Appendices

A Proofs of Theoretical Results

In this section, we will provide proofs of the various theoretical results from Section 3.3. We will first discuss the assumptions and technical conditions we use, then prove the lower bound result from Proposition 3.2 and finally utilize it to prove the performance guarantee for invariant representation learning (Proposition 3.3). Before diving into the proofs, we first recall the set of assumptions and technical conditions below:

A.1 Assumptions, Technical Conditions and Lemmas

Recall that our proofs operate under the following assumptions:

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

Additionally, we will also define statistical error that will appear in the bounds:

$$\varepsilon_{\text{stat}} \approx \sqrt{\frac{\log\left(\frac{|\mathcal{F}||\Phi| c_0}{\delta}\right)}{|\mathcal{D}|}} \tag{7}$$

We will use some helpful Lemmas that we list below, which can be proven using standard results from empirical risk minimization and concentration inequalities [45].

Lemma A.1 (Concentration of expected function values)**.** *For any given $f \in \mathcal{F}$ satisfying the assumptions above, and any given distribution π over designs $\mathcal{X}$, let $J_f(\pi) := \mathbb{E}_{\mathbf{x} \sim \pi}[f(\mathbf{x})]$. Then, given a set of N samples, $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N \sim \pi$, the empirical mean $\widehat{J}_f(\pi) = \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}_i)$ converges to $J_f(\pi)$ such that, with high probability $\geq 1 - \delta$ (c_0 is a universal constant),*

$$\left| J_f(\pi) - \widehat{J}_f(\pi) \right| \lesssim \sqrt{\frac{\log\left(\frac{c_0}{\delta}\right)}{N}}. \tag{8}$$

Building on Lemma A.1, we now bound the empirical risk and the population risk of a given loss function $\mathcal{L}(f)$ at the empirical risk minimizer using a standard uniform concentration argument:

Lemma A.2 (Concentration of empirical risk minimizer)**.** *Given any bounded, Lipschitz loss function, $\mathcal{L}$ and for a given $f \in \mathcal{F}$, let, $\mathcal{L}(f) := \mathbb{E}_{\mathbf{x} \sim \mu_{data}}[L(f, \mathbf{x})]$, and let the corresponding empirical loss function be denoted by: $\widehat{\mathcal{L}}(f) := \frac{1}{|\mathcal{D}|} \sum_i L(f, \mathbf{x}_i)$. Then, with high probability $\geq 1 - \delta$,*

$$\left| \mathcal{L}\left(\arg \inf_{f \in \mathcal{F}} \widehat{\mathcal{L}}(f) \right) - \mathcal{L}\left(\arg \inf_{f \in \mathcal{F}} \mathcal{L}(f) \right) \right| \lesssim \mathcal{O}\left(||\mathcal{L}||_\infty \sqrt{\frac{1}{|\mathcal{D}|} \log\left(\frac{|\mathcal{F}| c_0}{\delta} \right)} \right). \tag{9}$$

Finally, we discuss a technical condition that will be required to prove Proposition 3.2. We consider the following "continuity" property of the function class $\mathcal{F}$ with respect to a loss function that is a mixture of two loss functions:

Definition A.3 (Continuity of $\mathcal{F}$ with respect to $\mathcal{L}_\lambda$.)**.** We say that a function class $\mathcal{F}$ is continuous with respect to a given loss function $\mathcal{L}_\lambda(f) := \mathcal{L}_1(f) + \lambda \mathcal{L}_2(f)$, where $\lambda \in \mathbb{R}^+$ is a hyperparameter, when the following holds: denoting $f_\theta^\lambda := \inf_{f \in \mathcal{F}} \mathcal{L}_\lambda(f)$, the $\lim_{\lambda \to 0} f_\theta^\lambda$ exists, and corresponds to a minimizer of $\mathcal{L}_1(f)$:

$$\lim_{\lambda \to 0} \; f_\theta^\lambda \in \arg \inf_{f \in \mathcal{F}} \mathcal{L}_1(f). \tag{10}$$

Definition A.3 formalizes the intuition that an addition of the loss $\mathcal{L}_2$ affects the optimal solution in a continuous manner, and does not abruptly change the solution. We would expect such a criterion to be satisfied when optimizing smooth loss functions (e.g., squared error) over parameters dictated by a neural network, with appropriate explicit regularization (e.g., ℓ_2 regularization on parameters) or when the training procedure smoothens the solution using some form of implicit regularization.

We will assume that the function class obtained by composing $\mathcal{F}$ and Φ, $\mathcal{F} \circ \Phi$, obeys the continuity assumption with respect to the training objective in Equation 3. Concretely, in this case $\mathcal{L}_1$ and $\mathcal{L}_2$ are given by: $\mathcal{L}_1(f, \phi) = \mathbb{E}_{\mathbf{x}, y}[(f(\phi(\mathbf{x})) - y)^2]$ and $\mathcal{L}_2(f, \phi) = \text{disc}_{\mathcal{H}}(\mathbb{P}_{\pi}(\phi(\mathbf{x})), \mathbb{P}_{\mu_{\text{data}}}(\phi(\mathbf{x})))$, and both of the loss functions are smooth with respect to the functions ϕ and f. In our experiments, we utilized a square loss for the discrepancy $\text{disc}_{\mathcal{H}}$, and therefore, it is smooth. Definition A.3 in this case would reduce to the following condition:

Assumption A.4 (Definition A.3 applied to $\mathcal{F} \circ \Phi$.)**.** For any given threshold $\delta \in \mathbb{R}$ such that $|\lambda| \leq \delta$, there exists a non-decreasing function $\zeta : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ of λ such that, $\zeta(0) = 0$ and $||f_\theta^\lambda \circ \phi_\theta^\lambda - f^* \circ \phi^*||_\infty \leq \zeta(\lambda)$ for some $(f^*, \phi^*) \in \arg\inf_{f \in \mathcal{F}, \phi \in \Phi} \mathcal{L}_1(f, \phi)$.

Essentially, Assumption A.4 suggests that if we can utilize an appropriately chosen value of λ, then we can roughly match the solution obtained by just minimizing $\mathcal{L}_1$, which is the prediction error in our case. This would allow us to control the discrepancy of a solution obtained by only optimizing the prediction error.

A.2 Proof of Proposition 3.2

Proposition A.5 (Lower bounding the value under π.)**.** *Under the assumptions and criteria listed above, 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 $\geq 1 - \delta$ as:*

$$J(\pi) - J_\theta(\pi) \geq \underbrace{J(\mu_{data}) - J_\theta(\mu_{data})}_{(\blacksquare)} - \underbrace{C_{\mathcal{F}} \cdot disc_{\mathcal{H}}(\mathbb{P}_{\mu_{data}}(\phi(\mathbf{x})), \mathbb{P}_{\pi}(\phi(\mathbf{x})))}_{(*)} - 2\varepsilon'_{\mathcal{F},\Phi} - \varepsilon_{stat},$$

where $C_{\mathcal{F}}$ and $\varepsilon'_{\mathcal{F},\phi}$ are universal constants that only depend on the function class $\mathcal{F}, \Phi$ and other universal constants associated with the loss functions, and ε_{stat} is a statistical error term that decreases as the dataset size $|\mathcal{D}|$ increases.

Proof. Our high-level strategy would be to first decompose the terms into different components, and bound these components separately. We begin with the following decomposition:

$$\begin{aligned}
&J(\pi) - J_\theta(\pi) &(11)\\
&= \mathbb{E}_{\mathbf{x} \sim \pi(\mathbf{x})}\left[f(\mathbf{x}))\right] - \mathbb{E}_{\mathbf{x} \sim \pi(\mathbf{x})}\left[f_\theta(\phi(\mathbf{x})))\right] &(12)\\
&= \underbrace{\mathbb{E}_{\mathbf{x} \sim \mu_{\text{data}}(\mathbf{x})}\left[f(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x} \sim \mu_{\text{data}}(\mathbf{x})}\left[f_\theta(\phi(\mathbf{x}))\right]}_{(a)} &(13)\\
&\quad + \underbrace{\mathbb{E}_{\mathbf{x} \sim \pi(\mathbf{x})}\left[f(\mathbf{x}))\right] - \mathbb{E}_{\mathbf{x} \sim \pi(\mathbf{x})}\left[f_\theta(\phi(\mathbf{x})))\right] - \left(\mathbb{E}_{\mathbf{x} \sim \mu_{\text{data}}(\mathbf{x})}\left[f(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x} \sim \mu_{\text{data}}(\mathbf{x})}\left[f_\theta(\phi(\mathbf{x}))\right]\right)}_{:= \Delta(\pi, \mu_{\text{data}})}.
\end{aligned}$$

Note that term (a) exactly corresponds to the first term $\blacksquare = J(\mu_{\text{data}}) - J_\theta(\mu_{\text{data}})$ in the expression in the proposition. To obtain the rest of the terms, we will now lower bound the remainder $\Delta(\pi, \mu_{\text{data}})$. First note that we can rearrange this term into a more convenient form:

$$\Delta(\pi, \mu_{\text{data}}) := -\underbrace{\left(\mathbb{E}_{\mathbf{x} \sim \pi(\mathbf{x})}\left[f_\theta(\phi(\mathbf{x}))\right] - \mathbb{E}_{\mathbf{x} \sim \mu_{\text{data}}(\mathbf{x})}\left[[f_\theta(\phi(\mathbf{x}))]\right]\right)}_{(i)} + \underbrace{\left(\mathbb{E}_{\mathbf{x} \sim \pi(\mathbf{x})}\left[f(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x} \sim \mu_{\text{data}}(\mathbf{x})}\left[f(\mathbf{x})\right]\right)}_{(ii)}. \tag{14}$$

Term (i) in the above equation can directly be bounded to give rise to one of the terms that contributes to the discrepancy term the bound as shown below:

$$\begin{aligned}
(i) &:= -\int_{\mathbf{x}} (\pi(\mathbf{x}) - \mu_{\text{data}}(\mathbf{x})) \cdot f_\theta(\phi(\mathbf{x}))\, d\mathbf{x} &(15)\\
&= -\int_{\mathbf{z}} (\pi(\mathbf{z}) - \mu_{\text{data}}(\mathbf{z})) \cdot f_\theta(\mathbf{z})\, d\mathbf{z} &(16)\\
\implies (i) &\leq ||\widehat{f}||_{\text{L}} \cdot \text{disc}_{\mathcal{H}}(\mathbb{P}_{\pi}(\phi(\mathbf{x})), \mathbb{P}_{\mu_{\text{data}}}(\phi(\mathbf{x}))), &(17)
\end{aligned}$$

where the second step (Equation 16) follows by reparameterizing the first step (Equation 15) in terms of the representation $\mathbf{z} := \phi(\mathbf{x})$ and applying probability density change of variables simultaneously with the change of variables for integration, which leads to a cancellation of the Jacobian terms. $\mathbb{P}_{\mu_{\text{data}}}(\phi(\mathbf{x}))$ denotes the probability distribution of the representations $\phi(\mathbf{x})$ when $\mathbf{x} \sim \mu_{\text{data}}(\mathbf{x})$, and $\mathbb{P}_{\pi}(\phi(\mathbf{x}))$ denotes the probability distribution of the representation $\phi(\mathbf{x})$ when $\mathbf{x} \sim \pi(\mathbf{x})$. The last step follows from the fact that the discrepancy measure, $\text{disc}_{\mathcal{H}}(p, q)$, which is an integral probability metric (IPM) [42], upper bounds the total-variation divergence between p and q.

Now we tackle term (ii). First note that under the ε-realizability assumption, we note that there exists a $g \in \mathcal{F}, \phi^* \in \Phi$ such that $f(\mathbf{x}) \approx g(\phi^*(\mathbf{x}))$ in expectation under μ_{data} and π (Assumption 3.1 holds for any distribution Π). That is,

$$(ii) \leq \left|\mathbb{E}_{\mathbf{x}\sim\pi(\mathbf{x})}\left[g(\phi^*(\mathbf{x}))\right] - \mathbb{E}_{\mathbf{x}\sim\mu_{\text{data}}(\mathbf{x})}\left[g(\phi^*(\mathbf{x}))\right]\right| + 2\varepsilon_{\mathcal{F},\Phi} \tag{18}$$

$$\leq ||g||_{\text{L}} \left|\left|\mathbb{E}_{\mathbf{x}\sim\pi(\mathbf{x})}\left[\phi^*(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x}\sim\mu_{\text{data}}(\mathbf{x})}\left[\phi^*(\mathbf{x})\right]\right|\right| + 2\varepsilon_{\mathcal{F},\Phi} \tag{19}$$

$$\leq ||g||_{\text{L}} \cdot \text{disc}_{\mathcal{H}}(\mathbb{P}_{\pi}(\phi^*(\mathbf{x})), \mathbb{P}_{\mu_{\text{data}}}(\phi^*(\mathbf{x}))) + 2\varepsilon_{\mathcal{F},\Phi}, \tag{20}$$

where the second step follows from the Lipschitzness of $\mathcal{F}$ and the third step follows similar to Equation 17.

Finally, we will show that for an appropriately chosen $\lambda > 0$, $\text{disc}_{\mathcal{H}}(\mathbb{P}_{\pi}(\phi^*(\mathbf{x})), \mathbb{P}_{\mu_{\text{data}}}(\phi^*(\mathbf{x})))$ is close to $\text{disc}_{\mathcal{H}}(\mathbb{P}_{\pi}(\phi(\mathbf{x})), \mathbb{P}_{\mu_{\text{data}}}(\phi(\mathbf{x})))$. Applying Assumption A.4, we get that, for any given λ, there exists a ζ such that:

$$||f_\theta^\lambda \cdot \phi^\lambda - f^* \cdot \phi^*||_\infty \leq \zeta(\lambda) \tag{21}$$

$$\implies ||\mathcal{L}_2(f_\theta^\lambda, \phi^\lambda) - \mathcal{L}_2(f^*, \phi^*)||_\infty \leq C_{\mathcal{L}_2} \cdot \zeta(\lambda), \tag{22}$$

where the second step follows from the fact that the loss functions, $\mathcal{L}_1$ and $\mathcal{L}_2$ (in this case, $\text{disc}_{\mathcal{H}}$ and $(\cdot)^2$) are smooth functions of their arguments, and $C_{\mathcal{L}_2} := C_{\mathcal{H}}$ is the coefficient of smoothness when $\mathcal{L}_2 = \text{disc}_{\mathcal{H}}$. Thus, we get that:

$$\left|\text{disc}_{\mathcal{H}}(\mathbb{P}_{\pi}(\phi^\lambda(\mathbf{x})), \mathbb{P}_{\mu_{\text{data}}}(\phi^\lambda(\mathbf{x}))) - \text{disc}_{\mathcal{H}}(\mathbb{P}_{\pi}(\phi^*(\mathbf{x})), \mathbb{P}_{\mu_{\text{data}}}(\phi^*(\mathbf{x})))\right| \leq C_{\mathcal{H}} \cdot \zeta(\lambda). \tag{23}$$

Finally, we need to bound $\text{disc}_{\mathcal{H}}(\mathbb{P}_{\pi}(\phi^\lambda(\mathbf{x})), \mathbb{P}_{\mu_{\text{data}}}(\phi^\lambda(\mathbf{x})))$ in terms of the discrepancy value for the ϕ learned during training. Note that ϕ is obtained by minimizing the objective in Equation 3, which is the sample-based version of the objective used for obtaining f^λ. Using the uniform concentration argument from Lemma A.2, we can upper bound $\mathcal{L}(f^\lambda, \phi^\lambda)$ in terms of $\mathcal{L}(f_\theta, \phi)$:

$$\begin{aligned}\mathbb{E}_{\mathbf{x},y}[f^\lambda(\phi^\lambda(\mathbf{x})) - y)]^2 + \lambda\text{disc}_{\mathcal{H}}(\mathbb{P}_{\pi}(\phi^\lambda(\mathbf{x})), \mathbb{P}_{\mu_{\text{data}}}(\phi^\lambda(\mathbf{x}))) \lesssim \\ \mathbb{E}_{\mathbf{x},y}[f_\theta(\phi(\mathbf{x})) - y)]]^2 + \lambda\text{disc}_{\mathcal{H}}(\mathbb{P}_{\pi}(\phi(\mathbf{x})), \mathbb{P}_{\mu_{\text{data}}}(\phi(\mathbf{x}))) + \varepsilon_{\text{stat}}.\end{aligned}$$

Now, we can use the above relationship, alongside bounding $(f_\theta(\mathbf{x}) - y)^2 \leq B_0$ (which holds since the function is bounded):

$$\text{disc}_{\mathcal{H}}(\mathbb{P}_{\pi}(\phi^\lambda(\mathbf{x})), \mathbb{P}_{\mu_{\text{data}}}(\phi^\lambda(\mathbf{x}))) \leq \text{disc}_{\mathcal{H}}(\mathbb{P}_{\pi}(\phi(\mathbf{x})), \mathbb{P}_{\mu_{\text{data}}}(\phi(\mathbf{x}))) + C_{\mathcal{H}} \cdot \zeta + \frac{2B_0}{\lambda}, \tag{24}$$

where $\zeta \to 0$ as $\lambda \to 0$. Now, we can minimize the right hand side w.r.t. λ and obtain the desired result:

$$J(\pi) - J_\theta(\pi) \geq J(\mu_{\text{data}}) - J_\theta(\mu_{\text{data}}) - (2C+1) \cdot \text{disc}_{\mathcal{H}}(\mathbb{P}_{\mu_{\text{data}}}(\phi(\mathbf{x})), \mathbb{P}_{\pi}(\phi(\mathbf{x}))) - 2\varepsilon'_{\mathcal{F},\Phi} - \varepsilon_{\text{stat}},$$

where $\varepsilon'_{\mathcal{F},\Phi} = \varepsilon_{\mathcal{F},\Phi} + h(B_0, C_{\mathcal{H}})$, where $h(\cdot)$ is a bias that is a property of the loss functions, the function classes $\mathcal{F}$ and Φ, and the discrepancy metric. It does not depend on the learned functions f_θ, ϕ learned or the distribution π. This completes the proof.

$\square$

Remark A.6 (**The interplay between λ_0, $\mathcal{F}$ and Φ.**). In the proof above, we appeal to the continuity assumption (Assumption A.4) to enable us to bound the discrepancy measured in terms of the representation ϕ^* with respect to ϕ, and this requires adjusting λ. This is unavoidable in the worst case if the function classes $\mathcal{F}$ and Φ are arbitrary and do not contain any solution ϕ^* that attains a small training error, while being somewhat invariant. In such cases, the only way to attain a lower-bound is to not be invariant, and this is what our bound would suggest by setting λ to be on the smaller end of the spectrum. On the other hand, if the function class is more expressive, such that there exists a tuple (f^*, ϕ^*) that minimizes $\mathcal{L}_1(f, \phi))$ while being invariant, i.e., $\mathcal{L}_2(f^*, \phi^*)$ is small, then we can loosen this restriction over λ.

A.3 Proof of Proposition 3.3

Proposition A.7 (Performance guarantee for IOM)**.** *Under Assumption 3.1, the expected value of the ground truth objective under μ_{OPT}, $J(\mu_{\text{OPT}})$ is lower bounded by:*

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

We restate the proposition above. There is a typo in the main paper, where the term $\varepsilon'_{\mathcal{F},\Phi}$ got deleted from the bound when moving from Proposition 3.2 to 3.3. The above restatement of Proposition 3.3 provides the complete statement.

Proof. To prove this proposition, we can utilize the result from Proposition A.5, but now we need to reason about a specific $\pi = \mu_{\text{OPT}}$, where μ_{OPT} depends on the learned f_θ and ϕ. To account for such a π, we can instead bound the improvement $J(\pi) - J_\theta(\pi)$ for the worst possible $\pi \in \Pi$. While this does not change any term from the bound in Proposition A.5 that depends on the properties of the function class $\mathcal{F}$ or the representation class Φ, this would still affect the statistical error, which would now require a uniform concentration argument over the policy class Π. Therefore, the bound from Proposition A.5 becomes:

$$J(\mu_{\text{OPT}}) - J_\theta(\mu_{\text{OPT}}) \geq J(\mu_{\text{data}}) - J_\theta(\mu_{\text{data}}) - C' \cdot \text{disc}_{\mathcal{H}}(\mathbb{P}_{\mu_{\text{data}}}(\phi(\mathbf{x})), \mathbb{P}_{\mu_{\text{OPT}}}(\phi(\mathbf{x}))) - 2\varepsilon'_{\mathcal{F},\Phi} - \varepsilon_{\text{stat}}, \tag{25}$$

where $\varepsilon_{\text{stat}}$ is given by:

$$\varepsilon_{\text{stat}} = \mathcal{O}\left(\sqrt{\frac{\log\left(\frac{|\mathcal{F}||\Phi||\Pi|c_0}{\delta}\right)}{|\mathcal{D}|}}\right). \tag{26}$$

The term $\varepsilon^2_{\text{stat}}$ in the statement is a higher order term and is always dominated by $\varepsilon_{\text{stat}}$. Rearranging the terms completes the proof. ☐

A.4 Extension to any re-weighting of the dataset

In this section, we will extend the theoretical result to compare the performace of the optimized design to the average objective value of any re-weighting of the dataset, and this by definition covers the comparison against the best design in the dataset. To do so, note that Proposition A.5 can be restated when comparing to μ_{r}, a distribution induced by any specific re-weighting of the training dataset. This reformulation would give the following bound on the error:

$$J(\pi) - J_\theta(\pi) \geq \underbrace{J(\mu_{\text{r}}) - J_\theta(\mu_{\text{r}})}_{(\blacksquare)} - \underbrace{C_{\mathcal{F}} \cdot \text{disc}_{\mathcal{H}}(\mathbb{P}_{\mu_{\text{r}}}(\phi(\mathbf{x})), \mathbb{P}_{\pi}(\phi(\mathbf{x})))}_{(*)} - 2\varepsilon'_{\mathcal{F},\Phi} - \varepsilon_{\text{stat}}. \tag{27}$$

Since, the choice of μ_{r} is arbitrary (i.e., it can be any arbitrary weighting of the dataset distribution μ_{data}), we can now basically compute the tighest lower bound by maximizing the RHS with respect to the choice of μ_{r}. This would amount to:

$$J(\pi) - J_\theta(\pi) \geq \max_{\mu_{\text{r}} \in T} \left[\underbrace{J(\mu_{\text{r}}) - J_\theta(\mu_{\text{r}})}_{(\blacksquare)} - \underbrace{C_{\mathcal{F}} \cdot \text{disc}_{\mathcal{H}}(\mathbb{P}_{\mu_{\text{r}}}(\phi(\mathbf{x})), \mathbb{P}_{\pi}(\phi(\mathbf{x})))}_{(*)}\right] - 2\varepsilon'_{\mathcal{F},\Phi} - \varepsilon_{\text{stat}}, \tag{28}$$

where T denotes the set of distributions that arising from all possible re-weightings of the training data. For the special case when μ_{r} simply corresponds to a Dirac-delta distribution centered at the best point in the training dataset, which we will denote as $\mathbf{x}_{\text{best}}$, and $\mu_{\text{r}} = \delta_{\mathbf{x}=\mathbf{x}_{\text{best}}}$, we can write down the bound as:

$$J(\mu_{\text{OPT}}) \gtrsim f(\mathbf{x}_{\text{best}}) + \underbrace{J_\theta(\pi) - \widehat{f}_\theta(\mathbf{x}_{\text{best}})}_{\text{improvement}} - \underbrace{C_{\mathcal{F}} \cdot \text{disc}_{\mathcal{H}}(\mathbb{P}_{\mu_{\text{r}}}(\phi(\mathbf{x})), \mathbb{P}_{\pi}(\phi(\mathbf{x})))}_{(*)} - 2\varepsilon'_{\mathcal{F},\Phi} - \varepsilon_{\text{stat}}. \tag{29}$$

When converting Equation 29 to a performance guarantee, note that $\varepsilon_{\text{stat}}$, which is used to bound the discrepancy $\text{disc}_{\mathcal{H}}$ in Equation 29 would not decay as the size of the dataset $|\mathcal{D}|$ increases, as our reference point is just one single point $\mathbf{x}_{\text{best}}$ in the dataset. That said, note that as the dataset size increases, the groundtruth value of $\mathbf{x}_{\text{best}}$ shall also increase which would tighten the RHS of Equation 29 as the dataset size increases.

Therefore, our bound presents a tradeoff: when comparing $J(\mu_{\text{OPT}})$ to the average value of the dataset $J(\mu_{\text{data}})$, we can attain favorable guarantees that guarantee improvement as the dataset size increases because the statistical error reduces, and on the other hand, when we compare $J(\mu_{\text{OPT}})$ to $f(\mathbf{x}_{\text{best}})$, the statistical error may not decay as the dataset size increases, however, the function value $f(\mathbf{x}_{\text{best}})$ will increase. In summary, our bound in either case is expected to improve as the dataset size increases.

Remark: Proposition 3.3 is inspired from theoretical results in offline reinforcement learning, where it is common to lower bound the improvement of the learned policy over the behavior policy (see for example, works on "safe policy improvment" [28, 39, 25, 51, 8]). When comparing to the best point in the dataset, or equivalently, the best in-support policy in the case of reinforcement learning, the bounds would inevitably become weaker, since it is not clear if *any* offline MBO or offline RL algorithm would improve over the best point in the dataset, in the worst-case problem instance. We have added this as a limitation of the theoretical results, but would remark that this limitation is also existent in other areas where one must learn to optimize decisions from offline data. Nevertheless, our empirical results do show that IOM does improve over the best point in the dataset, as shown in Table 3.

B Related Work

Model-based optimization (MBO) or black-box optimization [6, 3, 41, 40, 33, 49] 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 [6, 9, 23]. 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. This distributional shift is distinct from what is studied in prior works that tackle misspecification in GPs and bandits [34, 35, 21] or methods that tackle robustness across contexts [22] in the online setting.

To address distribution shift, several recent works have proposed to be conservative. This is majorly the case in other decision-making problems such as offline reinforcement learning (RL) [30] and contextual bandits [36], 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 different ways: either by constraining the optimized designs to lie close to the training designs [6, 9, 23, 37, 24] using generative models [20, 12], or by learning a conservative [44, 26, 25] or calibrated [10, 49] learned model. While the latter class of methods has worked well in practice [43, 26], 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 5.2) this is not easy for conservative methods. On the contrary, IOM not only does not require a generative model over the design space, which can be intractable with high-dimensions (e.g., 5000-d designs in HopperController), but also attains better performance and provides a convenient tuning 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 [18, 19, 48] and off-policy evaluation in RL [31, 29]. 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.

C Experiment Details

C.1 Tasks and Datasets

Tasks. We provide brief description of the tasks we use for our empirical evaluation and the corresponding evaluation protocol below. More details can be found be found at Trabucco et al. [43].

- The Superconductor task aims at optimizing the superconducting materials which achieves high-critical temperatures. The original features are vectors containing specific physical properties, such as the mean atomic mass of the elements. Following prior work [43], we utilize the invertible input specification by representing different superconductors via a serialization of the chemical formula. The objective y is the critical temperature the resulting material could achieve. Here, we have $\mathbf{x} \in \mathbb{R}^{86}$ and $y \in \mathbb{R}^{+}$. The original dataset in Design-Bench has 21263 samples in total. However, following prior work [44, 43, 49], we remove the top 20% from the dataset to increase the difficulty and ensure the top-performing points are not visible during learning stage. **Evaluation**: To evaluate the performance of any given design, we use the oracle function provided in design-bench, which trains a random forest model (with the default hyper-parameters provided in [14]) on the whole original dataset.
- The Ant and D'Kitty Morphology task aim at designing the morphology of a quadrupedal robot such that the robot could be crawl quickly in certain direction. The Ant/D'Kitty task corresponds to the Ant/D'Kitty robot that used as the agent. Specifically, the input is the morphology of the agent with the objective value is the average return of the agent that uses the optimal control tailored for the morphology. More details about the controller design could be found at Trabucco et al. [43]. For the dimension of the design/feature space, we have $\mathbf{x} \in \mathbb{R}^{60}$ and $\mathbf{x} \in \mathbb{R}^{56}$ for D'Kitty. The original dataset contains 25009 samples for both Ant and D'Kitty. Following prior work [44, 49], we remove the 40% of top-performing samples and use the rest as the new dataset. **Evaluation**: To evaluate any given morphology, it is passed into the MuJoCo Ant or D'Kitty environment, and a pre-trained policy is used (trained using Soft Actor-Critic for 10 million steps), the sum of rewards in a trajectory is used as the final score.
- The Hopper-Controller task aims at designing a set of weights of a particular neural network policy that achieves high return for the MuJoCo Hopper-v2 environment. Basically, the policy architecture is a three-layer neural network with 64 hidden units and 5126 weights in total with the algorithm is trained using Proximal Policy Optimization [38] (with default parameters provided in Hill et al. [16]). Specifically, the input is a vector of flattened weight tensors, with $\mathbf{x} \in \mathbb{R}^{5126}$, the objective is the expected return from a single roll-out using this specific policy, with $y \in \mathbb{R}$. In the original dataset from Design-Bench, we have 3200 data points, and we do not discard any of them due to limited size of the dataset. **Evaluation:** To evaluate the design of a specific set of weight, which deploy PPO policy with this set of weight, and the sum of rewards of 1000 steps are collected using the agent, as the final score.

Dataset. For each task, we take the training dataset from design-bench, and in addition, randomly split it into training and validation with the ratio 7:3. The information of the original dataset is given in Table 2.

Table 2: Overview of the dataset information

Dataset	Sample Size	Dimensions	Type of design space
Superconductor	17010	86	Continuous
Ant Morphology	10004	60	Continuous
D'Kitty Morphology	10004	56	Continuous
Hopper Controller	3200	5126	Continuous
ChEMBL	1093	31	Discrete

C.2 Baseline Descriptions

In this section, we briefly introduce the baseline methods for model-based optimization in an algorithmic level, all the implementation details for the baselines are referred to Trabucco et al. [43].

- **CbAS** [6] trains a density model via variational auto-encoder (VAE). Basically at each step, it updates the density model via a weighted ELBO objective which move towards better designs, which estimated by a pre-trained proxy model $\hat{f}(\mathbf{x})$; then generating new samples from the updated distributions to serve the new dataset, and repeated this process.
- **Anto.CbAS** [9] is an improved version of the CbAS. For CbAS, it uses the fixed pre-trained proxy model $\hat{f}(\mathbf{x})$ all the time, to guide the distribution towards the optimal one. However, the proxy model will suffer from distributional shift as the density model changes. Hence, Auto.CbAS corrects the distributional shift by re-training the proxy model under the new optimized density model, via importance sample weights.
- **MINs** [23] learns an inverse mapping f^{-1} from the objective value y to the corresponding design $\mathbf{x}$. The inverse map is trained using the conditional GAN. Then it tries to search for the optimal y value during the optimization, and the optimized design is found by sampling from this inverse map.
- **COMs** [44] learns a conservative proxy model $\hat{f}(\mathbf{x})$. Compared with the vanilla gradient ascent, COMs augments the regression objective with an additional regularization term which aims to penalize the over-estimation error of the out-of-distribution points. The final optimal designed point is found by performing gradient ascent on this learned proxy model.
- **RoMA** [49] tries to overcome the brittleness of the deep neural networks that used in the proxy model. Basically, the method is a two-stage process, with the first stage is to train a proxy model $\hat{f}$ with Gaussian smoothing of the inputs, and the second stage is to update the solution while simultaneously adapt the model to achieve the smoothness in the input level for the current optimized point.
- **BO-qEI** [47] refers the Bayesian Optimization with quasi-Expected Improvement acquisition function. The method tries to fit the proxy model $\hat{f}$, and maximizing it via Gaussian process.
- **CMA-ES** [15] refers the covariance matrix adaptation. The idea is to maintain a Gaussian distribution of the optimized design. At each step, samples of designs are drawing from the Gaussian distribution, and the covariance matrix is re-fitted by the top-performing designs estimated by a pre-trained proxy model $\hat{f}(\mathbf{x})$.
- **Grad./Grad.Min/Grad.Mean** [44] are a family of methods that use gradient ascent to find the optimal design. All of them require a pre-trained proxy model. For Grad., the proxy model is trained by minimizing the L_2 norm of $\hat{f}(\mathbf{x})$ and the ground-truth value y. For Grad.Min, it learns 5 different proxy models, and use the minimum of it as the predicted value. For Grad.Mean, it uses the mean of the 5 models. In the optimization stage, gradient ascent on the proxy model is used to find the optimized point.
- **REINFORCE** [46] pre-trains a proxy model $\hat{f}(\mathbf{x})$ and use the policy gradient method to find the optimal design distributions that maximizes the $\hat{f}(\mathbf{x})$.

C.3 Implementation Details

For all the baseline methods, we directly report performance numbers associated with Design-Bench [43]. Below, we now discuss implementation details for the method developed in this paper, IOM. Our implementation builds off of the official implementation of COMs, provided by the authors Trabucco et al. [44]

Data preprocessing. Following prior work [44], we perform standardization for both the input $\mathbf{x}$ and objective value y before training the objective model $\hat{f}$. Essentially, we standardize $\mathbf{x}$ to $\tilde{\mathbf{x}} := \frac{\mathbf{x}-\mu_{\mathbf{x}}}{\sigma_{\mathbf{x}}}$, with $\mu_{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$ stands for the empirical mean and variance for $\mathbf{x}$, similarly for $\tilde{y}$ with $\tilde{y} := \frac{y-\mu_y}{\sigma_y}$.

Neural networks representing $\widehat{f}$ and ϕ. We parameterize the learned model $\widehat{f}$ and the representation $\phi(\cdot) \in \mathbb{R}^{64}$ using three hidden layer neural networks, that assume identical architectures for all the tasks. The representation network, $\phi_\eta(\mathbf{x})$ is a neural network with two hidden layers of size 2048, followed by Leaky ReLU (using default leak value of 0.3). The representation is of size 128. In some preliminary experiments, we observed that our method, IOM was pretty robust to the dimension of the representation, so we simply chose to utilize 128-dimensional representations for all the task, with no tuning. For the network representing $\widehat{f}$, we utilize a neural network with two hidden layers of size 1024, followed by Leaky ReLU (using default leak 0.3), the output is a 1-dimensional scalar used to predict the value y.

We utilize the χ^2-discrepancy, $\mathrm{disc}_{\mathcal{H}}$ in our experiments: $\mathrm{disc}_\phi(p, q) := \chi^2([\phi(\mathbf{x})]_{\mathbf{x}\sim p}, [\phi(\mathbf{x})]_{\mathbf{x}\sim q})$. This divergence can be instantiated in its variational form via a least-squares generative adversarial network (LS-GAN) [32] 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) [13], but found the χ^2-divergence to be the best performing version across the board.

Hyperparameters. We utilize identical hyperparameters for all tasks, and these hyperparameters are taken directly from the implementation of Trabucco et al. [44]. For training the learned model $f_\theta(\phi(\cdot))$, we use Adam optimizer with default learning rate 0.001 and $(\beta_1, \beta_2) = (0.9, 0.999)$. The batch size is 128 and the number of training epochs is 300. For training the representation network, the optimizer is Adam with learning rate 0.0003 and default $(\beta_1, \beta_2) = (0.9, 0.999)$. We represent μ_{OPT} as a non-parameteric distribution represented by 128 particles, initialized from 128 randomly sampled inputs $\mathbf{x}$ in the dataset. We then alternate between performing one gradient step on μ_{OPT} and one gradient step on f_θ and ϕ. To select a value for the hyperparameter λ (weight for the invariance regularizer term), we run our method with a wide range of λ values $\Lambda = \{0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 100.0\}$. In the main experimental results we report, following the convention in prior work [43], we choose λ to be uniform across all tasks, but λ is chosen via online evaluation. For our tuning procedure, we apply our tuning procedure where we keep the top 45% models which achieve a discriminator loss closes to 0.25, and select the one with the smallest validation prediction error among it . We perform an ablation of this strategy and the quantile of models that are kept in Section 5.2.

For the IOM-C variant we show in our experiments, we set the initial value of the Lagrange multiplier $\alpha = 0.3$–and this value is taken directly from Trabucco et al. [44]–and update it using Adam optimizer with learning rate 0.01.

Offline Hyperparameters tuning 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.

We release the code for our method along with additional details at the following anonymous website: `https://sites.google.com/view/iom-neurips`.

C.4 Overall Performance Comparisons

In Table 3, we provide the comparison of IOM and IOM-C with all baselines (including RoMA). To understand the trend in aggregated results, please check the plots in Figure 4.

D Additional results for early stopping the optimization of μ_{OPT}

We provide additional results for the effectiveness of our tuning strategy for early stopping. Results for the additional two tasks are presented in Figure 5. The top row shows the objective value of the optimized point as a function of the number of training epochs, and the bottom row shows the prediction error on the validation set. We use the vertical line to denote the point with the smallest MSE by our early stopping strategy. Among all tasks, our strategy is able to find a good checkpoint and this verifies the efficacy of our approach on different domains.

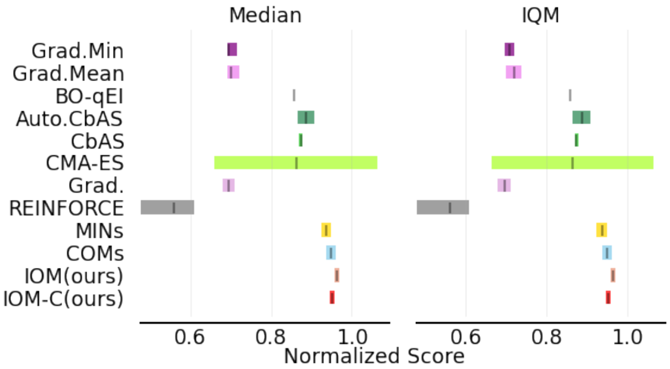

Figure 4: Median and IQM [1] (with 95% Stratified Bootstrap CIs) for the aggregated normalized score on Design-bench, for all the baseline methods. As we could not find the individual runs for RoMA [49], we report the mean and standard deviation for RoMA (copied from the results in Yu et al. [49]) in Appendix C.4.

Table 3: **Comparative evaluation of IOM and IOM-C against prior methods** in terms of the mean 100th-percentile score (following the protocol in Trabucco et al. [43]) and its standard deviation over 5 trials (the number for ROMA is copied from Yu et al. [49], and it is over 16 trails). We have now additionally evaluated the LCB+GP baseline that optimizes the design against the lower-confidence bound estimate obtained from the posterior of a GP fit to the training data. Observe that IOM outperforms this approach as well.

	Superconductor	**Ant Morphology**	**D'Kitty Morphology**	**Hopper Controller**
$\mathcal{D}$ (**best**)	0.399	0.565	0.884	1.0
Auto. CbAS	0.421 ± 0.045	0.882 ± 0.045	0.906 ± 0.006	0.137 ± 0.005
CbAS	**0.503 ± 0.069**	0.876 ± 0.031	0.892 ± 0.008	0.141 ± 0.012
MINs	0.469 ± 0.023	0.913 ± 0.036	0.945 ± 0.012	0.424 ± 0.166
COMs	0.439 ± 0.033	0.944 ± 0.016	0.949 ± 0.015	**2.056 ± 0.314**
RoMA	**0.562 ± 0.030**	0.875 ± 0.013	**1.036 ± 0.042**	1.867 ± 0.282
BO-qEI	0.402 ± 0.034	0.819 ± 0.000	0.896 ± 0.000	0.550 ± 0.118
CMA-ES	0.465 ± 0.024	**1.214 ± 0.732**	0.724 ± 0.001	0.604 ± 0.215
Grad.	**0.518 ± 0.024**	0.293 ± 0.023	0.874 ± 0.022	1.035 ± 0.482
Grad. Min	0.506 ± 0.009	0.479 ± 0.064	0.889 ± 0.011	**1.391 ± 0.589**
Grad. Mean	0.499 ± 0.017	0.445 ± 0.080	0.892 ± 0.011	**1.586 ± 0.454**
REINFORCE	0.481 ± 0.013	0.266 ± 0.032	0.562 ± 0.196	-0.020 ± 0.067
LCB+GP	N/A	0.765 ± 0.056	0.871 ± 0.009	1.736 ± 0.261
IOM (Ours)	**0.504 ± 0.040**	0.977 ± 0.009	0.949 ± 0.010	**2.444 ± 0.080**
IOM-C (Ours)	**0.511 ± 0.037**	0.966 ± 0.012	0.940 ± 0.007	**1.482 ± 0.650**

D.1 Performance of IOM and IOM-C under different invariance regularizer λ

In this section, we provide the final performance of IOM and IOM-C, under different values of the hyperparameter λ used for training (Equation 4). Results for all tasks are displayed in Figure 10. As is the case with any other prior method for data-driven decision making that adds a regularizer, both IOM and IOM-C are also sensitive to the hyper-parameter λ, which demonstrates the necessity of our proposed tuning scheme. And fortunately, as observed in Section 5, our tuning scheme can indeed attain good performance, for many choices of the ϵ (see Figure 3), making it easier to tune this hyperparameter λ in practice.

E Additional Results

Computational cost of tuning IOM via our offline workflow. In order to tune IOM via our offline workflow we first run IOM with multiple different hyperparameters. Concretely, in our experiments, we use 7 different hyperparameters, and then apply our offline tuning strategy, which only utilizes the loss values of the various runs, without incurring any significant computational cost of its own.

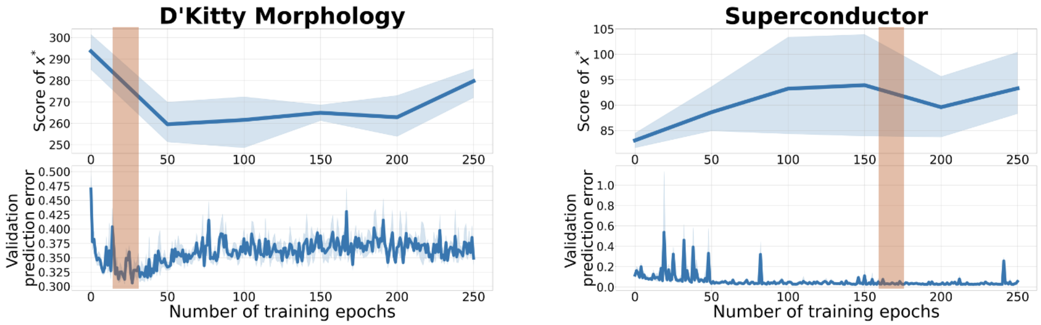

Figure 5: **Visualizing our early stopping scheme on two more tasks.** As per Section 4, the checkpoint selected is the epoch that has the smallest validation prediction error, and early stopping at this point generally performs well.

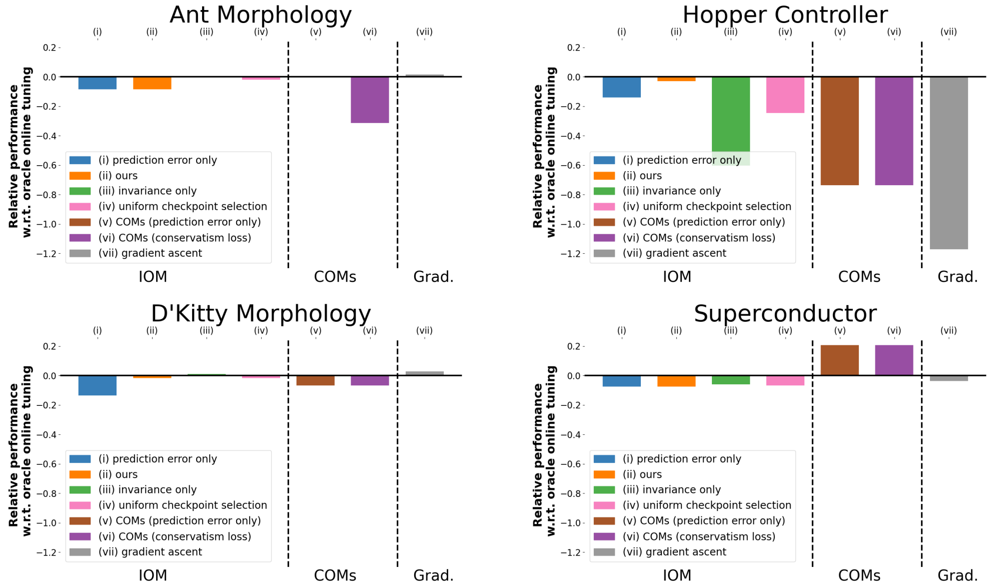

Figure 6: Suboptimality gaps of various offline tuning strategies for IOM when compared in each individual task.

Thus, our hyperparameter tuning strategy does not incur additional computational cost. When it comes to the cost of running IOM, we expect that beyond implementation differences, the cost of running IOM should not be significantly worse than COMs: the only additional component trained by IOM is the discriminator that is used to compute the χ^2-divergence on the representation, which in our implementation only adds 2 more fully connected layers.

COMs lead to worse training prediction error and more non-smooth functions compared to IOM. Next we attempt to gain insights into what might explain why IOM outperforms COMs in several of our tasks. We suspect that this is because applying a conservatism regularizer on the learned function value itself, like COMs [44], does in fact distort the learned objective function more than IOM. To understand if this is the case empirically, we plot two quantities: **(a)** the value of the training prediction error: $\mathbb{E}_{\mathbf{x},y\sim\mathcal{D}}\left[(\widehat{f}_\theta(\mathbf{x})-y)^2\right]$ for the best hyperparameter for COMs and IOM in Figure 7, and **(b)** the "smoothness" of the learned function w.r.t. input, measured via the norm of the gradient of f_θ with respect to $\mathbf{x}$, i.e., $\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[||\nabla_x f_\theta(\phi(\mathbf{x})||_2^2\right]$ in Figure 8. Observe that in both the Ant Morphology and Hopper Controller tasks, IOM is able to minimize the prediction error better than COMs, and the gradient norm for COMs is strictly higher than IOM indicating that conservatism on learned function leads to more non-smooth functions and reduces the ability of the learned function to fit the offline dataset.

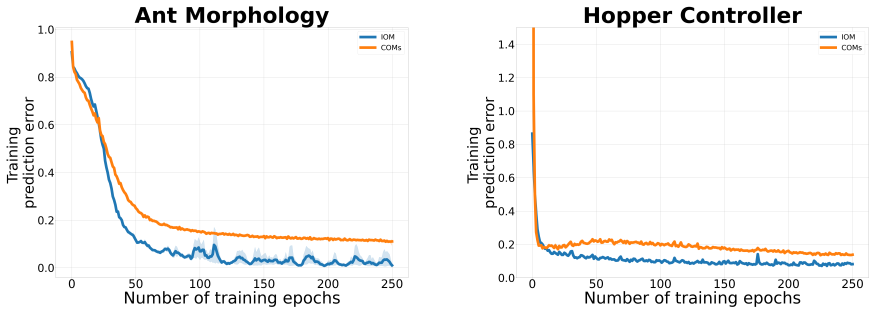

Figure 7: **Comparison of prediction error between IOM and COMs.** Note that COMs attains a worse prediction error than IOM.

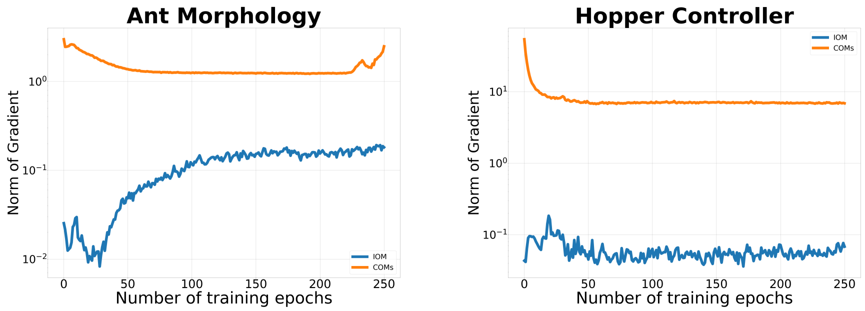

Figure 8: **Comparison of gradient norm between IOM and COMs during training (log scale).** Note that the gradient norm for COMs is clearly higher, indicating that COMs learn a more non-smooth function than IOM.

F Intuition Behind How IOM Works

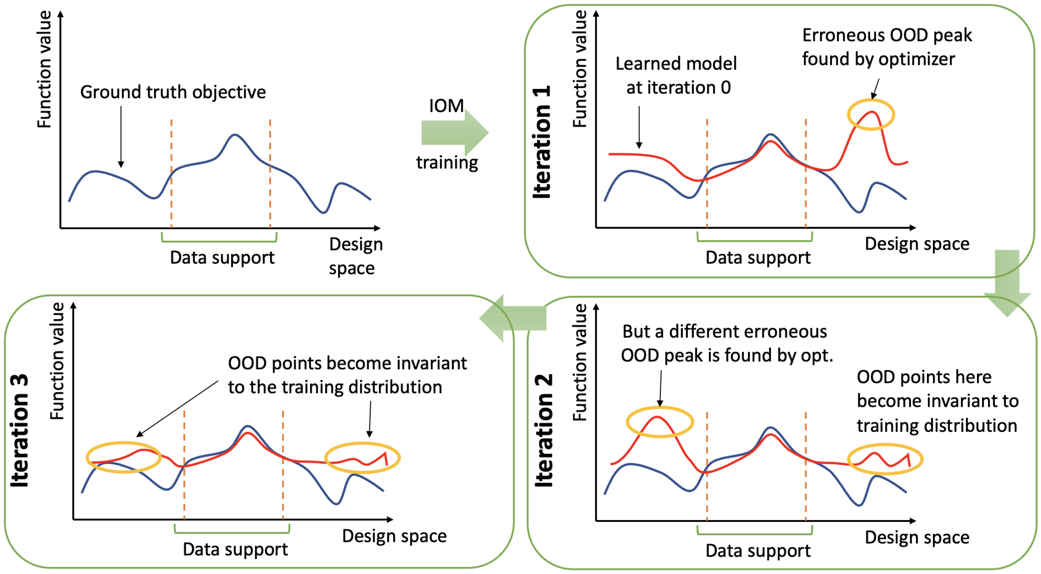
Function value
Ground truth objective
Data support
Design space
IOM
training
Iteration 1
Learned model at iteration 0
Erroneous OOD peak found by optimizer
Iteration 2
But a different erroneous OOD peak is found by opt.
OOD points here become invariant to training distribution
Iteration 3
OOD points become invariant to the training distribution

Figure 9: **Figure illustrating the intuition behind IOM.** When we apply the invariance regularizer prescribed by IOM during training, this invariance regularizer would attempt to make the distribution of the representation of the out-of-distribution (OOD) designs found by the optimizer match the distribution of the representations in the training dataset, which would make these points take on a function value that matches the average dataset value. By running this training multiple times, the IOM training procedure would gradually suppress the value of all out-of-distribution erroneous peaks in the learned function, and will enable us to find an optimal design that lies within the training distribution (though not necessarily in the training dataset.)

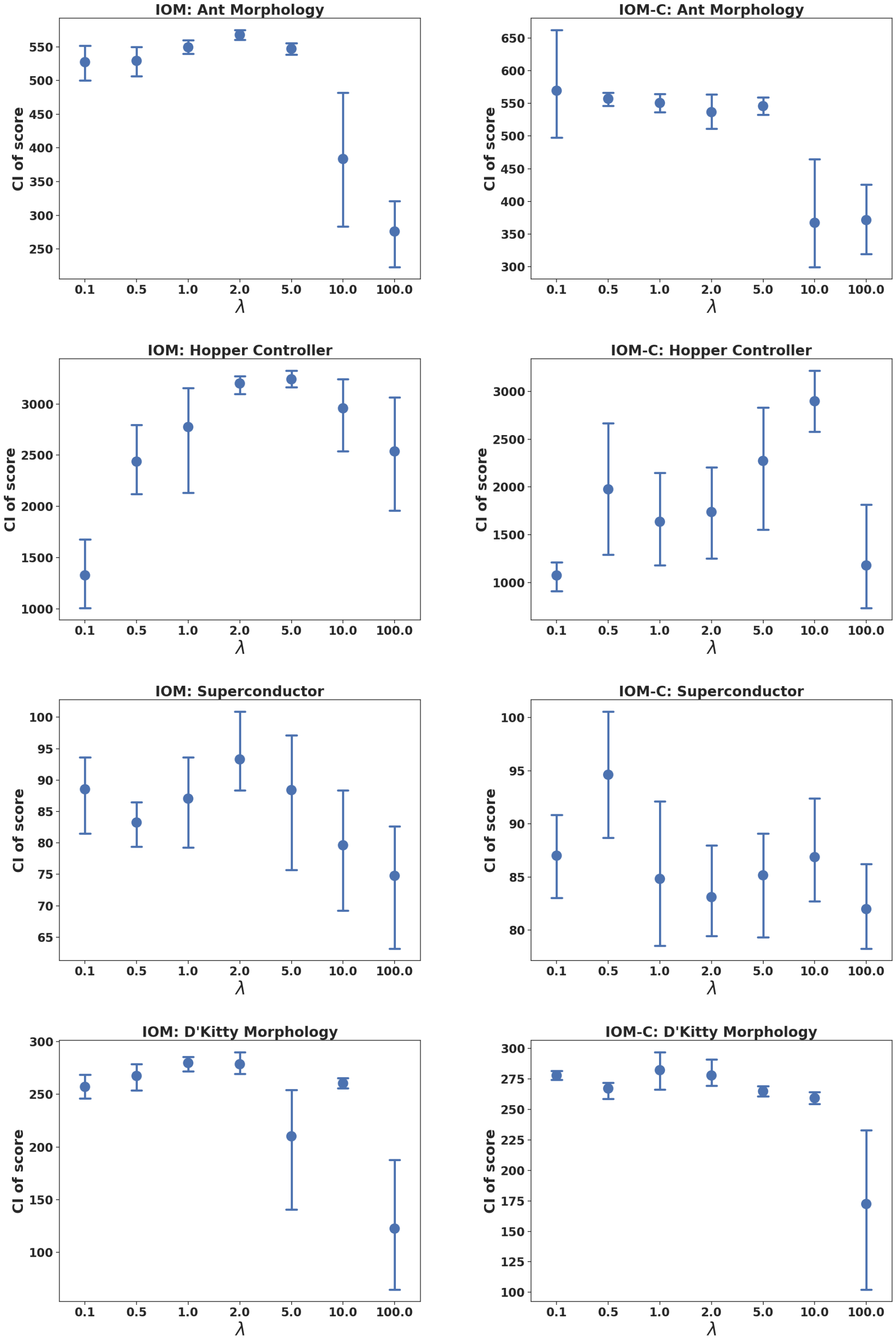

Figure 10: Score of final optimized $\mathbf{x}^*$ (with 2 standard deviations) of IOM and IOM-C under different invariance regularizer λ