Data-Driven Model-Based Optimization via Invariant Representation Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study the problem of data-driven model-based optimization, where the goal is to find the optimal design, provided access to only a static dataset, with no active data collection. The central challenge in data-driven model-based optimization is distributional shift, where the optimizer is fooled into producing out-of-distribution (OOD) designs that erroneously appear promising under a model trained on the provided data. To address this issue, we formulate model-based optimization as *domain adaptation*, where the goal is to make accurate predictions for the value of designs during optimization ("target domain"), when training only on the dataset ("source domain"). This perspective leads to IOM, our approach for addressing distributional shift by enforcing invariance between the learned representations of the training dataset and optimized designs. In IOM, if the optimized designs are too different from the training dataset, the representation will be forced to lose much of the information that distinguishes good designs from bad ones, making all choices seem mediocre. Critically, when the optimizer is aware of this representational tradeoff, it should choose not to stray too far from the training distribution, leading to a natural trade-off between distributional shift and learning performance.

1 Introduction

Many real-world applications of machine learning involve learning how to make better designs from data. This is formulated as data-driven model-based optimization (also known as black-box optimization [2]), in which an algorithm is provided with data from previous experiments consisting of $(\mathbf{x}, y)$ tuples, where $\mathbf{x}$ represents a design (e.g., a molecule [12] or robot controller [5]) and y represents its utility (e.g., certain property of the molecule or the effectiveness of a controller). The goal is to generate the best possible design, typically one that is better than the best one in the dataset. All of these problem domains are united by a common challenge: in order to improve, the designs learned by the algorithm must differ from the distribution of the designs in the training data, inducing *distributional shift* [22], which must be carefully handled.

Typical methods for solving such data-driven model-based optimization problems try to learn some form of a proxy model $\hat{f}(\cdot)$ that maps a design $\mathbf{x}$ to its objective value y via supervised learning on the static dataset, and then optimize the design $\mathbf{x}$ against this learned model. However, overestimation errors in the learned model will erroneously drive the optimizer towards low-value designs that "fool" the proxy into producing a high values [22]. To handle this, prior works have proposed to regularize the learned model to be conservative by penalizing its values on unseen designs [38, 42, 18], or explicitly constraining the optimizer (e.g., by using density estimation) [8, 22].

In this paper, we take a different perspective for the distributional shift challenges in the model-based optimization, and formulate it as domain adaptation, where the "target" domain consists of the distribution of designs that are chosen by the optimizer, and the source domain corresponds to the training set. Framing the problem in this way, we derive a class of methods that regulate distributional

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

shift at the *representation level*, and admit effective hyperparameter tuning strategies. The intuition behind our approach is that, if the internal representations of the learned model are made *invariant* to the differences between the designs seen in the training data vs. the optimized distribution, then excessive distributional shift is naturally discouraged because excessively invariance would lead the out-of-distribution points to attain mediocre values on average. That is, as the optimized decisions deviate more from the training data, the requirement to maintain invariance will cause the model's representation to lose information, and will hence force the predicted value to become closer to the average y value in the dataset. This will force the optimizer to stay within the distribution of the dataset, since out-of-distribution points will only be mediocre under the learned model. Of course, the representation cannot be perfectly invariant in general, because the optimized points are different from the training data, and hence the method must strike a balance. In practice, one can instantiate this idea using an adversarial training procedure.

Our main contribution is an offline, data-driven model-based optimization method based on a novel connection with domain adaptation. Our approach, IOM, optimizes the design against a learned model that admits invariant representations between optimized designs and the dataset. This enables controlling distributional shift and also provides a way to derive effective offline hyperparameter tuning strategies that we show actually work well in practice. IOM can be implemented simply by combining any supervised regression procedure with a regularizer that enforces invariance. The regularizer is implemented using any discrepancy measure between two distributions, and we instantiate IOM using the χ^2-discrepancy measure via a least-squares generative adversarial network. This procedure trains a discriminator to discriminate between the representations on the training dataset and optimized decisions, whereas the learned representations are trained to be as invariant as possible to fool the discriminator. We theoretically derive IOM from our domain adaptation formulation of model-based optimization and we also use this formulation to derive tuning strategies for several common hyperparameters in IOM. Empirically, we evaluate IOM on several tasks from Design-Bench [37] and find that it outperforms the best prior methods, and additionally, admits appealing offline hyperparameter tuning strategies unlike the prior methods.

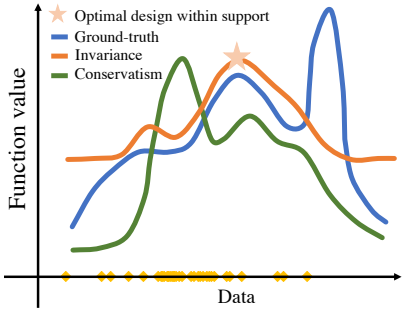

Figure 1: **Illustration showing the intuition behind IOM** on a 1D task. While conservatism regularizes the learned model to take low values on designs not observed in the dataset, training with invariance induces the out-of-distribution designs to attain objective values close to the average objective value in the training dataset. This discourages the optimizer from going out-of-distribution, as the best point within the train distribution is better.

2 Related Work

Model-based optimization (MBO) or black-box optimization [8, 3, 35, 34, 30, 42] has been studied widely in prior works. Among these works, the central challenge is in handling the distributional shift that arises between the distribution of designs seen in the dataset and those taken by the optimizer [8, 9, 22]. For methods that learn a model, this typically manifests itself as *model exploitation*, where the optimizer can find out-of-distribution designs for which the model erroneously predicts good values, analogously to adversarial examples.

To address distribution shift, several recent works have proposed to be conservative or pessimistic. This is majorly the case in other decision-making problems offline reinforcement learning (RL) [27] and contextual bandits [31], where instead of optimizing for a design, we must find the optimal action at each state. In offline MBO, prior works have tried to be conservative in two generally different ways: either by constraining the optimized designs to lie close to the training designs [8, 9, 22, 32, 23] using generative models [21, 13], or by learning a "robust" version of the learned model that is trained to be conservative [38, 25, 24] or calibrated [10, 42]. While the latter class of methods has worked well in practice [37, 25], these methods can sometimes be overly conservative and are quite sensitive to hyperparameters. Sensitivity to hyperparameters is acceptable as long as one can attain good performance using reasonable offline tuning strategies, without accessing the groundtruth function, but we find in our experiments (Section 6.2) this is not possible for some of these prior conservative methods. On the contrary, IOM not only does not require a generative model, attains better performance, and also provides for a convenient hyperparameter selection rule.

In this work we formulate data-driven model-based optimization as domain adaptation [4, 11], and derive algorithms for model-based optimization from this perspective. This perspective has been used to devise decision-making methods for treatment effect estimation [19, 20, 41] and off-policy evaluation in RL [28, 26]. While these prior works do use an invariance regularizer, they primarily focus on obtaining more accurate estimations, whereas our goal is to utilize invariance to combat distributional shift during optimization. Thus, our usage of invariant representations is tailored towards the optimizer, instead of aiming for accurate predictions everywhere like these prior works, which is impossible in general. Furthermore, our analysis uses invariant representations to derive procedures for tuning and early stopping, while these prior works do not explore these perspectives.

3 Preliminaries

The goal in offline model-based optimization (MBO) [37] is to produce designs that maximize some objective function $f(\mathbf{x})$, by only utilizing a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of designs $\mathbf{x}_i$ and their corresponding objective values y_i. No active queries to the groundtruth function are allowed. Typically offline MBO is formulated as finding the optimal design $\mathbf{x}^\star$. However, we will use a slightly more general formulation, where the goal is to optimize a *distribution* over designs $\mu(\mathbf{x})$. We will use this formulation largely for convenience of analysis, where the classic case is recovered when $\mu(\mathbf{x})$ is a Dirac delta function, though we also note that optimizing distributions over designs is often studied in settings such as synthetic biology, where the goal is to explicitly diversify the designs [6]. Thus, our goal is to find a distribution over designs $\mu_{\text{OPT}}(\mathbf{x})$ using only the static dataset $\mathcal{D}$, such that it maximizes the *expected value* of $f(\mathbf{x})$. This can be formalized as follows (Alg is an algorithm):

$$\mu_{\text{OPT}} := \max_{\mathsf{Alg}} \ \mathbb{E}_{\mathbf{x}\sim\mu(\mathbf{x})}[f(\mathbf{x})] := J(\mu) \quad \text{s.t.} \ \ \mu = \mathsf{Alg}(\mathcal{D}). \tag{1}$$

The methods we consider in this paper fit a learned model $f_\theta(\mathbf{x})$ to $(\mathbf{x}_i, y_i)$ pairs in the dataset via a standard supervised regression objective, with an additional regularizer. We will assume that f_θ is composed of two components: a representation $\phi \in \mathbb{R}^d$ that maps a given $\mathbf{x}$ into a d-dimensional vector, and a forward model, f_θ that converts the representational output into a scalar objective value, i.e., $f_\phi(\mathbf{x}) := f_\theta(\phi(\mathbf{x}))$. We will use the notation f_θ and f_ϕ interchangeably. Given a learned model $f_\theta(\phi(\mathbf{x}))$, one approach to obtain the optimized distribution μ_{OPT} is by optimizing the expected value $J_\theta(\mu) := \mathbb{E}_{\mathbf{x}\sim\mu}[f_\theta(\phi(\mathbf{x}))]$ under the learned model. This distribution $\mu_{\text{OPT}}(\mathbf{x})$ can be instantiated in many ways: while some prior work [22, 8] represents μ_{OPT} via a parametric density model over $\mathcal{X}$ and optimizes this density model to obtain μ_{OPT}, other prior work [38] uses a non-parametric representation for μ_{OPT} using a set of optimized "design particles". In this paper, we represent μ_{OPT} using a set of design particles, each of which is obtained by optimizing the learned objective function $f_\theta(\phi(\mathbf{x}))$ with respect to the input $\mathbf{x}$ via gradient descent, starting from i.i.d. samples from the dataset. Formally:

$$\mathbf{x}_{k+1}^i \leftarrow \mathbf{x}_k^i + \eta \nabla_{\mathbf{x}} f_\theta(\phi(\mathbf{x}))|_{\mathbf{x}=\mathbf{x}_k^i}, \ \text{ for } \ k \in [1, T], \quad \mathbf{x}_0^i \sim \mathcal{D} \tag{2}$$

The optimized distribution μ_{OPT} is then given by $\mu_{\text{OPT}} = \delta\{\mathbf{x}_T^1, \mathbf{x}_T^2, \cdots\}$.

The challenge of distributional shift in offline MBO. When the learned model $f_\theta(\phi(\mathbf{x}))$ is trained via supervised regression on the offline dataset $\mathcal{D}$: $\arg\min_\theta \frac{1}{n}\sum_{i=1}^n (f_\theta(\mathbf{x}_i) - y_i)^2$, the ERM principle states that $f_\theta(\phi(\mathbf{x}))$ will reasonably approximate $f(\mathbf{x})$ in expectation only under the distribution of the training dataset $\mathcal{D}$, i.e., μ_{data}, however this may not hold out of the training distribution. When optimizing the learned function, overestimation errors in $f_\theta(\phi(\mathbf{x}))$ will inevitably lead the optimizer to converge to points ($\mu_{\text{OPT}} \neq \mu_{\text{data}}$) that have a high value under the learned function (i.e., high $J_\theta(\mu_{\text{OPT}})$) but not a high ground truth value $J(\mu_{\text{OPT}})$. On the other hand, we show that when using our method IOM, this simple procedure is sufficient in finding good designs.

4 Data-Driven Offline Model-based Optimization as Domain Adaptation

In this section, we will formulate data-driven offline MBO as domain adaptation, which will allow us to derive our method, invariant objective models (IOM). Instead of attempting to simply constrain the optimizer to directly avoid distributional shift, our approach modifies the representations learned by the model $\hat{f}_\theta$ by adding an invariance regularizer during training such that there is minimal distribution shift under the learned invariant representation during optimization.

If we can ensure that the learned model f_θ is accurate on the distribution found by the optimizer, μ_{OPT}, then we can ensure that we will obtain good designs. Therefore, to handle the distributional shift, we can treat μ_{data} as the "source domain" and the optimized distribution μ_{OPT} as the "target domain."

Just as the goal in domain adaptation is to train on the source domain and make accurate predictions on the target domain, we will aim to train $f_\theta(\mathbf{x})$ on μ and with the goal of making more accurate predictions under μ_{OPT}. Of course, learning to make accurate predictions on *any* unseen distribution is impossible for any domain adaptation method [44], but in the case of MBO, the choice of the target distribution μ_{OPT} is also made by the algorithm itself, such that invariance can be enforced *both* by changing the representation and changing the target distribution μ_{OPT}.

4.1 Invariant Objective Models (IOM)

IOM controls distributional shift between the optimized distribution μ_{OPT} and the training distribution μ_{data} at the representational level, by regularizing the distribution of the features $\phi(\mathbf{x})$ under μ_{OPT} to match that under μ_{data}. We will first present the intuition behind the method, and then formalize it by setting up a bi-level optimization problem that we will also analyze theoretically.

Intuition: As the optimizer deviates from the training distribution μ, the invariance regularizer forces the distribution of optimized designs to resemble the distribution over representations $\phi(\mathbf{x})$ under μ_{data}, i.e., $\mathbb{P}_{\mu_{\text{OPT}}}(\phi(\mathbf{x})) \approx \mathbb{P}_{\mu_{\text{data}}}(\phi(\mathbf{x}))$. Such an invariance can be enforced when μ_{OPT} is sufficiently far away from the training distribution. This form of representational invariance in turn causes the expected value of the learned function f_θ under μ_{OPT} to appear (roughly) equal to the average objective value in the training dataset, i.e., $\mathbb{E}_{\mathbf{x}\sim\mu_{\text{OPT}}}[f_\theta(\phi(\mathbf{x}))] \approx \mathbb{E}_{\mathbf{x}\sim\mu_{\text{data}}}[f_\theta(\phi(\mathbf{x}))]$, if μ_{OPT} is too far away from the training distribution. As a result, the expected value of the learned model $f_\theta(\phi(\mathbf{x}))$ under μ_{OPT}, $J_\theta(\mu_{\text{OPT}})$, will be worse than the value of the best design within the training distribution (since the best design input sampled from μ_{data} will attain a higher objective value than the average value under μ_{data}). This would pull the optimizer closer to the training distribution, thereby acting as a guard against diverging too far away from the dataset. Iteratively regularizing ϕ to enforce invariance between μ_{OPT} and μ until convergence will restrict the optimizer from selecting erroneously overestimated out-of-distribution inputs. We will now present a formalization of this intuition in the form of a bi-level optimization problem that aims to maximize a lower-bound on the expected objective value attained under μ_{OPT}, and derive the training objectives for IOM using this bi-level optimization formulation.

Bi-level optimization for invariant objective models (IOM). To enforce invariance at a representational level, we utilize an invariance regularizer. Denoted as $\text{disc}_{\mathcal{H}}(p, q)$, our invariance regularizer is a discrepancy measure between two probability distributions p and q, w.r.t. a certain hypothesis class $\mathcal{H}$. For example, if $\mathcal{H}$ is the class of all 1-Lipschitz functions, $\text{disc}_{\mathcal{H}}$ is the 1-Wasserstein distance; if $\mathcal{H}$ is the class of linear functions, $\text{disc}_{\mathcal{H}}$ is the maximum mean discrepancy (MMD) [14] distance. In practice, we implement this via a generative adversarial network as we discuss in the next subsection. In addition to training f_θ via standard supervised regression on the labels in the training data, IOM trains the representation $\phi(\mathbf{x})$ and the model $\widehat{f}(\cdot)$ to additionally minimize the discrepancy $\text{disc}_{\mathcal{H}}(\mathbb{P}_\mu(\phi(\mathbf{x})), \mathbb{P}_{\mu_{\text{OPT}}}(\phi(\mathbf{x})))$ between distributions of representations under the optimized distribution, μ_{OPT} and the training distribution μ. This objective can be formalized as:

Algorithm 1 IOM: Training and Optimization

1: **Input**: training data $\mathcal{D}$, number of gradient steps $T = 50$ to optimize μ_{OPT} starting from the training distribution μ, training iteration K, batch size n,
2: **Initialize**: representation model $\phi_\eta(\cdot)$, learned model $\widehat{f}_\theta(\cdot)$, set of optimized design particles $\{\mathbf{x}_i^*\}_{i=1}^m$
3: **for** $k = 1 \cdots, K$ **do**,
4: Sample n training points $(\mathbf{x}_i, y_i) \sim \mathcal{D}$
5: Run one gradient step (Equation 4) w.r.t θ_k and ϕ_k to obtain ϕ_{k+1} and $f_{\theta_{k+1}}(\phi_{k+1}(\cdot))$.
6: Run one gradient step to optimize design particles w.r.t. $f_\theta(\phi(\mathbf{x}))$: $\mathbf{x}_i^* \leftarrow \mathbf{x}_i^* + \eta \nabla_{\mathbf{x}} f_{\theta_k}(\phi_k(\mathbf{x}_i^*))$.
7: **end for**

$$\max_{\mu_{\text{OPT}}} \quad \mathbb{E}_{\widehat{\mathbf{x}}\sim\mu_{\text{OPT}}}\left[f_\theta^*(\phi^*(\widehat{\mathbf{x}}))\right]$$
$$\text{s.t.} \ \ (\phi^*, f_\theta^*) = \arg\min_{\phi, \widehat{f}} \ \ \frac{1}{n}\sum_{i=1}^{n}(\widehat{f}(\phi(\mathbf{x}_i)) - y_i)^2 + \lambda \cdot \text{disc}_{\mathcal{H}}(\mathbb{P}_{\mu_{\text{data}}}(\phi(\mathbf{x})), \mathbb{P}_{\mu_{\text{OPT}}}(\phi(\mathbf{x}))), \quad (3)$$

where λ denotes a weighting hyperparameter. In Section 4.2, we will describe how we can convert this bi-level optimization problem into a practical MBO algorithm that can be implemented with high capacity deep neural networks. This is followed by a theoretical analysis to formally justify the objective in Section 4.3, and a discussion of how the hyperparameter λ can be tuned against a validation set, in Section 5, without any online access to the ground truth function.

4.2 Optimizing the Bi-Level Problem and the Practical IOM Algorithm

In this section, we will describe how we can convert the abstract bi-level optimization problem above into an objective that can be trained practically and discuss some practical implementation details. IOM alternates between solving the bi-level optimization problem with respect to the distribution μ_{OPT} and the learned model f_θ and ϕ independently.

For training the learned model and the representation f_θ and ϕ, IOM solves the inner optimization by running one-step of gradient descent (GD), utilizing the current snapshot of μ_{OPT} (denoted μ^t_{OPT}):

$$(\phi_{t+1}, \theta_{t+1}) = \mathsf{GD}_{\theta,\phi;t}\left(\frac{1}{n}\sum_{i=1}^{n}(f_\theta(\phi(\mathbf{x}_i)) - y_i)^2 + \lambda \cdot \text{disc}_{\mathcal{H}}(\mathbb{P}_{\mu_{\text{data}}}(\phi(\mathbf{x})), \mathbb{P}_{\mu^t_{\text{OPT}}}(\phi(\mathbf{x})))\right). \quad (4)$$

Simultaneously, we update the design particles representing $\mu^t_{\text{OPT}} := \delta\{\mathbf{x}^1_t, \mathbf{x}^2_t, \cdots\}$ towards optimizing the learned function $f_{\theta_{t+1}}(\phi_{t+1}(\mathbf{x}))$: $\mathbf{x}^i_{t+1} \leftarrow \mathbf{x}^i_t + \eta \nabla_{\mathbf{x}} f_{\theta_{t+1}}(\phi_{t+1}(\mathbf{x}))|_{\mathbf{x}=\mathbf{x}^i_t}$, $\forall i$. Pseudocode for IOM is shown in Algorithm 1.

Practical implementation details: We model the representation $\phi(\mathbf{x})$ and the learned function $f_\theta(\cdot)$ each as two-hidden layer ReLU networks with sizes 2048 and 1024, respectively. We utilize the χ^2-discrepancy, $\text{disc}_{\mathcal{H}}$ in our experiments: $\text{disc}_\phi(p, q) := \chi^2([\phi(\mathbf{x})]_{\mathbf{x}\sim p}, [\phi(\mathbf{x})]_{\mathbf{x}\sim q})$. This divergence can be instantiated in its variational form via a least-squares generative adversarial network (LS-GAN) [29] on the representations $\phi(\mathbf{x})$. Concretely, we train an LS-GAN discriminator to discriminate between the representations $\phi(\mathbf{x})$ under the training distribution $\mu(\mathbf{x})$ and the optimized distribution $\mu_{\text{OPT}}(\mathbf{x})$, whereas the feature representations under these distributions are optimized to be as invariant to each other as possible. We also evaluated a variety of discrepancy measures, including the maximum mean discrepancy (MMD) [14], but found the χ^2-divergence to be the best performing version across the board. More implementation details can be found in Appendix A.2.

4.3 Theoretical Analysis of Invariant Representation Learning for MBO

In this section, we will formally show that under certain standard assumptions, solving the bi-level optimization problem (Equation 3) enables us to find a better design than the dataset distribution μ_{data}, provably. To show this, we will combine theoretical tools for analyzing offline RL algorithms and domain adaptation [44]. Our goal would be to show that the expected value of the ground truth objective under any given distribution $\pi(\mathbf{x})$ of design inputs can be lower bounded in terms of the expected value under the learned function modulo some error terms. Then we will show that these error terms are controlled tightly when solving the bi-level problem, and finally appeal to uniform concentration to show that the performance of μ_{OPT} is indeed better with high probability.

Notation and assumptions. Following standard assumptions [26], we will analyze the setting when the learned representation lies in some space of functions, $\phi \in \Phi$, and the learned function f_θ lies in $f_\theta \in \mathcal{F}$. We will consider that the optimized distribution μ_{OPT} belongs to a function class, $\mu_{\text{OPT}} \in \Pi$. Our analysis will require that the ground truth model f is approximately realizable under some representation $\phi \in \Phi$ and for some $g \in \mathcal{F}$. That is, we assume:

Assumption 4.1 ($\varepsilon-$realizability)**.** For any distribution π over designs, there exists a representation $\phi \in \Phi$ and a $g \in \mathcal{F}$ such that $\min_{\phi,g} \max_{\pi\in\Pi} \mathbb{E}_{\mathbf{x}\sim\pi}\left[|f(\mathbf{x}) - g(\phi(\mathbf{x}))|\right] \leq \varepsilon_{\mathcal{F},\Phi}$.

In addition, we assume that $\forall f \in \mathcal{F}, ||f||_\infty < \infty$ and is Lipschitz continuous with constant C: $||f||_{\text{L}} \leq C$. This is true for most parameteric function classes, such as neural networks. Then, we can lower bound the average objective value under any distribution in terms of the invariance:

Proposition 4.2 ((Informal) Lower-bounding the ground truth value under π)**.** *Under Assumption 4.1 and the Lipschitz continuity on the function f_θ, the ground truth objective for any given $\pi \in \Pi$ can be lower bounded in terms of the learned objective model $f_\theta \in \mathcal{F}$ and representation $\phi \in \Phi$, with high probability $\geq 1 - \delta$ that:*

$$J(\pi) - J_\theta(\pi) \geq \underbrace{J(\mu_{data}) - J_\theta(\mu_{data})}_{(\blacksquare)} - \underbrace{C_{\mathcal{F}} \cdot disc_{\mathcal{H}}(\mathbb{P}_{\mu_{data}}(\phi(\mathbf{x})), \mathbb{P}_\pi(\phi(\mathbf{x})))}_{(*)} - \varepsilon_{\mathcal{F},\Phi} - \varepsilon_{stat},$$

where $C_{\mathcal{F}}$ is a uniform constant depending on the function class $\mathcal{F}$ and ε_{stat} refers to statistical error that decays inversely with the dataset size, $|\mathcal{D}|$.

A proof for Proposition 4.2 is in Appendix A.1. The $\blacksquare$ term can further be lower bounded using a standard concentration error bound for ERM, and can be controlled if we train $f_\theta(\phi(\cdot))$ minimize

prediction error against ground truth values of f. Now we utilize Proposition 4.2 and a uniform concentration argument to lower-bound the value under the optimized distribution μ_{OPT}.

Proposition 4.3 ((Informal) Performance guarantee for IOM)**.** *Under Assumption 4.1, the expected value of the ground truth objective under μ_{OPT}, $J(\mu_{\text{OPT}})$ is lower bounded by:*

$$J(\mu_{\text{OPT}}) \gtrsim J(\mu_{data}) - \mathcal{O}\left(\sqrt{\frac{\log \frac{|\mathcal{F}||\Phi||\Pi|}{\delta}}{|\mathcal{D}|}} + \frac{\log \frac{|\mathcal{F}||\Phi||\Pi|}{\delta}}{|\mathcal{D}|}\right) + \underbrace{J_\theta(\mu_{\text{OPT}}) - J_\theta(\mu_{data})}_{(\circ)} - (*). \quad (5)$$

A proof of Proposition 4.3 can be found in Appendix A.1. This proposition implies that as long as the optimizer finds designs that improve the learned objective function $f_\theta \in \mathcal{F}$, such that $(\circ)$ is positive, and the discrepancy term $(\star)$ is minimized, then the distribution over designs found by IOM, μ_{OPT}, will compete favorably against the distributions of designs in the dataset. Of course, these bounds present the worst-case guarantees for IOM, but as we will show in our experiments, IOM does clearly improve over the best designs in the offline dataset. In addition, this theoretical analysis also motivates the design of our scheme for performing hyperparameter tuning and model selection. This scheme will be based directly on the result from Proposition 4.2. We will discuss this connection when discussing tuning in Section 5.

Finally, we would like to highlight how invariance leads to "average" values if μ_{OPT} deviates too far away from the data distribution μ. If μ_{OPT} deviates too far, such that $\text{disc}_{\mathcal{H}}(\mathbb{P}_{\mu_{\text{data}}}(\phi), \mathbb{P}_{\mu_{\text{OPT}}}(\phi)) \leq \varepsilon'$, the learned function is bounded too close to the average learned value on the data i.e., $|\widehat{J}(\mu_{\text{OPT}}) - \widehat{J}(\mu_{\text{data}})| \leq C' \cdot \varepsilon'$. Thus, the learned function simply reverts to "predicting the mean objective value" of the training distribution as ε' decreases, thereby losing any information about the identity of the optimized distribution, when the optimizer goes too far out of μ.

5 Offline Workflow and Tuning for IOM

A central challenge in offline model-based optimization is that of offline workflow and hyperparameter tuning: not only must an algorithm be able to produce designs using a static dataset, but it must also prescribe a *fully offline* workflow for tuning hyperparameters so that the produced designs are as good as possible. This includes both explicit hyperparameters (such as the coefficient of conservatism or invariance, λ for IOM) and implicit hyperparameters (such as the early stopping point). The primary challenge stems from the fact that unlike supervised learning, the final performance in MBO is determined by the optimized distribution μ_{OPT}, which is different from the training distribution, μ_{data}, and we must reason about this performance fully offline.

In this section, we discuss how IOM admits a particularly convenient and *fully offline* hyperparameter tuning procedure that works well empirically. The goal of our tuning problem is to find a tuple consisting of the learned representation ϕ^λ, the learned model, f_θ^λ, and the optimized distribution μ_{OPT}^λ, out of a given set of tuples obtained for different values of λ. In other words, we wish to find the λ from the set of models $M_\lambda := (f_\theta^\lambda, \phi^\lambda, \mu_{\mu_{\text{OPT}}}^\lambda)$, such that the corresponding μ_{OPT}^λ attains the highest value under the ground truth objective. Crucially note that this must be done fully offline.

Our key idea: The primary question we wish to answer is: how can we identify if a given tuple $(f_\theta^\lambda, \phi^\lambda, \mu_{\text{OPT}}^\lambda)$ leads to good performance? In order for a tuple trained via IOM to perform well, we need two primary conditions: **(1)** it should be nearly invariant, and as a result, robust to distributional shift, and **(2)** the learned f_θ^λ must attain good generalization performance within the training distribution. Intuitively **(1)** guards us against distributional shift, and **(2)** enables us to find as good of a learned model on the dataset, that does not overfit or underfit as a result of the invariance regularizer. This forms the basis of our tuning method.

Offline tuning of IOM: To instantiate the idea practically, we first run IOM with a set of hyperparameters $\Lambda = \{\lambda_1, \lambda_2, \cdots\}$. Then for each run, we record the validation in-distribution error and the value of the invariance regularizer on a validation set. We then perform tuning in two steps: first, we filter all λ values that attain near-perfect invariance under a user-specified margin ε, i.e., $\text{disc}_{\mathcal{H}}(\mathbb{P}_{\mu_{\text{data}}}(\phi^\lambda(\mathbf{x})), \mathbb{P}_{\mu_{\text{OPT}}^\omega}(\phi^\lambda(\mathbf{x}))) \leq \epsilon$, and are hence guaranteed to be robust to distributional shift. Second, we now pick models that attain good performance within the training distribution by selecting λ values that attain the smallest validation prediction error: $(f_\theta(\phi(\mathbf{x})) - y)^2$, in addition to picking the early stopping point based on the smallest validation error. This enables us to pick λ that gives rise to a generalizing model $f_\theta^\lambda, \phi^\lambda$, while being most robust to distributional shift. This still leaves a

user-defined margin ε as a free parameter, but we find that it is comparatively insensitive and can be selected heuristically, as we show in our experiments in Section 6.2. Formally, this procedure corresponds to solving the following optimization:

$$\lambda^* := \arg\min_{\lambda \in \Lambda} \; \frac{1}{n} \sum_{i=1}^{n_{\text{val}}} (f_\theta^\lambda(\phi^\lambda(\mathbf{x}_i)) - y_i)^2 \;\; \text{s.t.} \;\; \text{disc}_{\mathcal{H}}(\mathbb{P}_{\mu_{\text{val}}}(\phi^\lambda(\mathbf{x})), \mathbb{P}_{\mu^\lambda_{\text{OPT}}}(\phi^\lambda(\mathbf{x}))) \leq \epsilon \tag{6}$$

Finally, for a more formal explanation of the soundness of this strategy, we note that applying Equation 6 provides a tight lower bound on the ground truth objective value in Proposition 4.2. Up to statistical error (which decays as $|\mathcal{D}|$ increases), the prediction error on a held-out validation set estimates ($\blacksquare$) in Proposition 4.2, and a smaller discrepancy (i.e., $\text{disc}_{\mathcal{H}} \leq \varepsilon$) controls the ($\star$) term in Prop. 4.2. $\varepsilon_{\text{stat}}$ is statistical error, and $\varepsilon_{\mathcal{F},\Phi}$ only depends on the function classes involved ($\mathcal{F}, \Phi$). This tuning λ per Equation 6 can be justified as aiming to maximize a lower bound on $J(\mu_{\text{OPT}})$.

Practical implementation. When implementing this procedure in practice, we must first select the value of ε. For obtaining perfect invariance, the value of ε must be equal to 0.25 (when the discriminator is trained using a χ^2-divergence least-squares GAN objective), and therefore, ε must be set close to 0.25. However, in practice, instead of selecting ε, we can select a fraction of all possible λ values that pass the filter. In our experiments, we sort the runs based on the absolute difference between the value of invariance regularizer (i.e., the discriminator training objective) and 0.5, and let the top 15% (top 3 out of the 7 λ values we tune on) of the runs be selected for the second step.

In Section 6, we validate this offline tuning strategy, both against other alternative offline strategies for tuning IOM as well as offline strategies for tuning prior methods (COMs [38] and gradient descent [37]). We find that our offline tuning strategy performs comparably to full online tuning without needing any online queries, whereas other strategies for IOM and other methods, lead to significantly poor performance compared to online tuning.

6 Experimental Evaluation

The goal of our empirical evaluation is to answer the following questions: **(1)** Does IOM outperform prior state-of-the-art methods for data-driven model-based optimization? **(2)** Is our offline tuning strategy for IOM better than other offline tuning strategies for IOM or other prior methods? **(3)** How sensitive is IOM to the choice of the hyperparameter λ? In this section, we will answer these questions via a comparative evaluation of IOM on several continuous data-driven design tasks from the standard tasks from the Design-Bench suite [37].

6.1 Empirical Evaluation on Benchmark Tasks

We compare IOM to a variety of prior methods for model-based optimization: CbAS [8], Autofocused CbAS [9] and MINs [22] that utilize generative models for constraining distributional shift; COMs [38] and RoMA [42] train robust models of the objective function by via conservatism and smoothness, respectively; conventional Bayesian optimization methods such as REINFORCE [39], CMA-ES [16] and BO-qEI [40]. We also compare the standard approach of learning a naïve model of the objective function, which is then optimized via gradient ascent. To better understand the benefits of enforcing invariance, we also compare IOM to IOM-C, which additionally applies a conservatism term on top of invariance that pushes up the learned values of designs in the dataset.

Tasks. We evaluate on four tasks with continuous-valued input space from the Design-Bench [37] benchmark for offline model-based optimization (MBO). Here we briefly summarize these tasks: **(1)** Superconductor: this task requires optimizing the properties of superconducting materials to maximize the operating temperature; the dataset consists of 17010 training points, and each input $\mathbf{x} \in \mathbb{R}^{86}$ represents a serialization of the chemical formula. **(2)** D'Kitty and **(3)** Ant robot morphology design: these tasks require selecting the optimal morphology of a quadrupedal robot, in order to maximize how quickly it can walk or crawl in a particular direction; the morphologies are represented by $\mathbf{x} \in \mathbb{R}^{56}$ and $\mathbf{x} \in \mathbb{R}^{60}$, respectively. **(4)** Hopper controller: this task reframes the classic hopper task [7] as a bandit problem, where instead of sequentially controlling the hopper, the goal is to directly output the parameters of a neural network controller that maximizes hopping speed; note that this task, as set up in Design-Bench, is *not* a RL problem, but rather a bandit problem with a 5126-dimensional continuous action space corresponding to neural network weights.

Evaluation protocol. Following prior work [37, 38, 42, 10], for each method, we query the learned model to obtain the top performing N points, i.e., $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$. We report the maximum objective value among these N samples. This protocol reasonably reflects a real-world design process, where a number of computationally produced designs are tested and the best one is used for deployment.

Hyperparameter tuning. For fair comparison against prior methods studied in Trabucco et al. [37], we follow the uniform tuning strategy for reporting results in this section: we run IOM on each task for a given set of λ values, and pick a single value of λ across every task. Note that this does require access to online evaluation, but prior works do use the same protocol. We will evaluate the efficacy of our offline tuning approach in Section 6.2 extensively.

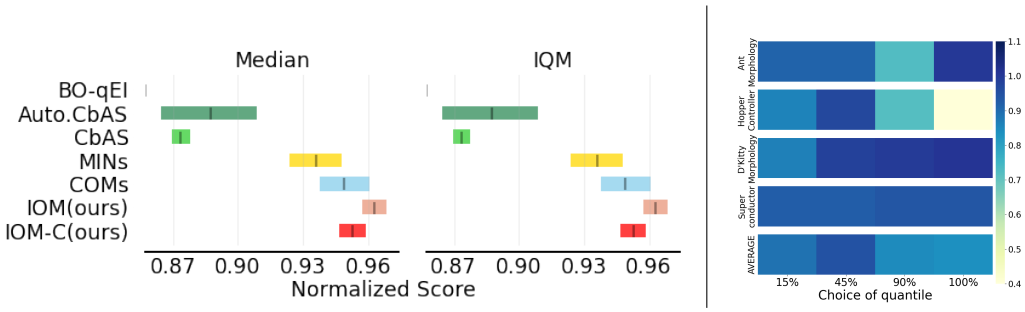

Figure 2: **Left**: Median and IQM [1] (with 95% Stratified Bootstrap CIs) for the aggregated normalized score on Design-bench. IOM improves over the prior methods, including those which use explicit conservatism although IOM does not. As we could not find the individual runs for RoMA [42], we report the mean and standard deviation for RoMA (copied from the results in Yu et al. [42]) in Appendix A.3. **Right**: The offline tuning performance matrix under different quantiles to filter the models based on discriminator loss, compared with the oracle uniform tuning, which shows our tuning strategy is robust to this hyper-parameter.

Results. Following the convention proposed by Trabucco et al. [37], we evaluate all methods in terms of normalized score (higher is better). In Figure 2 (left), we show the median and interquantile mean (IQM) [1] for the aggregated scores across all tasks. Due to space limits, we only report the scores for the top-performing baselines (See Appendix A.2 for the results for all baselines). IOM significantly outperforms all of the prior methods, including state-of-the-art methods that employ conservatism, such as COMs [38]. This indicates that invariance alone (IOM) can effectively handle distribution shifts and avoid over-estimation for out-of-distribution actions. Additionally, a comparison between the performance of IOM and IOM with conservatism (IOM-C) suggests that adding conservatism does not improve performance further. This further corroborates the fact that invariance is effective in preventing the optimizer from going out of distribution and conservatism is not needed.

6.2 Analysis of the Offline Workflow for Tuning IOM

We will now present experiments to understand the effectiveness of our offline tuning scheme (Section 5) for tuning IOM. It is important to highlight that, in general, offline tuning presents significant challenges for MBO. Prior works in bandits and reinforcement learning [43] have suggested that offline tuning should not work at all and, to the best of our knowledge, no prior work in offline MBO has proposed a principled and effective offline tuning method. In this section, we empirically validate our tuning strategy in comparison to other potential strategies, and compare results from our approach to an oracle strategy that uses online evaluation of the true function.

Comparisons. We compare our offline tuning strategy to various offline strategies: (1). only tunes IOM based on validation-set prediction error, analogous to supervised learning (called "prediction error only"); (2). a strategy that tunes IOM based on only the value of the invariance regularizer (called "invariance only"); (3). a method that utilizes our tuning strategy for picking λ, but not for picking the checkpoint ("uniform checkpoint selection") and two offline tuning strategies for tuning other prior methods: (a) tuning COMs [38] using validation-set prediction error (called "prediction error only") and conservatism loss (called "conservatism loss only"); and (b) tuning an objective model trained via gradient ascent [37] using validation MSE error ("gradient ascent"). We report the drop in performance relative to the corresponding oracle online strategy, i.e., the relative drop in performance of the method (IOM, COM, gradient ascent) when tuned offline using the particular strategy compared to oracle online tuning for the same method.

Results. We present our results in Figure 3 (right) comparing the efficacy of various tuning schemes across various methods averaged over all tasks. While offline tuning does lead to a reduction in performance compared to oracle online tuning, we find that the drop is smallest for our offline tuning strategy applied to IOM. All other tuning strategies based on either only prediction error or the invariance regularizer lead to worse performance. This indicates that our tuning strategy is effective in tuning IOM. Furthermore, offline tuning strategies that utilize a validation set with prior methods, such as COMs and gradient ascent, also lead to a significant decrease in performance relative to the oracle. This is quite appealing as it suggests that IOM is more amenable to offline tuning strategies relative to prior offline MBO algorithms. Even if these prior methods can perform well when provided access to online evaluations, they might start to fail in real-world MBO problems when one needs to tune them fully offline.

Ablation analyses. Next we aim to study the sensitivity of our offline tuning strategy with respect to the non-rigorous, user-specified parameter ε in Equation 6 and checkpoint selection based on the validation prediction error. Figure 2 (right) shows the offline tuning performance heatmap for all tasks (along their aggregated performance in the bottom row) under different choice of quantiles (where quantile $x\%$ means the top $x\%$ of the models according to the value of the invariance regularizer, and the given value of ϵ were chosen). The color in each position of the matrix corresponds to the ratio of performance of our offline tuning strategy for a given quantile value, compared to that of uniform oracle tuning. As shown in the bottom row, aggregated over all tasks, we can see our offline selection strategy is quite robust to the hyperparameter ϵ. Figure 3 (left) evaluates the efficacy of our checkpoint selection strategy, we plot the objective value $J(\mu_{\text{OPT}})$ attained as a function of the number of training epochs (i.e., checkpoints), alongside the values for the prediction MSE in the validation set for the corresponding epochs. The checkpoints selected by our early stopping scheme are shown via the vertical line. Observe that indeed the checkpoint for the lowest validation MSE corresponds to a rather good optimized distribution μ_{OPT} for different tasks, no matter the optimal training epochs happen at the beginning or near the end, which validates our early stopping strategy. More detailed results showing the efficacy of early stopping for both IOM can be found in Appendix A.3.

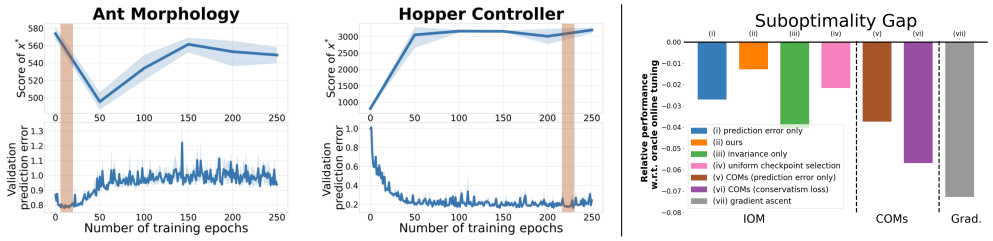

Figure 3: **Left**: Visualizing our checkpoint selection scheme on two tasks, and no matter the best checkpoint happens at the beginning or end of the training, our strategy based on validation in-distribution MSE could capture the best performance. **Right**: Offline tuning vs oracle uniform online tuning for IOM, COMs and Gradient-ascent. Among all the methods, IOM achieves the smallest regret for offline tuning, and our specific tuning strategy works best compared with other tuning strategies for IOM.

7 Discussion

In this work we study offline data-driven model based optimization (MBO), where the goal is to produce optimized designs, purely using a static dataset. We proposed to view offline data-driven model-based optimization through the lens of domain adaptation, and proposed a method that trains a model of the objective with an additional regularizer to promote invariance between the representations learned by the model in expectation under the training data distribution and the distribution of optimized points. We show that this leads to appealing theoretical bounds, a practically effective data-driven optimization method, and an appealing workflow for selecting the hyperparameters with offline data. Our method outperforms prior approaches, which illustrates the practical benefits of invariance. We also validate the effectiveness of the accompanying tuning strategy. While we evaluate our method on model-based optimization problems, our approach can in principle also be extended to contextual bandits and even to, sequential decision-making problems such as offline reinforcement learning. Exploring these broad applications of invariant representation learning for addressing distributional shift is an exciting avenue for future work. A limitation of the problem setting we study more generally is that it may not be practically feasible to just produce optimized designs in a fully data-driven manner and some online evaluation may be inevitable.

References

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

[2] 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.

[3] 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.

[4] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1): 151–175, 2010.

[5] 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.

[6] Surojit Biswas, Grigory Khimulya, Ethan C Alley, Kevin M Esvelt, and George M Church. Low-n protein engineering with data-efficient deep learning. *Nature methods*, 18(4):389–396, 2021.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[8] 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.

[9] Clara Fannjiang and Jennifer Listgarten. Autofocused oracles for model-based design. *arXiv preprint arXiv:2006.08052*, 2020.

[10] 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.

[11] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[12] 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] 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.

[14] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[15] 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.

[16] Nikolaus Hansen. The CMA evolution strategy: A comparing review. In José Antonio 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.

[17] 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.

[18] Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? *arXiv preprint arXiv:2012.15085*, 2020.

[19] Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *International conference on machine learning*, pages 3020–3029. PMLR, 2016.

[20] Fredrik D Johansson, Nathan Kallus, Uri Shalit, and David Sontag. Learning weighted representations for generalization across designs. *arXiv preprint arXiv:1802.08598*, 2018.

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

[22] Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. *NeurIPS*, 2019.

[23] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.

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

[25] 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.

[26] Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim. Representation balancing offline model-based reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QpNz8r_Ri2Y.

[27] 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.

[28] Yao Liu, Omer Gottesman, Aniruddh Raghu, Matthieu Komorowski, Aldo Faisal, Finale Doshi-Velez, and Emma Brunskill. Representation balancing mdps for off-policy policy evaluation. *arXiv preprint arXiv:1805.09044*, 2018.

[29] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[30] 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.

[31] Thanh Nguyen-Tang, Sunil Gupta, A Tuan Nguyen, and Svetha Venkatesh. Offline neural contextual bandits: Pessimism, optimization and generalization. *arXiv preprint arXiv:2111.13807*, 2021.

[32] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[33] 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.

[34] 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.

[35] 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.

[36] Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert RG Lanckriet. On integral probability metrics,\phi-divergences and binary classification. *arXiv preprint arXiv:0901.2698*, 2009.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] Liuyi Yao, Sheng Li, Yaliang Li, Mengdi Huai, Jing Gao, and Aidong Zhang. Representation learning for treatment effect estimation from observational data. *Advances in Neural Information Processing Systems*, 31, 2018.

[42] 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.

[43] Andrea Zanette. Exponential lower bounds for batch reinforcement learning: Batch rl can be exponentially harder than online rl. In *International Conference on Machine Learning*, pages 12287–12297. PMLR, 2021.

[44] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pages 7523–7532. PMLR, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discuss limitations of our work in Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work pertains to fundamental algorithm design for model-based optimization, and it does not have any particular negative societal impacts beyond those that apply more generally to model-based optimization methods.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We discuss the full set of assumptions in Section 4.3.
   (b) Did you include complete proofs of all theoretical results? [Yes] Our proofs are included in Appendix A.1.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include it with the supplementary material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We include these details in Appendix A.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We reported error bars in our results, and also followed guidelines for rigorous statistical evaluation using the protocol from Agarwal et al. [1].

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] These details can be found in Appendix A.2.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cited Design Bench.
   (b) Did you mention the license of the assets? [Yes] Details are found in Appendix A.2.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]