, W. and Abbeel, P. Geometric programming for aircraft design optimization. volume 52, 04 2012. ISBN 978-1-60086-937-2. doi: 10.2514/6.2012-1680.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. Attentive neural processes. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkE6PjC9KX.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *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.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2013. URL http://arxiv.org/abs/1312.6114. cite arxiv:1312.6114.

Kumar, A. and Levine, S. Model inversion networks for model-based optimization. *arXiv preprint arXiv:1912.13464*, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Liao, T., Wang, G., Yang, B., Lee, R., Pister, K., Levine, S., and Calandra, R. 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.

Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Bae, S., et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

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

Rubinstein, R. Y. Optimization of computer simulation models with rare events. *European Journal of Operations Research*, 99:89–112, 1996.

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

Santurkar, S., Ilyas, A., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Image synthesis with a single (robust) classifier. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 1260–1271, 2019.

Sarkisyan, K. S., Bolotin, D. A., Meer, M. V., Usmanova, D. R., Mishin, A. S., Sharonov, G. V., Ivankov, D. N., Bozhanova, N. G., Baranov, M. S., Soylemez, O., Bogatyreva, N. S., Vlasov, P. K., Egorov, E. S., Logacheva, M. D., Kondrashov, A. S., Chudakov, D. M., Putintseva, E. V., Mamedov, I. Z., Tawfik, D. S., Lukyanov, K. A., and Kondrashov, F. A. 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.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL `http://arxiv.org/abs/1707.06347`.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104: 148–175, 2016.

Snoek, J., Larochelle, H., and Adams, R. P. 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`.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., and Adams, R. Scalable bayesian optimization using deep neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, 2015.

Thomas, P., Theocharous, G., and Ghavamzadeh, M. High-confidence off-policy evaluation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Trabucco, B., Geng, X., Kumar, A., and Levine, S. Design-bench: Benchmarks for data-driven offline model-based optimization, 2021. URL `https://github.com/brandontrabucco/design-bench`.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I. J., Boneh, D., and McDaniel, P. D. 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`.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992a. doi: 10.1007/BF00992696. URL `https://doi.org/10.1007/BF00992696`.

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

Wilson, J. T., Moriconi, R., Hutter, F., and Deisenroth, M. P. The reparameterization trick for acquisition functions. *CoRR*, abs/1712.00424, 2017. URL `http://arxiv.org/abs/1712.00424`.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. 2017. URL `https://arxiv.org/abs/1611.01578`.