Exploring validation metrics for offline model-based optimisation

Anonymous authors
Paper under double-blind review

Abstract

In offline model-based optimisation (MBO) we are interested in using machine learning to design candidates that maximise some measure of desirability through an expensive but real-world scoring process. Offline MBO tries to approximate this expensive scoring function and use that to evaluate generated designs, however evaluation is non-exact because one approximation is being evaluated with another. Instead, we ask ourselves: if we did have the real world scoring function at hand, what cheap-to-compute validation metrics would correlate best with this? Since the real-world scoring function is available for simulated MBO datasets, insights obtained from this can be transferred over to real-world offline MBO tasks where the real-world scoring function is expensive to compute. To address this, we propose a conceptual evaluation framework that is amenable to measuring extrapolation, and demonstrate this on two conditional variants of denoising diffusion models. Empirically, we find that two of the proposed validation metrics correlate very well with the ground truth. Furthermore, an additional analysis reveals that controlling the trade-off between sample quality and diversity (via classifier guidance) is extremely crucial to generating high scoring samples.

1 Introduction

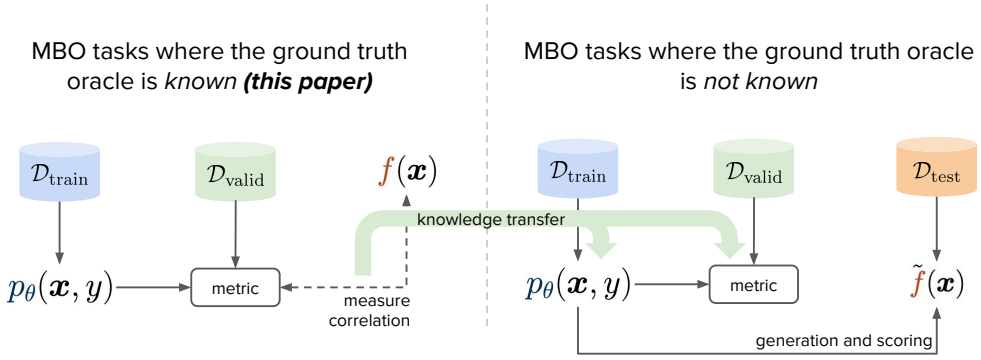


Figure 1: We want to produce designs $\boldsymbol{x}$ that score high according to the ground truth oracle $y = f(\boldsymbol{x})$, but this is usually prohibitively expensive to compute since it involves executing a real-world process. If we instead considered datasets where the ground truth oracle is cheap to compute (for instance simulations), we can search for cheap-to-compute validation metrics that correlate well with the ground truth. In principle, this can facilitate faster and more economical generation of novel designs for real-world tasks where the ground truth oracle is expensive to compute.

In model-based optimisation (MBO), we wish to learn a model of some unknown objective function $f : \mathcal{X} \to \mathcal{Y}$ where f is the ground truth 'oracle', $\boldsymbol{x} \in \mathcal{X}$ is some characterisation of an input and $y \in \mathbb{R}^+$ is some score

assigned to the input. The larger the score is, the more desirable $\boldsymbol{x}$ is according to some desiderata. In practice, such a function (a real world process) is often prohibitively expensive to compute because it involves executing a real-world process. For instance if $\boldsymbol{x} \in \mathcal{X}$ is a specification of a protein that binds to a specific receptor and we want to maximise the binding potential, then actually determining this would involve synthesising and testing it in a wet lab. In other cases, synthesising and testing a candidate may also be dangerous, for instance components for vehicles or aircraft. In MBO, we want to learn models that can *extrapolate* – that is, generate inputs whose scores are beyond that of what we have seen in our dataset. Because the most reliable way of validating these extrapolated designs also happens to be the most expensive (i.e. evaluating the ground truth oracle), we are motivated to search for metrics that correlate well with this function because this can translate to substantial economic savings.

We specifically consider the subfield of MBO that is *data-driven* (leverages machine learning) and is *offline*. Unlike online, the offline case doesn't assume an active learning loop where the ground truth can periodically be queried for more labels. Since we only deal with this particular instantiation of MBO, for the remainder of this paper we will simply say 'MBO' instead of 'offline data-driven MBO'. In MBO, very simple approach to generation is to approximate the ground truth oracle f by training an approximate oracle $f_\theta(\boldsymbol{x})$ (a classifier) from some dataset $\mathcal{D}$, and exploiting it through gradient ascent to generate a high-scoring candidate:

$$\begin{aligned} \boldsymbol{x}^* &= \arg\max_{\boldsymbol{x}} f(\boldsymbol{x}) \approx \arg\max_{\boldsymbol{x}} f_\theta(\boldsymbol{x}), \\ &\implies \boldsymbol{x}_{t+1} \leftarrow \boldsymbol{x}_t + \alpha \nabla_{\boldsymbol{x}} f_\theta(\boldsymbol{x}), \text{ for } t = \{1, \dots, T\}, \end{aligned} \tag{1}$$

The issue here however is that for most problems, this will produce an input that is either *invalid* (e.g. not possible to synthesise) or is poor yet receives a large score from the approximate oracle (*overestimation*). This is the case when the space of valid inputs lies on a low-dimensional manifold in a much higher dimension space (Kumar & Levine, 2020). How these problems are mitigated depends on whether one approaches MBO from a discriminative modelling point of view (Fu & Levine, 2021; Trabucco et al., 2021; Chen et al., 2022) (this usually involves regularising the oracle f_θ), or a generative modelling one (Brookes et al., 2019; Fannjiang & Listgarten, 2020; Kumar & Levine, 2020), where learning some notion of the data distribution $p(\boldsymbol{x})$ is emphasised. In the case of the latter, the approximate oracle $f_\theta(\boldsymbol{x})$ can be thought of as parameterising its *probabilistic* form $p_\theta(y|\boldsymbol{x})$. This, combined with an approximation of the *unconditional* data distribution $p_\theta(\boldsymbol{x})$ defines a joint density over the data $p_\theta(\boldsymbol{x}, y) = p_\theta(\boldsymbol{x})p_\theta(y|\boldsymbol{x})$. For the remainder of this paper, we will describe our evaluation framework with such statistical language.

1.1 Difficulties in evaluation

To understand some of the difficulties in MBO evaluation, we give a quick refresher on a typical training and evaluation pipeline in machine learning. Most pipelines are based on maximum likelihood estimation which the goal is to find some parameter set θ which maximises the expected probability (likelihood) of samples from the ground truth with respect to $p_\theta(\boldsymbol{x}, y)$. Given a finite number of samples $\mathcal{D}$ from the ground truth, it is common practice to split such a dataset into three sets: training $\mathcal{D}_{\text{train}}$, validation $\mathcal{D}_{\text{valid}}$, and test $\mathcal{D}_{\text{test}}$. Models are trained to maximise the expected likelihood over $\mathcal{D}_{\text{train}}$ and are validated on $\mathcal{D}_{\text{valid}}$, and this usually comprises a back-and-forth process. Whatever is considered to be the best model via the validation set is then used to compute a final likelihood (or related metric) on the test set $\mathcal{D}_{\text{test}}$. The latter is a convenient consequence of the theory underlying maximum likelhood estimation: the ground truth generating process $p(\boldsymbol{x}, y)$ *isn't necessary* to see how well $p_\theta(\boldsymbol{x}, y)$ performs when all is said and done – the theory simply says we just need a sufficiently large number of samples from it.

Conversely, in MBO training and validation play a similar role as mentioned earlier, but ultimately we would like to *synthesise* new examples that are intended to be *high scoring*, and we can think of this as being analogous to generating our own test set, but it is one which does not come from the same distribution as the training set. Since these examples are generated with an approximating model (e.g. f_θ), the only way we can be certain what their true y's actually are is via the ground truth oracle f, which we don't have access to. While one *could* approximate ground truth oracle with an evaluation oracle trained on a held-out test set, the issue is now that we are evaluating one approximation (our generative model) with another approximation.

While this is an inevitable dilemma in offline MBO, we can still try to mitigate this uncertainty by carefully thinking about how we should evaluate our models.

Let us assume that we did indeed have access to the ground truth oracle f and that it was cheap to compute. We could ask ourselves the question: **what validation metrics correlate the best with the ground truth**? Here, a *validation metric* is some measure of desirability of the generative model that is cheap to compute and is a function of $\mathcal{D}_{\text{valid}}$. We note that there is a circular issue at play here: to answer this question, the ground truth f needs to be cheap to compute, which negates the need for validation proxies in the first place. However, we believe that if the study we propose is conducted at scale with a sufficiently large number of datasets that are both difficult and diverse (for which the ground truth is cheap to evaluate), this knowledge can be *transferred* to real world problems where the ground truth oracle is too expensive to liberally execute. By 'transfer', we do not mean that pertaining to transfer learning (at least not primarily), but rather the transfer of *human knowledge* on how to best train MBO algorithms, and this is illustrated this in Figure 1. We solidify our point further by noting that a similar analogy can be made for reinforcement learning (RL). In RL, the real world 'reward function' is usually expensive or dangerous to compute, but a substantial literature is concerned with training and evaluating agents in simulated environments where the cost of evaluating the reward function is significantly cheaper and safer (Todorov et al., 2012; Brockman et al., 2016). These simulated environments can either be used for pre-training ('sim2real' knowldge transfer), or through human knowledge (figuring out which algorithms are most robust to deploy in the real world).

1.2 Contributions

We lay out our contributions as follows:[1]

- We propose a conceptual evaluation framework for *generative models* in offline MBO, where we would like to find *cheap-to-compute* validation metrics that correlate well with the ground truth oracle. While this requires access to the ground truth oracle, we believe that empirical insights derived from this evaluation framework can be transferred to MBO tasks where the ground truth oracle is too expensive to compute (see Figure 1).
- While our proposed evaluation framework is agnostic to the class of generative model, we specifically demonstrate it using the recently-proposed class of denoising diffusion probabilistic models (DDPMs) (Ho et al., 2020). For this class of model, we examine two conditional variants: classifier-based (Dhariwal & Nichol, 2021) and classifier-free (Ho & Salimans, 2022) guidance. Since DDPMs appear to be relatively unexplored in MBO, we consider our empirical results on these class of models to be an orthogonal contribution.
- We explore five validation metrics in our work against four datasets in the Design Bench (Trabucco et al., 2022) framework, and explain their pros and cons. Empirically, we find that for the classifier-free variant, two of the five metrics (score and agreement) correlate consistently well with the ground truth. For classifier-based, there are almost no differences, and we conjecture why this is the case.
- Furthermore, we find that generating good samples is a careful trade-off between sample quality and sample diversity, which underlies a common dilemma in generative modelling. We show that this trade-off can be expressed easily for either conditional denoising diffusion model variants as a simple hyperparameter.

2 Proposed framework

We first motivate our framework through a statistical lens. Let us assume we have access to some learned $p_\theta(\boldsymbol{x}, y)$. We know that the probability of sampling an $(\boldsymbol{x}, y)$ is proportional to its predicted *likelihood*. However, samples that are 'likely' in nature would not be high scoring, otherwise MBO would not be needed in the first place. Conversely, the high scoring examples lie in low density regions of the real data distribution. Therefore, we have to be careful in how we use our generative model to sample. Because of this, we do not

[1]Furthermore, an anonymised version of our code can be found here: `https://github.com/anonymouscat2434/exploring-valid-metrics-mbo`

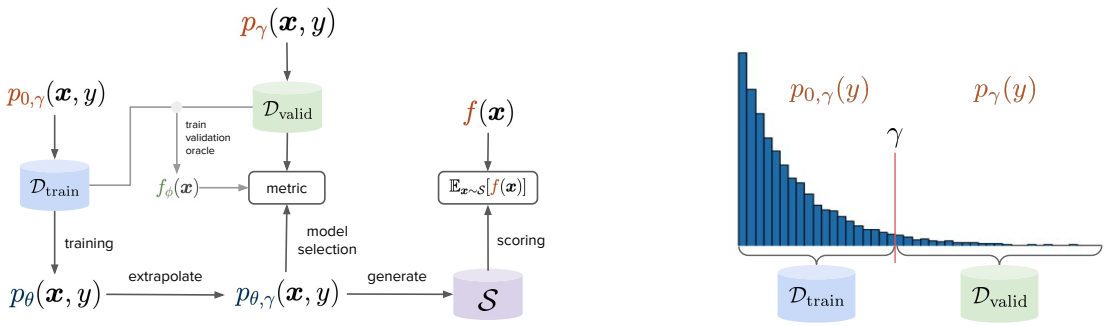


Figure 2: (2a): Our framework is split into four parts: training / extrapolation (2.1), model selection and generation (2.2). Model selection involves selecting for the model θ which minimises a validation metric, which in turn may be either a function of $\mathcal{D}_{\text{valid}}$, the approximate validation oracle, f_ϕ, or both. (2b): given the full dataset $\mathcal{D}$, we split the dataset such that all examples with y's less than or equal to γ are assigned to $\mathcal{D}_{\text{train}}$, and those greater than γ assigned to the validation set $\mathcal{D}_{\text{valid}}$. These splits can be seen as samples from their respective 'truncated' ground truth distributions.

wish to use likelihood to select for *likely candidates*; instead, we wish to use it as a means to measure how well a generative model can *extrapolate*, and this is crucial when it comes to performing model selection on a validation set. One way we can measure extrapolation is to evaluate likelihood (or some proxy of it) on *high scoring* examples, examples whose scores have *never been observed* during training. We can achieve this through careful construction of our training and validation sets without having to leave the offline setting. Assume the full dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i\}_{i=1}^n$ and $(\boldsymbol{x}, y) \sim p(\boldsymbol{x}, y)$, the ground truth joint distribution. Given some threshold γ, we can imagine dealing with two truncated forms of the ground truth $p_{0,\gamma}(\boldsymbol{x}, y)$ and $p_\gamma(\boldsymbol{x}, y)$, where:

$$p_{0,\gamma}(\boldsymbol{x}, y) = \{(\boldsymbol{x}, y) \sim p(\boldsymbol{x}, y) | y \in [0, \gamma]\} \tag{2}$$

$$p_\gamma(\boldsymbol{x}, y) = \{(\boldsymbol{x}, y) \sim p(\boldsymbol{x}, y) | y \in (\gamma, \infty]\} \tag{3}$$

Therefore, if we split $\mathcal{D}$ based on γ then we can think of the left split $\mathcal{D}_{\text{train}}$ as a finite collection of samples from $p_\gamma(\boldsymbol{x}, y)$ and the right split $\mathcal{D}_{\text{valid}}$ is for $p_{0,\gamma}(\boldsymbol{x}, y)$. This is illustrated in Figure 2b.

2.1 Training and extrapolation

Let us assume a conditional generative model of the form $p_\theta(\boldsymbol{x}|y)$, which has been trained on $\mathcal{D}_{\text{train}}$. If we denote the distribution over y's in the training set as $p_{0,\gamma}(y)$ then through Bayes' rule we can write the joint likelihood as: $p_\theta(\boldsymbol{x}, y) = p_\theta(\boldsymbol{x}|y)p_{0,\gamma}(y)$. This equation essentially says: to generate a sample $(\boldsymbol{x}, y)$ from this joint distribution, we first sample $y \sim p_{0,\gamma}(y)$, then we sample $\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|y)$. Decomposing the joint distribution into a likelihood and a prior over y means that we can change the latter at any time. For instance, if we wanted to construct an 'extrapolated' version of this model, we can simply replace the prior in this equation with $p_\gamma(y)$, which is the prior distribution over y for the validation set. We define this as the *extrapolated model* (Figure 2, *extrapolate* caption):[2]

$$p_{\theta,\gamma}(\boldsymbol{x}, y) = p_\theta(\boldsymbol{x}|y)p_\gamma(y). \tag{4}$$

Of course, it is not clear to what extent the extrapolated model would be able to generate high quality inputs from y's much larger than what it has observed during training. This is why we need to perform model selection via the use of some validation metric that characterises the model's ability to extrapolate.

[2]There are other ways to infer an extrapolated model from a 'base' model, and we describe some of these approaches in Section 3.

2.2 Model selection and generation

Suppose we trained many models, each differing by their set of hyperparameters. If we denote the j'th model's weights as θ_j, then model selection amounts to selecting a $\theta^* \in \Theta = \{\theta_j\}_{j=1}^m$ such that a validation metric is minimised. The simplest example would be the expected log likelihood over samples in the validation set:

$$\theta^* = \arg\min_{\theta \in \Theta} \frac{1}{|\mathcal{D}_{\text{valid}}|} \sum_{(\boldsymbol{x},y) \in \mathcal{D}_{\text{valid}}} \log p_\theta(\boldsymbol{x}, y). \tag{5}$$

Since we would like to select for models that generate high scoring candidates, we could devise a validation metric which favours models whose extrapolated variants (Equation 4) generate the highest scoring candidates with respect to the validation oracle. We could express this as the following:

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{\tilde{\boldsymbol{x}}, y \sim p_{\theta,\gamma}(\boldsymbol{x},y)} f_\phi(\tilde{\boldsymbol{x}}) = \arg\min_{\theta \in \Theta} \mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_\theta(\boldsymbol{x}|y), y \sim p_\gamma(y)} f_\phi(\tilde{\boldsymbol{x}}). \tag{6}$$

In order to be consistent with the evaluation setup of Design Bench, we consider a variant of Equation 6 where the expectation is computed with respect to the 'best' 128 candidates output by the model. Here, 'best' is a criteria that is up to the practioner to specify, and we use a simple criteria for ours: simply generate a sufficient large set candidates by sampling from $p_{\theta,\gamma}$ (we do 128×10), then sort those candidates by their predicted validation oracle scores and retain the top 128 candidates. This can be written as the following equation:

$$\boxed{\mathcal{M}_{\text{score}}(\mathcal{S}; \theta, \phi) = \frac{1}{K} \sum_{i=1}^{K} f_\phi\big(\text{sorted}(\mathcal{S}; f_\phi)_i\big), \quad \mathcal{S}_i \sim p_{\theta,\gamma}(\boldsymbol{x}, y)} \tag{7}$$

where $\text{sorted}(\mathcal{S}, f_\phi)$ sorts $\mathcal{S} = \{\tilde{\boldsymbol{x}}_i\}_i$ in descending order via the magnitude of prediction. This is the first validation metric we propose, and a few more will be introduced over the course of this paper. A summary of these metrics is described in Table 1.

To determine which validation metrics correlate the most with the ground truth, we first need to devise a *test metric*. This test metric is the same as Equation 7 but with the validation oracle in the outer parentheses replaced by the test oracle:

$$\boxed{\mathcal{M}_{\text{test-score}}(\mathcal{S}; \theta, \phi) = \frac{1}{K} \sum_{i=1}^{K} f\big(\text{sorted}(\mathcal{S}; f_\phi)_i\big), \quad \mathcal{S}_i \sim p_{\theta,\gamma}(\boldsymbol{x}, y)} \tag{8}$$

3 Related work

Design Bench Design Bench is an evaluation framework by Trabucco et al. (2022) that facilitates the training and evaluation of MBO algorithms. Design Bench, as of time of writing, provides *four discrete* and *four continuous* datasets. These datasets can be further categorised based on two attributes: whether a ground truth oracle exists or not, and whether the input distribution is fully observed (i.e. the combined train and test splits contain all possible input combinations). In terms of evaluation, Design Bench does not prescribe a validation set (only a training set and test oracle), which we argue is important in order to address the core question of our work, which is finding validation metrics that correlate well with the ground truth oracle. While Trabucco et al. (2022) does allude to validation sets in the appendix, these do not convey the same semantic meaning as our validation set since theirs is assumed to be a subsample of the training set, and therefore come from the training distibution. Lastly, while the same appendix provides examples for different validation metrics per baseline, the overall paper itself is concerned with establishing reliable benchmarks for comparison, rather than comparing validation metrics directly.

Table 1: Validation metrics considered in this work. We would like to determine which of these metrics is most correlated with the ground truth oracle, as determined by the test score (Equation 8). For $\mathcal{M}_{\text{FD}}$ and $\mathcal{M}_{\text{DC}}$, $\boldsymbol{X}_{\text{valid}}$ denotes the $\boldsymbol{x}$'s in $\mathcal{D}_{\text{valid}}$ and $\tilde{\boldsymbol{X}}$ denotes samples generated from the extrapolated model $p_{\theta,\gamma}$, and $|\tilde{\boldsymbol{X}}| = |\boldsymbol{X}_{\text{valid}}|$. Note that we measure the negative of $\mathcal{M}_{\text{DC}}$ and $\mathcal{M}_{\text{score}}$ to be consistent, i.e. for any validation metric considered we would like to *minimise* it.

Metric	Pros ✓	Cons ✗
-$\mathcal{M}_{\text{score}}(\mathcal{S}; \theta, \phi)$ (Eq. 7)	• simply the validation variant of the test score (Eqn. 8)	• relies on external classifier (approximate oracle) • does not take into account diversity of generated candidates
$\mathcal{M}_{\text{Agr}}(\mathcal{D}_{\text{valid}}; \theta)$ (Eq. 10)	• theoretical connection to reverse KL divergence (see Section A.3.2) • model agnostic (can compare any class of generative model)	• relies on external classifier (approximate oracle) • does not take into account diversity of generated candidates
$\mathcal{M}_{\text{C-DSM}}(\mathcal{D}_{\text{valid}}; \theta)$ (Eq. 15)	• same as training loss	• denoising score matching loss implements forward KL, which may have a stronger bias towards recall
$\mathcal{M}_{\text{FD}}(\boldsymbol{X}_{\text{valid}}, \tilde{\boldsymbol{X}})$ (Eq. S16)	• commonly-used and widely studied • model agnostic (can compare any class of generative model)	• bias towards recall (Sec. A.2) • relies on external feature extractor, results can widely vary between them
-$\mathcal{M}_{\text{DC}}(\boldsymbol{X}_{\text{valid}}, \tilde{\boldsymbol{X}})$ (Eq. S20)	• improved version of Kynkäänniemi et al. (2019) • model agnostic (can compare any class of generative model)	• relies on external feature extractor, results can widely vary between them

Validation metrics To be best of our knowledge, a rigorous exploration of validation metrics has not yet been explored in MBO. The choice of validation metric is indeed partly influenced by the generative model, since one can simply assign the validation metric to be the same as the training loss but evaluated on the validation set. For example, if we choose likelihood-based generative models (essentially almost all generative models apart from GANs), then we can simply evaluate the likelihood on the validation set and use that as the validation metric (Equation 5). However, it has been well established that likelihood is a relatively poor measure of sample quality and is more biased towards sample coverage (Huszár, 2015; Theis et al., 2015; Dosovitskiy & Brox, 2016). While GANs have made it difficult to evaluate likelihoods (they are non-likelihood-based generative models), it has fortunately given rise to an extensive literature proposing 'likelihood-free' evaluation metrics (Borji, 2022), and these are extremely useful to explore for this study for two reasons. Firstly, likelihood-free metrics are *agnostic* to the class of generative model used, and secondly they are able to probe various aspects of generative models that are hard to capture with just likelihood. As an example, the Fréchet Distance (FID) (Heusel et al., 2017) is commonly used to evaluate the realism of generated samples with respect to a reference distribution, and correlates well with human judgement of sample quality. Furthermore, metrics based on precision and recall can be used to quantify sample quality and sample coverage, respectively (Sajjadi et al., 2018; Kynkäänniemi et al., 2019).

In most existing works mentioned, the discussion of validation metrics and model selection are rather opaque, and instead most of the focus is on the final evaluation, i.e. the ground truth or approximate test oracle's scores on the generated candidates. However, in Trabucco et al. (2022) (their Appendix F) some examples are given as to what validation metrics could be used.

3.1 Modelling approaches

Model inversion networks Generative modelling for MBO was proposed by Kumar & Levine (2020), under the name *model inversion networks* (MINs). The name is in reference to the fact that one can learn the *inverse* of the oracle $f_\theta^{-1} : \mathcal{Y} \to \mathcal{X}$, which is a generative model. In their work, GANs are chosen for the

generative model, whose generator we will denote as $G_\theta(\boldsymbol{z}, y)$.[3] At *generation time* (i.e. after training) the authors propose the learning of the following prior distribution as a way to extrapolate the generative model[4]:

$$p_\zeta(\boldsymbol{z}, y)^* := \underset{p_\zeta(\boldsymbol{z},y)}{\arg\max}\ \mathbb{E}_{\boldsymbol{z},y\sim p_\zeta(\boldsymbol{z},y)} y + \mathbb{E}_{(\boldsymbol{z},y)\sim p_\zeta}\Big[\epsilon_1 \log p_\theta(y|G_\theta(\boldsymbol{z}, y)) + \epsilon_2 \log p(\boldsymbol{z})\Big], \tag{9}$$

where ϵ_1 and ϵ_2 are hyperparameters that weight the *agreement* and the prior probability of the candidate $\boldsymbol{z}$. The agreement is measuring the log likelihood of $\tilde{\boldsymbol{x}} = G_\theta(\boldsymbol{z}, y)$ being classified as y under the training oracle f_θ (expressed probabilistically as $p_\theta(y|\boldsymbol{x})$), and can be thought of measuring to what extent the classifier and generative model 'agree' that $\tilde{\boldsymbol{x}}$ has a score of y. The log density $p(\boldsymbol{z})$ can be thought of as a regulariser to ensure that the generated candidate $\boldsymbol{z}$ is likely under the latent distribution of the GAN.

We note that agreement can be easily turned into a validation metric by simply substituting $p_\theta(y|\boldsymbol{x})$ for the validation oracle $p_\phi(y|\boldsymbol{x})$. Note that if we assume that $p_\phi(y|\boldsymbol{x})$ is a Gaussian, then the log density of some input y turns into the mean squared error up to some constant terms, so we we can simply write agreement out as $\mathbb{E}_{y\sim p_\gamma(y)}\|f_\phi(G_\theta(\boldsymbol{z}, y)) - y\|^2$. This leads us to our second validation metric, which we formally define as:

$$\boxed{\mathcal{M}_{\text{Agr}}(\mathcal{D}_{\text{valid}}; \theta) = \frac{1}{|\mathcal{D}_{\text{valid}}|} \sum_{(\boldsymbol{x},y)\sim\mathcal{D}_{\text{valid}}} \|f_\phi(\tilde{\boldsymbol{x}}_i) - y\|^2, \quad \text{where } \tilde{\boldsymbol{x}}_i \sim p_\theta(\boldsymbol{x}|y)} \tag{10}$$

Discriminative approaches In the introduction we noted that MBO methods can be seen as approaching the problem from either a discriminative or a generative point of view (though some overlap can also certainly exist between the two). In the former case, regularising the approximate oracle $f_\theta(\boldsymbol{x})$ is key, and it is also the model that is sampled from (e.g. as per gradient ascent in Equation 1). The key idea is that the approximate oracle should act conservatively or pessimistically in out-of-distribution regions. Some examples include mining for and penalising adversarial examples (Trabucco et al., 2021), encouraging model smoothness Yu et al. (2021), conservative statistical estimators such as normalised maximum likelihood (Fu & Levine, 2021), learning bidirectional mappings (Chen et al., 2022), and mitigating domain shift (Qi et al., 2022).

Generative approaches Brookes et al. (2019) propose the use of variational inference to learn a sampling distribution $p_\theta(\boldsymbol{x}|S)$ given a probabilistic form of the oracle $p_\theta(S|\boldsymbol{x})$ as well as a pre-trained prior distribution over the data $p_\theta(\boldsymbol{x})$. Here, S is some desirable range of y's, and therefore $p_\theta(\boldsymbol{x}|S)$ can be thought of as the 'extrapolated' generative model. Lastly, Fannjiang & Listgarten (2020) proposes MBO training within the context of a min-max game between the generative model and approximate oracle. Given some target range $y \in S$ an iterative min-max game is performed where the generative model $p_\theta(\boldsymbol{x})$ updates its parameters to maximise the expected conditional probability over samples generated from that range, and the approximate oracle $f_\theta(\boldsymbol{x})$ updates its parameters to minimise the error between the generated predictions and that of the ground truth. Since the latter isn't accessible, an approximation of the error is used instead. In relation to our evaluation framework, the extrapolated model would essentially be the final set of weights $\theta^{(t)}$ for $p_\theta(\boldsymbol{x}|S)|_{\theta=\theta^{(t)}}$ when the min-max game has reached equilibrium.

For both approaches, there is a notion of leveraging an initial generative model $p_\theta(\boldsymbol{x})$ and fine-tuning it with the oracle so that it generates higher-scoring samples in regions that it was not initially trained on. Both the min-max and variational inference techniques can be thought of as creating the 'extrapolated' model within the context of our evaluation framework (Figure 2). Therefore, while our evaluation framework does not preclude these more sophisticated techniques, we have chosen to use the simplest extrapolation technique possible – which is simply switching out the prior distribution – as explained in Section 2.1.

Diffusion models Recently, diffusion models (Ho et al., 2020) have attracted significant interest due to their competitive performance and ease of training. They are also very closely related to score-based

[3]Note that we can still employ the same sampling notation here, i.e. $\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|y)$ implies we sample from the prior $\boldsymbol{z} \sim p(\boldsymbol{z})$, then produce a sample $\boldsymbol{x} = G_\theta(\boldsymbol{z}, y)$.

[4]In Kumar & Levine (2020) this optimisation is expressed for a single $(\boldsymbol{z}, y)$ pair, but here we formalise it as learning a joint distribution over these two variables. If this optimisation is expressed for a minibatch of $(\boldsymbol{z}, y)$'s, then it can be seen as learning an empirical distribution over those variables.

generative models (Song & Ermon, 2019; 2020). In diffusion, the task is to learn a neural network that can denoise any $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t-1}$, where $q(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T)$ defines a joint distribution over increasingly perturbed versions of the real data $q(\boldsymbol{x}_0)$. Assuming that $q(\boldsymbol{x}_T) \approx p(\boldsymbol{x}_T)$ for some prior distribution over $\boldsymbol{x}_T$, to generate a sample Langevin MCMC is used to progressively denoise a prior sample $\boldsymbol{x}_T$ into $\boldsymbol{x}_0$, and the result is a sample from the distribution $p_\theta(\boldsymbol{x}_0)$.[5] To the best of our knowledge, we are not aware of any existing works that evaluate diffusion or score-based generative models on MBO datasets provided by Design Bench, and therefore we consider our exploration into diffusion models here as an orthogonal contribution.

4 Experiments and Discussion

In this section we work towards answering the following question: which validation metrics correlate the best with the ground truth oracle? So far two have been defined:

- The 'score metric' $\mathcal{M}_{\text{score}}$, which is just using the approximate validation oracle to score candidates generated by the model;
- the 'agreement' $\mathcal{M}_{\text{Agr}}$ (Section 3, Paragraph 3.1), which is an equation originally proposed in Kumar & Levine (2020) but re-purposed for model selection.

Furthermore, since we demonstrate our validation framework using conditional denoising diffusion probabilistic models (conditional DDPMs), one metric is essentially the DDPM training loss but evaluated on the validation set, which we call $\mathcal{M}_{\text{C-DSM}}$ (Equation 15), which we explain in Paragraph 4. The last two validation metrics – Frechet distance and density/coverage – are inspired from the GAN literature and measure distances between the truncated ground truth and extrapolated distribution. These are also useful since they capture notions of precison and recall, which are both very important in generative modelling. In the interest of space, we defer their explanations to appendix Sections A.2.

Dataset Our codebase is built on top of the Design Bench (Trabucco et al., 2022) framework. We consider all continuous datasets in Design Bench datasets: Ant Morphology, D'Kitty Morphology, Superconductor, and Hopper. Continuous datasets are chosen since we are using Gaussian denoising diffusion models, though discrete variants also exist and we leave this to future work.

Both morphology datasets are ones in which the morphology of a robot must be optimised in order to maximise a reward function. For these datasets, the ground truth oracle is a morphology-conditioned controller. For Superconductor, the ground truth oracle is not accessible and therefore the test oracle is also approximate (which we simply call the 'approximate test oracle'). For Hopper, the goal is to sample a large (≈ 5000 dimensional) set of weights which are used to parameterise a controller.

Data splits While our framework is built on top of Design Bench, as mentioned in Section 3 the evaluation differs slightly. In Design Bench, only $\mathcal{D}$ and $\mathcal{D}_{\text{train}}$ are exposed, and any users intending to perform validation should derive $\mathcal{D}_{\text{valid}}$ from $\mathcal{D}_{\text{train}}$. Because we require our validation set to be a higher-scoring distribution than the training set, we break this convention and define the validation split to be $\mathcal{D}_{\text{train}} \setminus \mathcal{D}$, i.e. their set difference. Note that *if* a ground truth oracle exists, there is no need to define a $\mathcal{D}_{\text{test}}$. Otherwise, a random 50% subsample of $(\mathcal{D}_{\text{train}} \setminus \mathcal{D})$ is assigned to $\mathcal{D}_{\text{valid}}$ and the other 50% assigned to $\mathcal{D}_{\text{test}}$. For Ant, Kitty, and Hopper, the test oracles are exact. For Superconductor, we use the pre-trained test oracle `RandomForest-v0` provided with the framework (which was trained on the full dataset $\mathcal{D}$).

One nuance with the Hopper dataset is that the full dataset $\mathcal{D}$ and the training set $\mathcal{D}_{\text{train}}$ are equivalent, presumably because of the scarcity of examples. This means that a validation set cannot be extracted unless the training set itself is redefined, and this means that the training set in our framework is no longer identical to that originally proposed by Design Bench. To address this, we compute the median y with respect to $\mathcal{D}_{\text{train}}$, and take the lower half as $\mathcal{D}_{\text{train}}$ and the upper half as $\mathcal{D}_{\text{valid}}$. We call the final dataset 'Hopper 50%', to distinguish it from the original dataset.

[5]The Langevin MCMC procedure is theoretically guaranteed to produce a sample from $p_\theta(\boldsymbol{x})$ (Welling & Teh, 2011). As opposed to Equation 1, where no noise is injected into the procedure and therefore samples are mode seeking.

Oracle pre-training The validation oracle f_ϕ is an MLP comprising of four hidden layer, trained on $\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{valid}}$. Note that in the case of a dataset not having a ground truth oracle f, this is not to be confused with the *approximate test oracle*, which is trained on $\mathcal{D} = \mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{valid}} \cap \mathcal{D}_{\text{test}}$.

Architecture The architecture that we use is a convolutional U-Net from HuggingFace's 'annotated diffusion model' [6], whose convolutional operators have been replaced with fully connected layers (since Ant and Kitty morphology inputs are flat vectors). For Hopper, we use 1D convolutions since MLPs performed poorly and significantly blew up the number of learnable parameters.

For all experiments we train with the ADAM optimiser (Kingma & Ba, 2014), with a learning rate of 2×10^{-5}, $\beta = (0.0, 0.9)$, and diffusion timesteps $T = 200$. Experiments are trained for 5000 epochs with single P-100 GPUs. Input data is normalised with the min and max values per feature, with the min and max values computed over the training set $\mathcal{D}_{\text{train}}$. The same is computed for the score variable y, i.e. all examples in the training set have a normalised score $y \in [0, 1]$ and those in the validation set have scores $y > 1$.

Training We consider the denoising diffusion probabilistic model (DDPM) proposed by Ho et al. (2020). DDPMs are currently state-of-the-art in many tasks and avoid some of the issues associated with other classes of generative model. Like VAEs, DDPMs maximise an evidence lower bound, which in turn approximates the forward KL divergence between the data and generative distributions. Using typical DDPM notation, $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$ is the real data, and $q(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ for $t \in \{1, \ldots, T\}$ defines progressively noisier distributions, and p_θ parameterises a neural net reverse this process. In practice, when such distributions are Gaussian, each of the T KL terms in the ELBO can be simplified to the following noise prediction task, where we train a neural network $\epsilon_\theta(\boldsymbol{x}_t, t)$ to predict the noise from $\boldsymbol{x}_t \sim q(\boldsymbol{x}_t|\boldsymbol{x}_0)$:

$$\mathcal{L}_{\text{DSM}}(\theta) = \mathbb{E}_{\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0), \boldsymbol{x}_t \sim q(\boldsymbol{x}_t|\boldsymbol{x}_0), \epsilon_t} \left[\|\epsilon_t - \epsilon_\theta(\boldsymbol{x}_t, t)\|^2 \right], \tag{11}$$

where ϵ_t is the noise used to produce $\boldsymbol{x}_t$ via $\boldsymbol{x}_t := \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon_t$, $\epsilon_t \sim \mathcal{N}(0, 1)$, and $\{\alpha_t\}_{t=1}^T$ defines a noising schedule. Lastly, sampling $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$ is approximated with those from $\mathcal{D}_{\text{train}}$.

Conditioning Note that Equation 11 defines an *unconditional* model $p_\theta(\boldsymbol{x})$. In order to be able to condition on the score y, we can consider two options.[7] The first is *classifier-based guidance* (Dhariwal & Nichol, 2021). Here, we train an *unconditional* diffusion model $p_\theta(\boldsymbol{x})$, but during *validation* we leverage the validation oracle via equation:

$$\epsilon_\theta(\boldsymbol{x}_t, t, y) = \underbrace{\epsilon_\theta(\boldsymbol{x}_t, t) - \sqrt{1 - \bar{\alpha}_t} w \nabla_{\boldsymbol{x}_t} \log p_\phi(y|\boldsymbol{x}_t; t)}_{\text{classifier-based guidance}} \tag{12}$$

where $w \in \mathbb{R}^+$ is a hyperparameter. Note that $p_\phi(y|\boldsymbol{x}_t; t)$ – with the extra conditioning on t – is not the same as the original validation set oracle. The former is trained to also predict y from any noisy distribution $q(\boldsymbol{x}_t)$. The reason for this is because using the original validation oracle $p_\phi(y|\boldsymbol{x})$ in Equation 12 may contribute bad gradient for noisy $\boldsymbol{x}$'s that are too far from the training distribution.

In *classifier-free guidance* (Ho & Salimans, 2022), the noise predictor is trained both conditionally and unconditionally using label dropout. That is, ϵ_θ can also be conditioned on y as well:

$$\mathcal{L}_{\text{C-DSM}}(\theta; \tau) = \mathbb{E}_{(\boldsymbol{x}_0, y) \sim \mathcal{D}_{\text{train}}, \boldsymbol{x}_t \sim q(\boldsymbol{x}_t|\boldsymbol{x}_0), \lambda \sim U(0,1)} \|\epsilon_t - \epsilon_\theta(\boldsymbol{x}_t, \mathbf{1}_{\lambda < \tau}(y), t)\|, \tag{13}$$

where $\mathbf{1}_{\lambda < \tau}(y)$ is the indicator function and returns a null token if $\lambda < \tau$ otherwise y is returned. The additional significance of this is that at generation time, one can choose various tradeoffs of (conditional) sample quality and diversity by using the following noise predictor, for some hyperparameter w:

$$\bar{\epsilon_\theta}(\boldsymbol{x}_t, t, y) = \underbrace{(w + 1)\epsilon_\theta(\boldsymbol{x}_t, t, y) - w\epsilon_\theta(\boldsymbol{x}, t)}_{\text{classifier-free guidance}} \tag{14}$$

[6] `https://huggingface.co/blog/annotated-diffusion`

[7] While it is possible to derive a conditional ELBO from which a DDPM can be derived from, the most popular method for conditioning in DDPMs appears to be via guiding an unconditional model instead.

Validation The last validation metric we propose is specific to experiments run with classifier-free guidance. We use the *conditional* noise predictor in Equation 14 to construct the conditional form of Equation 11:

$$\boxed{\mathcal{M}_{\text{C-DSM}}(\mathcal{D}_{\text{valid}}; \theta) = \frac{1}{\mathcal{D}_{\text{valid}}} \sum_{(\boldsymbol{x},y)\sim\mathcal{D}_{\text{valid}}} \|\epsilon_t - \epsilon_\theta(\boldsymbol{x}_t, y, t)\|^2} \tag{15}$$

For each validation metric and dataset, we run many experiments where each experiment is a particular instantiation of hyperparameters (see A.4.1). During the course of training we periodically keep track of the smallest value observed so far for that particular validation metric and save its model weights as a checkpoint (which is essentially early stopping). For each experiment, we load the early-stopped checkpoint, generate candidates and compute the final test score as per Equation 8. Therefore, if we have run m experiments then we generate an m-length list of tuples, where the first tuple element is the value of that particular validation metric and the second tuple element is the actual test score. From this m list the Pearson correlation is computed.

4.1 Results

Classifier-free guidance In Figure 3 we plot the Pearson correlations achieved for each dataset as well as each diffusion variant, classifier-free guidance ('cfg') and classifier-based ('cg'). (Due to space constraints, we have also plotted the Pearson correlations on a per-dataset and per-validation metric level and these are shown in the appendix.) Since all validation metrics are intended to be minimised, we are interested in metrics that correlate the *most negatively* with the final test score, i.e. the smaller some validation metric is, the larger the average test score as per Equation 8. We first consider the first four columns of the barplot, which correspond to just the classifier-free guidance experiments for all four datasets. The two metrics which perform consistency well are $\mathcal{M}_{\text{score}}$ and $\mathcal{M}_{\text{Agr}}$. However, $\mathcal{M}_{\text{FD}}$ also appears to perform well for Kitty and $\mathcal{M}_{\text{C-DSM}}$ for Superconductor.

Classifier-based guidance For c.g. the differences between metrics are much less pronounced, and some care must be taken in the interpretation of these results. Since classifier-based guidance requires a classifier to be given to the diffusion model during *generation* (*not* training), we chose to use the validation oracle f_ϕ (Equation 12), which means that the diffusion model gets to exploit gradients derived from a dataset of higher scoring designs. This was not done for the purposes of obtaining advantageous results, but rather because our evaluation framework technically permits it, since the validation set (or artifacts derived from it) can be used as long as it's not in the training stage (Figure 2a). This however confounds our analysis, since it isn't clear how well the diffusion model can extrapolate in the absence of such gradients. While we could just simply replace f_ϕ with a 'training oracle' which has only been trained on the training set, we can simply interpret the results in light of this information. At the same time, we believe the reason for the metrics all performing similarly here is because the differences between validation metrics are less important when the generative model gets to 'make up' for any defiencies by taking gradients with respect to an oracle that has been trained on larger scoring designs.[8]

Sample quality vs diversity We find that the guidance hyperparameter w makes a huge difference to the final results, and this is shown in Figure 14. Each point is colour-coded by w, which specifies the strength of the 'implicit' classifier that is derived (Equation 14). With respect to just the two most strongly correlated metrics per dataset (as per Figure 3), larger w's appear to be more favourable for Ant Morphology ($\mathcal{M}_{\text{score}}$ and $\mathcal{M}_{\text{Agr}}$). For Kitty Morphology, $w \approx 1$ is best for $\mathcal{M}_{\text{score}}$ and a $w \approx 0$ is best for $\mathcal{M}_{\text{FD}}$. This indicates that performance is sensitive to w, and this appears to underscore a well-established 'dilemma' in generative modelling, which is the trade-off between *sample quality* and *sample diversity* (Ho & Salimans, 2022; Brock et al., 2018; Burgess et al., 2018; Dhariwal & Nichol, 2021). While the ideal case would be to satisfy both properties equally, in practice some trade-offs have to be made. If sample quality is too heavily weighted, then sample diversity suffers and as a result the final set of (supposedly high scoring) generated candidates

[8]This analogy becomes much clearer if we consider the extreme case: if the validation oracle *was* the same as the ground truth f, we would expect a perfect correlation of -1.

may actually be classed as a badly scoring mode by the ground truth oracle. Conversely, if sample diversity is too heavily weighted then we miss out on high-scoring modes of the data.

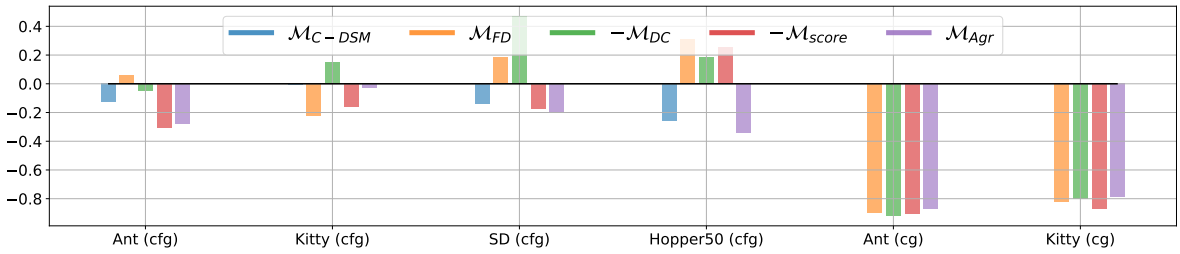
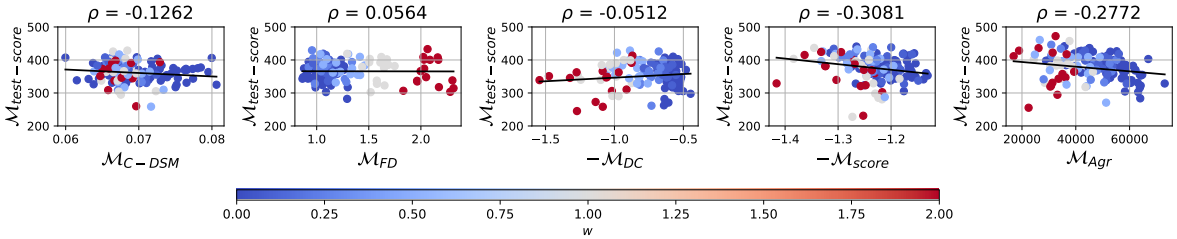
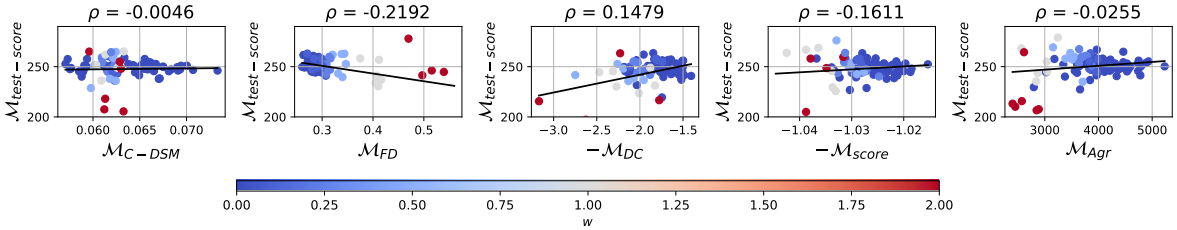


Figure 3: The Pearson correlation computed for each dataset / diffusion variant. Pearson correlations are computed as per the description in Paragraph 4. Since each validation metric is something that should be minimised, the ideal metric should be highly negatively correlated with the test score (Equation 8), which is to be maximised.



(a) Ant Morphology, classifier-free guidance (c.f.g) The Pearson correlations shown above each subplot correspond to those shown for the same corresponding experiment in Figure 3. Here, we can see that $\mathcal{M}_{\text{score}}$ and $\mathcal{M}_{\text{Agr}}$ are the most negatively correlated with the test score. Points are colour-coded by guidance parameter w (see Equation 14).



(b) Kitty Morphology (c.f.g.)

Figure 4: Correlation plots for the Ant and Kitty Morphology, for the classifier-free guidance (c.f.g.) diffusion variant. Each point is colour-coded by w, which specifies the strength of the 'implicit' classifier that is derived (Equation 14). We can see that w makes a discernible difference with respect to most of the plots shown. With respect to just the two most strongly correlated metrics per dataset: larger w's appear to be more favourable for Ant Morphology ($\mathcal{M}_{\text{score}}$ and $\mathcal{M}_{\text{Agr}}$); for Kitty Morphology, $w \approx 1$ is best for $\mathcal{M}_{\text{score}}$ and a $w \approx 0$ is best for $\mathcal{M}_{\text{FD}}$. For additional plots for other datasets, please see Section A.5.

In Table 2 we present the final test scores for both diffusion variants and both datasets, as well as provide Design Bench's results as reference. Since we cannot compare our version of Hopper's results to any existing benchmarks, we omit them from this table (though its correlation plots are shown in Section A.5.4).

5 Conclusion

In this work, we asked a fundamental question pertaining to evaluation in offline MBO: what validation metrics correlate well with the ground truth oracle? The key idea is that if we can run our presented study at

	Ant Morphology	D'Kitty Morphology	Superconductor
$\mathcal{D}_{\text{train}}$	0.565	0.884	0.400
Auto. CbAS	0.882 ± 0.045	0.906 ± 0.006	0.421 ± 0.045
CbAS	0.876 ± 0.031	0.892 ± 0.008	0.503 ± 0.069
BO-qEI	0.819 ± 0.000	0.896 ± 0.000	0.402 ± 0.034
CMA-ES	**1.214 ± 0.732**	0.724 ± 0.001	0.465 ± 0.024
Grad.	0.293 ± 0.023	0.874 ± 0.022	**0.518 ± 0.024**
Grad. Min	0.479 ± 0.064	0.889 ± 0.011	0.506 ± 0.009
Grad. Mean	0.445 ± 0.080	0.892 ± 0.011	0.499 ± 0.017
REINFORCE	0.266 ± 0.032	0.562 ± 0.196	0.481 ± 0.013
MINs	0.445 ± 0.080	0.892 ± 0.011	0.499 ± 0.017
COMs	0.944 ± 0.016	**0.949 ± 0.015**	0.439 ± 0.033
Cond. Diffusion (c.f.g.)	0.954 ± 0.025	0.972 ± 0.006	0.645 ± 0.115
Cond. Diffusion (c.g.)	0.975 ± 0.016	0.974 ± 0.007	–

Table 2: 100th percentile test scores for methods from Design Bench (Trabucco et al., 2022) as well as our diffusion results shown in the last two rows, with c.f.g standing for *classifier-free guidance* (Equation 14) and c.g. standing for *classifier-guidance* (Equation 12). Each result is an average computed over six different runs (seeds). Test scores are min-max normalised with respect to the smallest and largest oracle scores in the *full* dataset, i.e. any scores greater than 1 are *greater* than any score observed in the full dataset. Design Bench results are shown for illustrative purposes only – while our training sets are equivalent to theirs, we use a held-out validation set to guide model selection, which makes a direct comparison difficult.

scale for a both a difficult and diverse range of datasets for which the ground truth is *known*, insights derived from those findings (such as what are robust validation metrics) can be transferred to more real-world offline MBO tasks where the actual ground truth oracle is expensive to evaluate. To approach this, our evaluation framework is designed to measure how well a generative model extrapolates: the training and validation sets are seen as coming from different γ-*truncated* distributions, where examples in the validation set comprise a range of y's that are not covered by the training set and are *larger* than those in the training set. Therefore, from the point of view of the generative model, the validation set is out-of-distribution. Because model selection involves measuring some notion of desirability on the validation set (via a validation metric), we are effectively trying to select for models that can *extrapolate*.

We demonstrated our framework on four continuous datasets prescribed by Design Bench, as well as across five different validation metrics and two forms of label conditioning for Gaussian diffusion models: classifier-free and classifier-based guidance. The five validation metrics we chose were inspired by existing MBO works as well as the GAN literature, with the latter having proposed many metrics that are agnostic to the class of generative model. For the classifier-free conditional variant, we identified validation metrics that performed consistently well, namely the score and agreement metrics. We do not claim that these metrics would be useful across different generative models however – ideally, the same experiments should be run on other model variants such as GANs and VAEs.[9] Lastly, we identified that balancing the trade-off between sample quality and sample diversity is extremely important for accurately generating extrapolated examples. Fortunately, for both diffusion variants controlling this trade-off is as simple as tuning a single hyperparameter.

For future work, a wider variety of datasets should be considered such as non-robotics datasets as well as discrete ones (which for instance can be addressed with discrete diffusion variants). Since the ground truth oracle is required for a precise evaluation, this may call for the proposal of new datasets. Finally, following an official release of the code we hope that the work that is presented will encourage further exploration into good validation metrics and practices for model-based optimisation.

References

Asmussen, Søren and Glynn, Peter W. *Stochastic simulation: algorithms and analysis*, volume 57. Springer, 2007.

[9]For instance, while $\mathcal{M}_{\text{score}}$ performed quite well, we would not expect this to be the case for discriminative model-based approaches, since there is a high risk of generating adversarial or overscored examples.

Borji, Ali. Pros and cons of GAN evaluation measures: New developments. *Computer Vision and Image Understanding*, 215:103329, 2022.

Brock, Andrew, Donahue, Jeff, and Simonyan, Karen. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

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

Brookes, David, Park, Hahnbeom, and Listgarten, Jennifer. Conditioning by adaptive sampling for robust design. In *International conference on machine learning*, pp. 773–782. PMLR, 2019.

Burgess, Christopher P, Higgins, Irina, Pal, Arka, Matthey, Loic, Watters, Nick, Desjardins, Guillaume, and Lerchner, Alexander. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Chen, Can, Zhang, Yingxue, Fu, Jie, Liu, Xue, and Coates, Mark. Bidirectional learning for offline infinite-width model-based optimization. *ArXiv*, abs/2209.07507, 2022.

Dhariwal, Prafulla and Nichol, Alexander. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Dosovitskiy, Alexey and Brox, Thomas. Generating images with perceptual similarity metrics based on deep networks. *Advances in neural information processing systems*, 29, 2016.

Dowson, DC and Landau, BV666017. The Fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, 12(3):450–455, 1982.

Fannjiang, Clara and Listgarten, Jennifer. Autofocused oracles for model-based design. *Advances in Neural Information Processing Systems*, 33:12945–12956, 2020.

Fu, Justin and Levine, Sergey. Offline model-based optimization via normalized maximum likelihood estimation. *arXiv preprint arXiv:2102.07970*, 2021.

Hamidieh, Kam. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354, 2018.

Heusel, Martin, Ramsauer, Hubert, Unterthiner, Thomas, Nessler, Bernhard, Klambauer, Günter, and Hochreiter, Sepp. GANs trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. URL `http://arxiv.org/abs/1706.08500`.

Ho, Jonathan and Salimans, Tim. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

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

Huszár, Ferenc. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kumar, Aviral and Levine, Sergey. Model inversion networks for model-based optimization. *Advances in Neural Information Processing Systems*, 33:5126–5137, 2020.

Kynkäänniemi, Tuomas, Karras, Tero, Laine, Samuli, Lehtinen, Jaakko, and Aila, Timo. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

O'Donoghue, Brendan, Osband, Ian, and Ionescu, Catalin. Making sense of reinforcement learning and probabilistic inference. *arXiv preprint arXiv:2001.00805*, 2020.

Piche, Alexandre, Pardinas, Rafael, Vazquez, David, Mordatch, Igor, and Pal, Chris. Implicit offline reinforcement learning via supervised learning. *arXiv preprint arXiv:2210.12272*, 2022.

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

Sajjadi, Mehdi SM, Bachem, Olivier, Lucic, Mario, Bousquet, Olivier, and Gelly, Sylvain. Assessing generative models via precision and recall. *Advances in neural information processing systems*, 31, 2018.

Song, Yang and Ermon, Stefano. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Song, Yang and Ermon, Stefano. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

Theis, Lucas, Oord, Aäron van den, and Bethge, Matthias. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Todorov, Emanuel, Erez, Tom, and Tassa, Yuval. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Trabucco, Brandon, Kumar, Aviral, Geng, Xinyang, and Levine, Sergey. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pp. 10358–10368. PMLR, 2021.

Trabucco, Brandon, Geng, Xinyang, Kumar, Aviral, and Levine, Sergey. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*, pp. 21658–21676. PMLR, 2022.

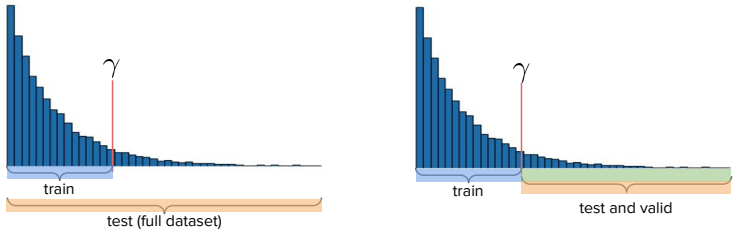
Welling, Max and Teh, Yee W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Weng, Lilian. What are diffusion models? *lilianweng.github.io*, Jul 2021. URL `https://lilianweng.github.io/posts/2021-07-11-diffusion-models/`.

Yu, Sihyun, Ahn, Sungsoo, Song, Le, and Shin, Jinwoo. Roma: Robust model adaptation for offline model-based optimization. *Advances in Neural Information Processing Systems*, 34:4619–4631, 2021.

A Appendix

A.1 Training, validation, and test splits



(a) Design Bench (b) Ours (on top of Design Bench)

Figure S5: Design Bench only prescribes a training split. The full dataset (which we simply call the test set here) is also accessible but is not meant to be accessed during training. If the dataset does not contain a ground truth oracle, the test set is what is used to train the final (test) oracle. Furthermore, the lack of a prescribed validation set makes it difficult for us to employ our conceptual evaluation framework detailed in Figure 2. To address this, we retain the training set as shown in S5a, but denote everything else (examples whose scores are $> \gamma$) to comprise either validation or testing examples (S5b). To distinguish between the two, we can simply randomly shuffle these examples and denote the first half as validation and the second half as test. However, if there exists a ground truth oracle for the dataset, there is no need for a separate test set, and both halves will be for validation.

A.2 Validation metrics

Frechet Distance 'Likelihood-free' metrics are used almost exclusively in the GAN literature because there is no straightforward way to compute likelihoods for this class of models, i.e. $p_\theta(\boldsymbol{x}|y)$ cannot be evaluated. Furthermore, the search for good metrics is still an active topic of research (Borji, 2022). Common likelihood-free metrics involve measuring some distance between distributions in some predefined feature space. For instance, for GANs trained on natural image datasets the *Fréchet Inception Distance* (FID) (Heusel et al., 2017) is used to fit Gaussians to both distributions with respect to the feature space of an InceptionNet classifier trained on ImageNet. Since the acronym 'FID' specifically refers to a very specific InceptionNet-based model, we will simply call it 'FD'. If we assume that FD is computed in some latent space characterised by an arbitrary feature extractor $f_h : \mathcal{X} \to \mathcal{H}$, then FD can be computed in closed form as follows (Dowson & Landau, 1982):

$$\boxed{\mathcal{M}_{\text{FD}}(\boldsymbol{X}, \tilde{\boldsymbol{X}}; f_h) = |\mu(f_h(\boldsymbol{X})) - \mu(f_h(\tilde{\boldsymbol{X}}))| \ + \text{Tr}(\Sigma(f_h(\boldsymbol{X})) + \Sigma(f_h(\tilde{\boldsymbol{X}})) - 2\Sigma(f_h(\tilde{\boldsymbol{X}}))\Sigma(f_h(\tilde{\boldsymbol{X}}))^{\frac{1}{2}})} \tag{S16a}$$

where $\mathcal{M}_{\text{FD}} \in \mathbb{R}^+$ and lower FD is better. FD is also known as the 2-Wasserstein distance. Here, $\boldsymbol{X} \in \mathbb{R}^{N \times p}$ denotes N samples coming from a reference distribution (i.e. ground truth) and $\tilde{\boldsymbol{X}}$ are samples coming from the generative model. $\boldsymbol{H} = f_h(\boldsymbol{X})$ denotes these inputs mapped to some feature space. Conveniently, we can simply define the feature space to be with respect to some hidden layer of the *validation oracle*. One caveat of FD is that it may have a stronger bias towards recall (mode coverage) than precision (sample quality) (Kynkäänniemi et al., 2019) and that it reports a single number, which makes it difficult to tease apart how well the model contributes to precision and recall. Furthermore, while there exists a canonical network architecture and set of weights to use for evaluating generative models on natural image datasets (i.e. a particular Inception-V3 network that gives rise to the Frechet *Inception* Distance), this is not the case for other types of datasets. This means that, unless a particular feature extractor is agreed upon, comparing results between papers is non-trivial.

Density and coverage We also consider 'density and coverage', which corresponds to an improved version of the 'precision and recall' metric proposed in Kynkäänniemi et al. (2019). In essence, these methods estimate the manifold of both the real and fake data distributions in latent space via the aggregation of hyperspheres centered on each point, and these are used to define precision and recall: precision is the *proportion of fake data that can be explained by real data* (in latent space), and recall is the *proportion of real data that can be explained by fake data* (again, in latent space).

Similar to FD (Paragraph A.2), let us denote $\boldsymbol{H}_i$ as the example $\boldsymbol{X}_i$ embedded in latent space. Let us also define $B(\boldsymbol{H}_i, \mathrm{NND}_K(\boldsymbol{H}_i)$ as the hypersphere centered on $\boldsymbol{H}_i$ whose radius is the k-nearest neighbour, and k is a user-specified parameter. 'Density' (the improved precision metric) is defined as:

$$\mathcal{M}_{\text{density}}(\boldsymbol{H}, \tilde{\boldsymbol{H}}; k) = \frac{1}{kN} \sum_{j=1}^{M} \underbrace{\sum_{i=1}^{N} \mathbf{1}\Big\{\tilde{\boldsymbol{H}}_j \in B(\boldsymbol{H}_i, \mathrm{NND}_k(\boldsymbol{H}_i))\Big\}}_{\substack{\text{how many real neighbourhoods} \\ \text{does fake sample } \tilde{\boldsymbol{x}}_j \text{ belong to?}}}, \tag{S17}$$

where $\mathbf{1}(\cdot)$ is the indicator function, and large values corresponds to a better density. While coverage (improved 'recall') can be similarly defined by switching around the real and fake terms like so, the authors choose to still leverage a manifold around real samples due to the concern of potentially too many outliers in the generated distribution $\tilde{\boldsymbol{H}}$. As a result, their coverage is defined as:

$$\mathcal{M}_{\text{coverage}}(\boldsymbol{H}, \tilde{\boldsymbol{H}}; k) = \frac{1}{N} \sum_{i=1}^{N} \bigcup_{j=1}^{M} \mathbf{1}\Big\{\tilde{\boldsymbol{H}}_j \in B(\boldsymbol{H}_i, \mathrm{NND}_k(\boldsymbol{H}_i))\Big\} \tag{S18}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \underbrace{\mathbf{1}\Big\{\exists j \text{ s.t. } \tilde{\boldsymbol{H}}_j \in B(\boldsymbol{H}_i, \mathrm{NND}_k(\boldsymbol{H}_i))\Big\}}_{\substack{\text{is there } \textit{any} \text{ fake sample belonging} \\ \text{to } \boldsymbol{x}_i\text{'s neighbourhood?}}}, \tag{S19}$$

where, again, a larger value corresponds to a better coverage. This leads us to the addition of both metrics, $\mathcal{M}_{\text{DC}}$, which is simply:

$$\boxed{\mathcal{M}_{\text{DC}}(\boldsymbol{X}, \tilde{\boldsymbol{X}}; f_h, k) = \mathcal{M}_{\text{density}}(f_h(\boldsymbol{X}), f_h(\tilde{\boldsymbol{X}}); k) + \mathcal{M}_{\text{coverage}}(f_h(\boldsymbol{X}), f_h(\tilde{\boldsymbol{X}}); k)} \tag{S20a}$$

Similar to FD, we use the validation oracle f_θ to project samples into the latent space. We do not tune k and simply leave it to $k = 3$, which is a recommended default.

A.3 Related work

A.3.1 Conditioning by adaptive sampling

CbAS (Brookes et al., 2019), like our proposed method, approaches MBO from a generative modelling perspective. Given some pre-trained 'prior' generative model on the input data $p_\theta(\boldsymbol{x})$, the authors propose the derivation of the conditional generative model $p_\theta(\boldsymbol{x}|y)$ via Bayes' rule:

$$p_\theta(\boldsymbol{x}|y) = \frac{p(y|\boldsymbol{x})p_\theta(\boldsymbol{x})}{p_\theta(y)} = \frac{p(y|\boldsymbol{x})p_\theta(\boldsymbol{x})}{\int_{\boldsymbol{x}} p(y|\boldsymbol{x})p_\theta(\boldsymbol{x})d\boldsymbol{x}}, \tag{S21}$$

where $p(y|\boldsymbol{x})$ denotes the oracle in probabilistic form, and is not required to be differentiable. More generally, the authors use S to denote some target range of y's that would be desirable to condition on, for instance if $p(S|\boldsymbol{x}) = \int_y p(y|\boldsymbol{x})\mathbf{1}_{y \in S} dy$ then:

$$p_\theta(\boldsymbol{x}|S) = \frac{p(S|\boldsymbol{x})p_\theta(\boldsymbol{x})}{p_\theta(S)} = \frac{p(S|\boldsymbol{x})p_\theta(\boldsymbol{x})}{\int_{\boldsymbol{x}} p(S|\boldsymbol{x})p_\theta(\boldsymbol{x})d\boldsymbol{x}}, \tag{S22}$$

Due to the intractability of the denominator term, the authors propose the use of variational inference to learn a sampling network $q_\zeta(\boldsymbol{x})$ that is as close as possible to $p_\theta(\boldsymbol{x}|S)$ as measured by the forward KL divergence. Here, let us use $p_\theta(S|\boldsymbol{x})$ in place of $p(S|\boldsymbol{x})$, and assume the oracle $p_\theta(S|\boldsymbol{x})$ was trained on $\mathcal{D}_{\text{train}}$:[10]:

$$
\begin{aligned}
\zeta^* &= \arg\min_\zeta \underbrace{\text{KL}\Big[p_\theta(\boldsymbol{x}|S) \parallel q_\zeta(\boldsymbol{x})\Big]}_{\text{forward KL}} \\
&= \arg\min_\zeta \sum_{\boldsymbol{x}} p_\theta(\boldsymbol{x}|S) \log \Big(p_\theta(\boldsymbol{x}|S) - q_\zeta(\boldsymbol{x})\Big) \\
&= \arg\min_\zeta \sum_{\boldsymbol{x}} \Big[p_\theta(\boldsymbol{x}|S) \log p_\theta(\boldsymbol{x}|S) - p_\theta(\boldsymbol{x}|S) \log q_\zeta(\boldsymbol{x})\Big] \\
&= \arg\max_\zeta \ \mathbb{H}[p_\theta(\boldsymbol{x}|S)] + \sum_{\boldsymbol{x}} \Big[\frac{p_\theta(S|\boldsymbol{x})p_\theta(\boldsymbol{x})}{p_\theta(S)} \log q_\zeta(\boldsymbol{x})\Big] \\
&= \arg\max_\zeta \ \underbrace{\mathbb{H}[p_\theta(\boldsymbol{x}|S)]}_{\text{const.}} + \underbrace{\frac{1}{p_\theta(S)}}_{\text{const.}} \sum_{\boldsymbol{x}} \Big[p_\theta(S|\boldsymbol{x})p_\theta(\boldsymbol{x}) \log q_\zeta(\boldsymbol{x})\Big] \\
&= \arg\max_\zeta \ \mathbb{E}_{\boldsymbol{x}\sim p_\theta(\boldsymbol{x})}\Big[\underbrace{p_\theta(S|\boldsymbol{x})}_{\text{oracle}} \log q_\zeta(\boldsymbol{x})\Big].
\end{aligned}
\tag{S23}
$$

The authors mention that in practice importance sampling must be used for Equation S23. This is because the expectation is over samples in $p_\theta(\boldsymbol{x})$, which in turn was trained on only examples with (relatively) small y. i.e. those in $\mathcal{D}_{\text{train}}$. Because of this, $p(S|\boldsymbol{x})$ is likely to be small in magnitude for most samples. For more details, we defer the reader to the original paper (Brookes et al., 2019).

To relate the training of CbAS to our evaluation framework, we can instead consider Equation S23 as part of the *validation* part of our evaluation framework. In other words, if we define $S := [\gamma, \infty]$ and use the validation oracle p_ϕ in place of $p_\theta(S|\boldsymbol{x})$, then we can optimise for the *extrapolated model* as the following:

$$
\zeta^* = \arg\min_\zeta \ \mathbb{E}_{\boldsymbol{x}\sim p_\theta(\boldsymbol{x})}\Big[\underbrace{p_\phi(S|\boldsymbol{x})}_{\text{oracle}} \log q_\zeta(\boldsymbol{x})\Big] \tag{S24}
$$

Generally speaking, validation metrics should not be optimised over directly since they are functions of the validation set, and the purpose of a validation set in turn is to give a less biased measure of generalisation than the same metric computed on the training set. However, this may not be too big of a deal here since we are not taking gradients with respect to the oracle.

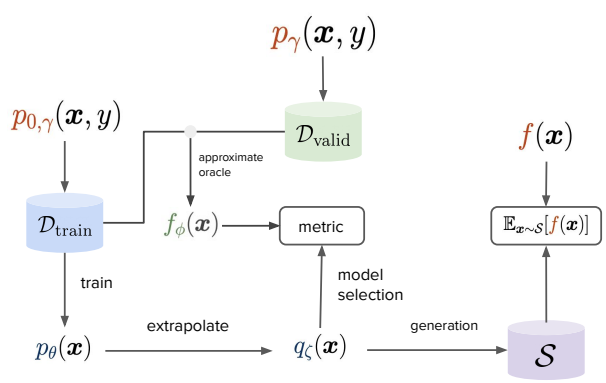


Figure S6: The training and evaluation of CbAS Brookes et al. (2019) in the context of our evaluation framework. The extrapolation equation is described in Equation S23 and involves variational inference to fine-tune $p_\theta(\boldsymbol{x})$ into a search model $q_\zeta(\boldsymbol{x})$.

[10]In their paper the symbol ϕ is used, but here we use ζ since the former is used to denote the validation oracle.

A.3.2 Model inversion networks and the reverse KL divergence

It turns out that there is an interesting connection between the agreement and the *reverse KL divergence* between a specific kind of augmented model distribution and the truncated ground truth $p_{0,\gamma}(\boldsymbol{x}, y)$. To see this, let us re-consider the *generation time* optimisation performed in Kumar & Levine (2020) (which we called *MIN-Opt*), which tries to find a good candidate $\boldsymbol{x} = G_\theta(\boldsymbol{z}, y)$ via the following optimisation:

$$y^*, \boldsymbol{z}^* = \arg\max_{y,\boldsymbol{z}} \; y + \epsilon_1 \underbrace{\log p_\theta(y|G_\theta(\boldsymbol{z}, y))}_{\text{agreement}} + \epsilon_2 \underbrace{\log p(\boldsymbol{z})}_{\text{prior over z}} \tag{S25}$$

$\boldsymbol{z}$ and y can be generated by performing gradient ascent with respect to y and $\boldsymbol{z}$. We can also express MIN-Opt with respect to a batch of $(y, \boldsymbol{z})$'s, and this can be elegantly written if we express it as optimising over a *distribution* $p_\zeta(\boldsymbol{z}, y)$. Then we can find such a distribution that maximises the *expected value* of Equation S25 over samples drawn from $p_\zeta(\boldsymbol{z}, y)$:

$$p_\zeta(\boldsymbol{z}, y)^* := \arg\max_{p_\zeta(\boldsymbol{z},y)} \; \mathbb{E}_{\boldsymbol{z},y\sim p_\zeta(\boldsymbol{z},y)} \Big[y + \epsilon_1 \Big(\log p_\theta(y|G_\theta(\boldsymbol{z}, y)) + \epsilon_2 \log p(\boldsymbol{z}) \Big) \Big], \tag{S26}$$

where e.g. ζ parameterises the distribution, e.g. a mean and variance if we assume it is Gaussian. Although MIN-Opt was intended to be used at generation time to optimise for good candidates, we can also treat it as a validation metric, especially if we replace the training oracle $p_\theta(y|\boldsymbol{x})$ with the validation oracle $p_\phi(y|\boldsymbol{x})$. For the sake of convenience, let us also replace ϵ_1 and ϵ_2 with one hyperparameter η. This hyperparameter can be seen as expressing a trade-off between selecting for large scores y versus ones with large agreement and density under the prior distribution. This gives us the following:

$$\begin{aligned}
p_\zeta(\boldsymbol{z}, y)^* &= \arg\max_{p_\zeta(\boldsymbol{z},y)} \; \mathbb{E}_{y,\boldsymbol{z}\sim p_\zeta(\boldsymbol{z},y)} \Big[y + \eta \Big(\log p_\phi(y|G_\theta(\boldsymbol{z}, y)) + \log p(\boldsymbol{z}) \Big) \Big] \\
&= \arg\max_{p_\zeta(\boldsymbol{z},y)} \; \mathbb{E}_{y\sim p_\zeta(y)} y + \eta \mathbb{E}_{y,\boldsymbol{z}\sim p_\zeta(\boldsymbol{z},y)} \Big[\log p_\phi(y|G_\theta(\boldsymbol{z}, y)) + \log p(\boldsymbol{z}) \Big] && \text{(S27)} \\
&= \arg\max_{p_\zeta(\boldsymbol{z},y)} \; \mathbb{E}_{y\sim p_\zeta(y)} y + \eta \mathbb{E}_{\boldsymbol{x},y,\boldsymbol{z}\sim p_\theta(\boldsymbol{x}|y,\boldsymbol{z}) p_\zeta(\boldsymbol{z},y)} \Big[\log p_\phi(y|\boldsymbol{x}) + \log p(\boldsymbol{z}) \Big]. && \text{(S28)}
\end{aligned}$$

Note that in the last line we instead use the notation $\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|y, \boldsymbol{z})$ (a delta distribution) in place of $\boldsymbol{x} = G_\theta(\boldsymbol{z}, y)$, which is a deterministic operation. We can show that Equation S27 has a very close resemblence to minimising the reverse KL divergence between a specific kind of augmented model and the γ-truncated ground truth, with respect to our distribution $p_\zeta(\boldsymbol{z}, y)$. Suppose that instead of the typical augmented model $p_{\theta,\gamma}(\boldsymbol{x}, y) = p_\theta(\boldsymbol{x}|y)p_\gamma(y)$ we consider one where $\boldsymbol{z}$ and y are drawn from a learnable joint distribution $p_\zeta(\boldsymbol{z}, y)$, and we simply denote $p_\theta(\boldsymbol{x}|y, \boldsymbol{z})$ to be a delta distribution (since $\boldsymbol{x} = G_\theta(\boldsymbol{z}, y)$ is deterministic). We can write this new augmented model as the following:

$$p_{\theta,\zeta}(\boldsymbol{x}, y) = \int_{\boldsymbol{z}} \underbrace{p_\theta(\boldsymbol{x}|y, \boldsymbol{z})}_{\text{GAN}} p_\zeta(\boldsymbol{z}, y) d\boldsymbol{z}. \tag{S29}$$

Although this distribution is not tractable, we will only be using it to make the derivations more clear.

Let us work backwards here: if we take Equation S27 but substitute the inner square bracket terms for the *reverse KL* divergence between the augmented model of Equation S29 and the ground truth $p_\gamma(\boldsymbol{x}, y)$, we

obtain the following:

$$
\begin{aligned}
p_\zeta(\boldsymbol{z}, y)^* &:= \arg\min_{y \sim p_\zeta} \; -\mathbb{E}_{p_\zeta(y)} y + \eta \underbrace{\mathrm{KL}\Big[p_{\theta,\zeta}(\boldsymbol{x}, y) \parallel p_\gamma(\boldsymbol{x}, y)\Big]}_{\text{reverse KL}} \\
&= \arg\min_{p_\zeta} -\mathbb{E}_{y \sim p_\zeta(y)} y + \eta \Big[\mathbb{E}_{\boldsymbol{x}, y \sim p_{\theta,\zeta}(\boldsymbol{x}, y)} \log p_{\theta,\zeta}(\boldsymbol{x}, y) - \mathbb{E}_{\boldsymbol{x}, y \sim p_{\theta,\zeta}(\boldsymbol{x}, y)} \log p_\gamma(\boldsymbol{x}, y)\Big] \\
&= \arg\max_{p_\zeta} \; \mathbb{E}_{y \sim p_\zeta(y)} y - \eta \Big[\mathbb{E}_{\boldsymbol{x}, y \sim p_{\theta,\zeta}(\boldsymbol{x}, y)} \log p_{\theta,\zeta}(\boldsymbol{x}, y) + \mathbb{E}_{\boldsymbol{x}, y \sim p_{\theta,\zeta}(\boldsymbol{x}, y)} \log p_\gamma(\boldsymbol{x}, y)\Big] \\
&= \arg\max_{p_\zeta} \; \mathbb{E}_{y \sim p_\zeta(y)} y + \eta \Big[\mathbb{H}[p_{\theta,\zeta}] + \mathbb{E}_{\boldsymbol{x}, y \sim p_{\theta,\zeta}(\boldsymbol{x}, y)} \log p_\gamma(\boldsymbol{x}, y)\Big] \\
&= \arg\max_{p_\zeta} \; \mathbb{E}_{y \sim p_\zeta(y)} y + \eta \Big[\underbrace{\mathbb{H}[p_{\theta,\zeta}]}_{\text{entropy}} + \mathbb{E}_{\boldsymbol{x}, y, \boldsymbol{z} \sim p_\theta(\boldsymbol{x}|y,\boldsymbol{z}) p_\zeta(\boldsymbol{z}, y)} \big[\underbrace{\log p(y|\boldsymbol{x})}_{\text{agreement}} + \log p_\gamma(\boldsymbol{x})\big]\Big] \\
&\approx \arg\max_{p_\zeta} \; \mathbb{E}_{y \sim p_\zeta(y)} y + \eta \Big[\underbrace{\mathbb{H}[p_{\theta,\zeta}]}_{\text{entropy}} + \mathbb{E}_{\boldsymbol{x}, y, \boldsymbol{z} \sim p_\theta(\boldsymbol{x}|y,\boldsymbol{z}) p_\zeta(\boldsymbol{z}, y)} \big[\underbrace{\log p_\phi(y|\boldsymbol{x})}_{\text{agreement}} + \log p_\gamma(\boldsymbol{x})\big]\Big].
\end{aligned} \tag{S30}
$$

The entropy term is not tractable because we cannot evaluate its likelihood. For the remaining two terms inside the expectation, the agreement can be approximated with the validation oracle $p_\phi(y|\boldsymbol{x})$. Howeer, it would not be practical to estimate $\log p_\gamma(x)$ since that would require us to train a separate density $p_\phi(\boldsymbol{x})$ to approximate it.

For clarity, let us repeat Equation S27 here:

$$
p_\zeta(\boldsymbol{z}, y)^* = \arg\max_{p_\zeta(\boldsymbol{z}, y)} \; \mathbb{E}_{y \sim p_\zeta(y)} y + \eta \mathbb{E}_{\boldsymbol{x}, y, \boldsymbol{z} \sim p_\theta(\boldsymbol{x}|y,\boldsymbol{z}) p_\zeta(\boldsymbol{z}, y)} \Big[\log p_\phi(y|\boldsymbol{x}) + \log p(\boldsymbol{z})\Big]. \tag{S31}
$$

The difference between the two is that (1) there is no entropy term; and (2) $\log p_\gamma(\boldsymbol{x})$ term is replaced with $\log p(\boldsymbol{z})$ for MIN-Opt, which is tractable since the know the prior distribution for the GAN. From these observations, we can conclude that MIN-Opt (S27) comprises an approximation of the reverse KL divergence where the entropy term is omitted and the log density of the *data* is replaced with the log density of the *prior*.

A.3.3 Exponentially tilted densities

Once the best model has been found via an appropriate validation metric, one can train the same type of model on the full dataset $\mathcal{D}$ using the same hyperparameters as before. Ultimately, we would like to be able to generate candidates whose y's exceed that of the entire dataset, and at the same time at plausible according to our generative model. To control how much we trade-off high likelihood versus high scoring candidates, we can consider the *exponentially tilted density* (Asmussen & Glynn, 2007; O'Donoghue et al., 2020; Piche et al., 2022):

$$
p_{\theta,\gamma}(\boldsymbol{x}, y) \exp(\eta^{-1} y - \kappa(\eta)), \tag{S32}
$$

where $\kappa(\eta)$ is a normalisation constant, and smaller η puts larger emphasis on sampling from regions where y is large. Taking the log of Equation S32, we arrive at:

$$
\boldsymbol{x}^*, y^* = \arg\max_{\boldsymbol{x}, y} \; \log p_{\theta,\gamma}(\boldsymbol{x}, y) + \frac{1}{\eta} y, \tag{S33}
$$

In practice, it would not be clear what the best η should be, but a reasonable strategy is to consider a range of η's, where larger values encode a higher tolerance for 'risk' since these values favour higher scoring candidates at the cost of likelihood. Note that for VAEs and diffusion models, $\log p_\theta(\boldsymbol{x}, y)$ will need to be approximated with the ELBO. Interestingly, since diffusion models have an extremely close connection to score-based models, one could 'convert' a diffusion model to a score-based model (Weng, 2021) and derive $\nabla_{\boldsymbol{x},y} \log p_\theta(\boldsymbol{x}, y)$, and this would make sampling trivial.

One potential issue however relates to our empirical observation that predicted scores for generated candidates exhibit very high variance, i.e. the agreement scores are very high (see Figures S7b and S9b). In other words, when we sample some $\boldsymbol{x}, y \sim p_{\theta,\gamma}(\boldsymbol{x}, y)$ (i.e. from the *augmented model*) there is significant uncertainty as to whether $\boldsymbol{x}$ really does have a score of y. One potential remedy is to take inspiration from the MIN-Opt generation procedure (Section A.3.2) and add the agreement term to Equation S33:

$$\boldsymbol{x}^*, y^* = \arg\max_{\boldsymbol{x},y} \ \log p_{\theta,\gamma}(\boldsymbol{x}, y) + \frac{1}{\eta} y + \alpha \log p_\phi(y|\boldsymbol{x}). \tag{S34}$$

Due to time constraints, we leave additional experimentation here to future work.

A.4 Additional training details

A.4.1 Hyperparameters

The architecture that we use is a convolutional U-Net from HuggingFace's 'annotated diffusion model' [11], whose convolutional operators have been replaced with fully connected layers (since Ant and Kitty morphology inputs are flat vectors).

For all experiments we train with the ADAM optimiser (Kingma & Ba, 2014), with a learning rate of 2×10^{-5}, $\beta = (0.0, 0.9)$, and diffusion timesteps $T = 200$. Experiments are trained for 5000 epochs with single P-100 GPUs. Input data is normalised with the min and max values per feature, with the min and max values computed over the training set $\mathcal{D}_{\text{train}}$. The same is computed for the score variable y, i.e. all examples in the training set have a normalised score $y \in [0, 1]$ and those in the validation set have scores $y > 1$.

Here we list hyperparameters that differ between experiments:

- `diffusion_kwargs.tau`: for classifier-free diffusion models, this is the probability of dropping the label (score) y and replacing it with a null token. For classifier guidance models, this is fixed to $\tau = 1$ since this would correspond to training a completely unconditional model.
- `gen_kwargs.dim`: channel multiplier for U-Net architecture
- `diffusion_kwargs.w_cg`: for classifier-based guidance, this is the w that corresponds to the w in Equation 12.

A.4.2 Hyperparameters explored for classifier-free guidance

```
{
    'diffusion_kwargs.tau': {0.05, 0.1, 0.2, 0.4, 0.5},
    'gen_kwargs.dim': {128, 256}
}
```

A.4.3 Hyperparameters explored for classifier guidance

```
{
    'diffusion_kwargs.w_cg': {1.0, 10.0, 100.0},
    'epochs': {5000, 10000},
    'gen_kwargs.dim': {128, 256}
}
```

A.4.4 Classifier guidance derivation

Let us denote $\boldsymbol{x}_t$ as the random variable from the distribution $q(\boldsymbol{x}_t)$, denoting noisy input $\boldsymbol{x}$ at timestep t. Through Bayes' rule we know that $q(\boldsymbol{x}_t|y) = \frac{q(\boldsymbol{x}_t, y)}{q(y)} = \frac{q(y|\boldsymbol{x}_t)q(\boldsymbol{x}_t)}{q(y)}$. Taking the score $\nabla_{\boldsymbol{x}_t} \log q(\boldsymbol{x}_t|y)$ (which

[11] https://huggingface.co/blog/annotated-diffusion

does not depend on $q(y)$), we get:

$$\nabla_{\boldsymbol{x}_t} \log q(\boldsymbol{x}_t|y) = \nabla_{\boldsymbol{x}_t} \log q(y|\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log q(\boldsymbol{x}_t) \tag{S35}$$

$$\approx \frac{-1}{\sqrt{1-\bar{\alpha}_t}} \Big(\epsilon_\theta(\boldsymbol{x}_t, t, y) - \epsilon_\theta(\boldsymbol{x}_t, t) \Big), \tag{S36}$$

where in the last line we make clear the connection between the score function and the noise predictor ϵ_θ Weng (2021). Since we would like to derive the conditional score, we can simply re-arrange the equation to obtain it:

$$\epsilon_\theta(\boldsymbol{x}_t, t, y) = \epsilon_\theta(\boldsymbol{x}_t, t) - \sqrt{1-\bar{\alpha}_t} \nabla_{\boldsymbol{x}_t} \log q(y|\boldsymbol{x}_t) \tag{S37}$$

$$\approx \epsilon_\theta(\boldsymbol{x}_t, t) - \sqrt{1-\bar{\alpha}_t} \nabla_{\boldsymbol{x}_t} \log p_\theta(y|\boldsymbol{x}_t), \tag{S38}$$

where we approximate the classifier $q(y|\boldsymbol{x}_t)$ with our (approximate) training oracle $p_\theta(y|\boldsymbol{x}_t)$. In practice, we can also define the weighted version as follows, which allows us to balance between conditional sample quality and sample diversity:

$$\epsilon_\theta(\boldsymbol{x}_t, t, y; w) = \epsilon_\theta(\boldsymbol{x}_t, t) - \sqrt{1-\bar{\alpha}_t} w \nabla_{\boldsymbol{x}_t} \log p_\theta(y|\boldsymbol{x}_t), \tag{S39}$$

Therefore, in order to perform classifier-guided generation, we replace $\epsilon_\theta(\boldsymbol{x}_t, t)$ in whatever generation algorithm we use with $\epsilon_\theta(\boldsymbol{x}_t, t, y; w)$ instead.

A.4.5 Classifier-free guidance

In classifier-free guidance a conditional score estimator $\epsilon_\theta(\boldsymbol{x}_t, y, t)$ is estimated via the algorithm described in Ho & Salimans (2022), where the y token is dropped during training according to some probability τ. If y is dropped it is replaced with some unconditional token. In other words, the noise predictor (score estimator) is trained both conditionally and unconditionally, which means we have both $\epsilon_\theta(\boldsymbol{x}_t, t)$ as well as $\epsilon_\theta(\boldsymbol{x}_t, y, t)$.

From Bayes' rule, we know that: $p(y|\boldsymbol{x}_t) = \frac{p(y, \boldsymbol{x}_t)}{p(\boldsymbol{x}_t)} = \frac{p(\boldsymbol{x}_t|y)p(y)}{p(\boldsymbol{x}_t)}$, and that therefore the score $\nabla_{\boldsymbol{x}_t} \log p(y|\boldsymbol{x}_t)$ is:

$$\nabla_{\boldsymbol{x}_t} \log p(y|\boldsymbol{x}_t) = \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|y) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t) \tag{S40}$$

We simply plug this into Equation 12 to remove the dependence on $p_\theta(y|\boldsymbol{x}_t)$:

$$\epsilon_\theta(\boldsymbol{x}_t, t, y; w) = \epsilon_\theta(\boldsymbol{x}_t, t) - \sqrt{1-\bar{\alpha}_t} w \nabla_{\boldsymbol{x}_t} \log p_\theta(y|\boldsymbol{x}_t) \tag{S41}$$

$$= \epsilon_\theta(\boldsymbol{x}_t, t) - \sqrt{1-\bar{\alpha}_t} w \Big[\nabla_{\boldsymbol{x}_t} \log p_\theta(\boldsymbol{x}_t|y) - \nabla_{\boldsymbol{x}_t} \log p_\theta(\boldsymbol{x}_t) \Big] \tag{S42}$$

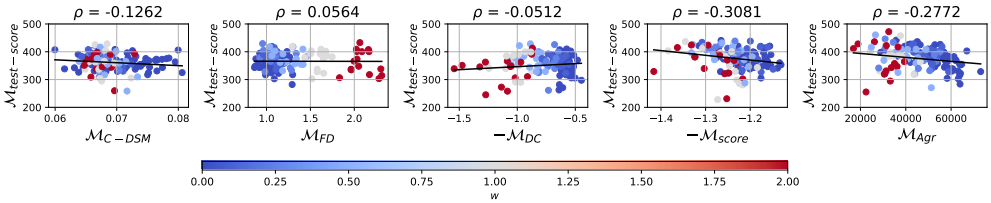
$$= \epsilon_\theta(\boldsymbol{x}_t, t) - \sqrt{1-\bar{\alpha}_t} w \Big[\frac{-1}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\boldsymbol{x}_t, y, t) - \frac{-1}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\boldsymbol{x}_t, t) \Big] \tag{S43}$$

$$= \epsilon_\theta(\boldsymbol{x}_t, t) + w\epsilon_\theta(\boldsymbol{x}_t, y, t) - w\epsilon_\theta(\boldsymbol{x}_t, t) \tag{S44}$$

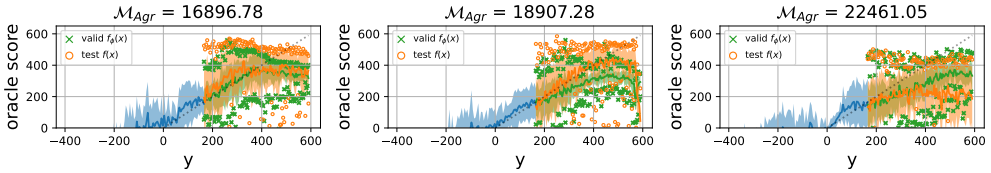
$$= \epsilon_\theta(\boldsymbol{x}_t, t) + w \Big(\epsilon_\theta(\boldsymbol{x}_t, y, t) - \epsilon_\theta(\boldsymbol{x}_t, t) \Big) \tag{S45}$$

A.5 Correlation and agreement plots

A.5.1 Ant Morphology

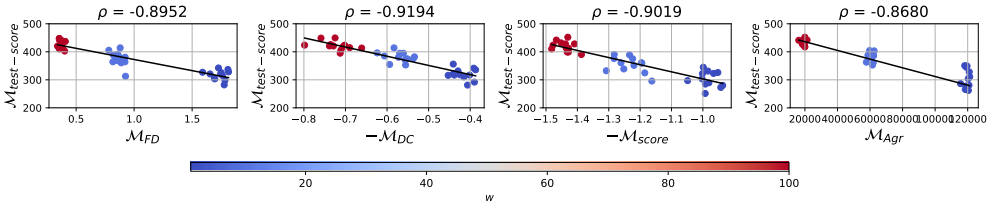


(a) For each validation metric, we plot each experiment's smallest-achieved metric versus the test score (Equation 8). For this dataset, the best performing metrics are $\mathcal{M}_{\text{score}}$ and $\mathcal{M}_{\text{Agr}}$. Coloured points represent the hyperparameter w, which represents the strength of (implicit) classifier guidance.

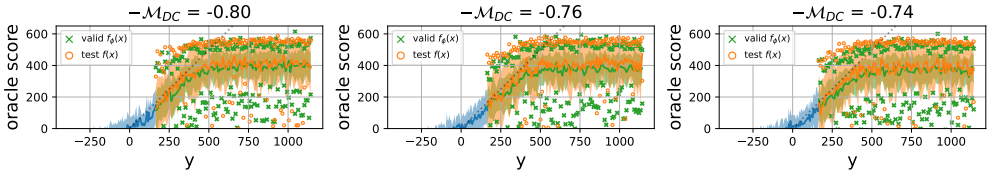


(b) We plot the conditioning $y \in \text{linspace}(y_{\min}, y_{\max})$ against the validation/test oracle predictions for candidates conditionally generated with that y. We call these 'agreement plots' since the sum of squared residuals for each point would constitute the agreement (with a perfect agreement of zero corresponding to a diagonal dotted line on each graph). Here we demonstrate this amongst the best three models *with respect to* $\mathcal{M}_{\text{Agr}}$, since this is most correlated with the test oracle (see Figure S7a). The shaded regions denote ± 1 standard deviation from the mean, and the marker symbols denote the max/min score for each y with respect to either oracle.

Figure S7: Results on Ant Morphology dataset, using the classifier-free guidance variant (Equation 14).



(a) (Due to space constraints, please see Caption S7a for full description.) Here, validation metrics are all roughly similar to each other in terms of Spearman correlation. The guidance hyperparameter $w \in \{1, 10, 100\}$ is coloured orange, green, and red, respectively.

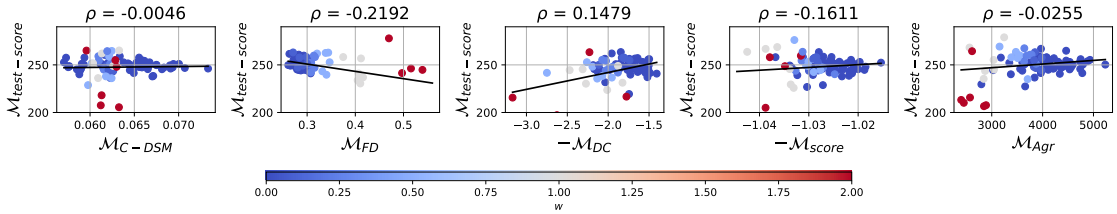


(b) (Due to space constraints, please see Caption S7b for full description.) Based on Figure S8a, the validation metric most correlated with the test oracle is $\mathcal{M}_{\text{Agr}}$. We show the best three experiments here corresponding to that metric.

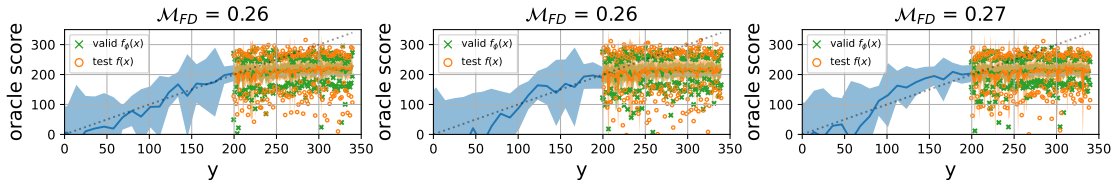
Figure S8: Results on Ant Morphology dataset, using the classifier guidance variant (Equation 12).

A.5.2 D'Kitty Morphology

See Figures S9 and S10.

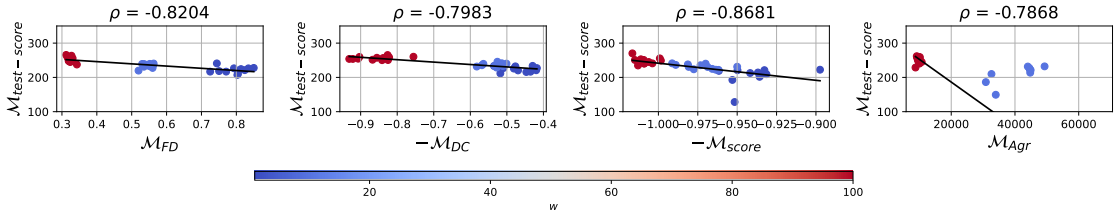


(a) (Due to space constraints, please see Caption S7a for full description.) Here, the validation metric which correlates most with the test oracle is $\mathcal{M}_{\text{FD}}$, with a score of -0.22.

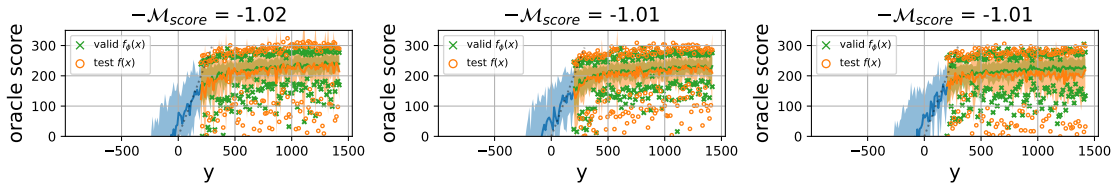


(b) (Due to space constraints, please see Caption S7b for full description.) Based on Figure S9a, the validation metric most correlated with the test oracle is $\mathcal{M}_{\text{Agr}}$. We show the best three experiments here corresponding to that metric.

Figure S9: Results on D'Kitty Morphology dataset, using the classifier-free guidance variant (Equation 14).



(a) (Due to space constraints, please see Caption S7a for full description.) Here, the validation metric which correlates most with the test oracle is $\mathcal{M}_{\text{score}}$, with a Spearman correlation of $\rho = -0.8681$.

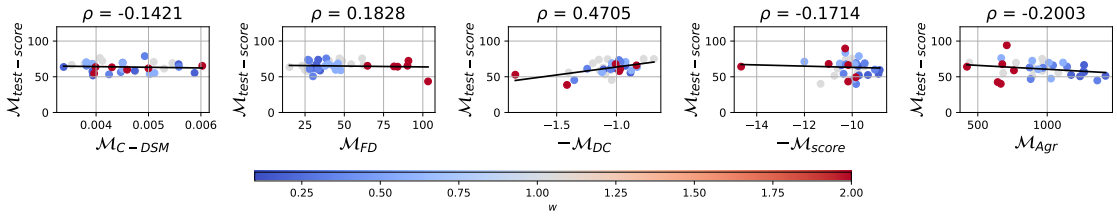


(b) (Due to space constraints, please see Caption for full description.) Based on Figure S10a, the validation metric most correlated with the test oracle is $\mathcal{M}_{\text{Agr}}$. We show the best three experiments here corresponding to that metric.

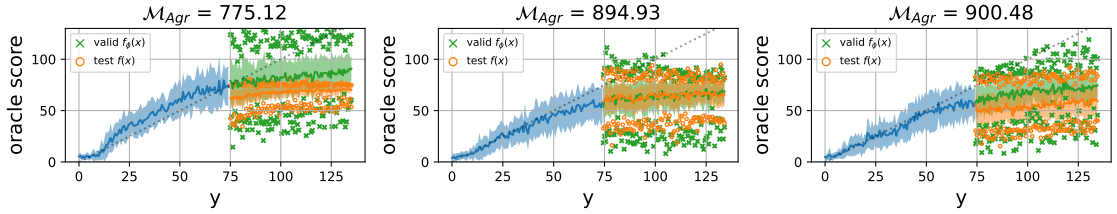
Figure S10: Results on D'Kitty Morphology dataset, using the classifier guidance variant (Equation 12).

A.5.3 Superconductor

See Figure S11.



(a) (Due to space constraints, please see Caption S7a for full description.) Here, the validation metric which correlates most with the test oracle is $\mathcal{M}_{\text{Agr}}$, with a Spearman correlation of $\rho = -0.20$. Points are colour-coded according to w, the strength of the classifier guidance.



(b) (Due to space constraints, please see Caption S7b for full description.) Best three models with respect to the validation metric $\mathcal{M}_{\text{Agr}}$.

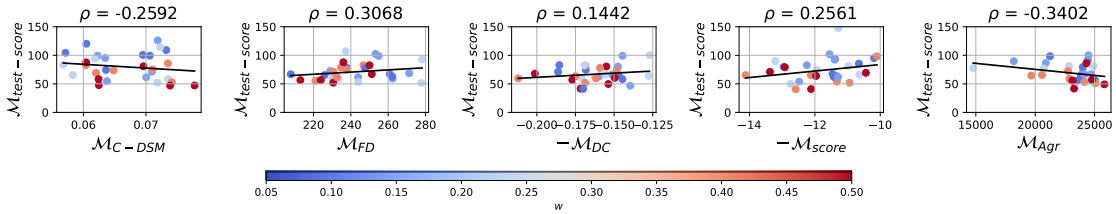
Figure S11: Results on Superconductor dataset (Hamidieh, 2018), using the classifier-free guidance variant (Equation 14).

	50th pt.	100th pt.
$\mathcal{D}_{\text{train}}$	–	0.400
Auto. CbAS	0.131 ± 0.010	0.421 ± 0.045
CbAS	0.111 ± 0.017	0.503 ± 0.069
BO-qEI	0.300 ± 0.015	0.402 ± 0.034
CMA-ES	0.379 ± 0.003	0.465 ± 0.024
Grad.	0.476 ± 0.022	**0.518 ± 0.024**
Grad. Min	**0.471 ± 0.016**	0.506 ± 0.009
Grad. Mean	0.469 ± 0.022	0.499 ± 0.017
REINFORCE	0.463 ± 0.016	0.481 ± 0.013
MINs	0.336 ± 0.016	0.499 ± 0.017
COMs	0.386 ± 0.018	0.439 ± 0.033
Cond. Diffusion (c.f.g.)	**0.518 ± 0.045**	**0.636 ± 0.034**

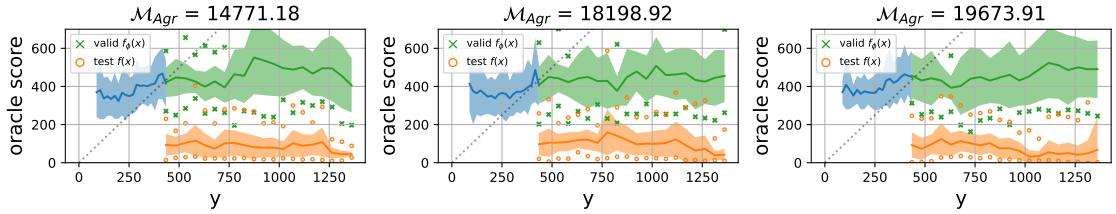
Table S3: 100th and 50th percentile test scores for the Superconductor dataset. Results above our conditional diffusion results (bottom-most row) were extracted from Design Bench Trabucco et al. (2022). For our experiments, each result is an average computed over three different runs (random seeds). Test scores are min-max normalised with respect to the smallest and largest oracle scores in the *full* dataset, i.e. any scores greater than 1 are *greater* than any score observed in the full dataset. Design Bench results are shown for illustrative purposes only, and are not directly comparable to our results due to differences in evaluation setup.

A.5.4 Hopper (50%)

See Figure S12.



(a) (Due to space constraints, please see Caption S7a for full description.) Here, the validation metric which correlates most with the test oracle is $\mathcal{M}_{\text{Agr}}$, with a Spearman correlation of $\rho = -0.3402$. Points are colour-coded according to w, the strength of the classifier guidance.



(b) (Due to space constraints, please see Caption for full description.) Based on Figure S12a, the validation metric most correlated with the test oracle is $\mathcal{M}_{\text{Agr}}$. We show the best three experiments here corresponding to that metric.

Figure S12: Results on Hopper 50%, using the classifier-free guidance variant (Equation 14).

A.6 Additional results

	Ant Morphology	D'Kitty Morphology	Superconductor
Auto. CbAS	0.364 ± 0.014	0.736 ± 0.025	0.131 ± 0.010
CbAS	0.384 ± 0.016	0.753 ± 0.008	0.017 ± 0.503
BO-qEI	0.567 ± 0.000	0.883 ± 0.000	0.300 ± 0.015
CMA-ES	-0.045 ± 0.004	0.684 ± 0.016	0.379 ± 0.003
Gradient Ascent	0.134 ± 0.018	0.509 ± 0.200	**0.476 ± 0.022**
Grad. Min	0.185 ± 0.008	0.746 ± 0.034	0.471 ± 0.016
Grad. Mean	0.187 ± 0.009	0.748 ± 0.024	0.469 ± 0.022
MINs	**0.618 ± 0.040**	**0.887 ± 0.004**	0.336 ± 0.016
REINFORCE	0.138 ± 0.032	0.356 ± 0.131	0.463 ± 0.016
COMs	0.519 ± 0.026	0.885 ± 0.003	0.386 ± 0.018
Cond. Diffusion (c.f.g.)	0.831 ± 0.052	0.930 ± 0.004	0.492 ± 0.112
Cond. Diffusion (c.g.)	0.880 ± 0.012	0.935 ± 0.006	–

Table S4: 50th percentile test scores for methods from Design Bench (Trabucco et al., 2022) as well as our diffusion results shown in the last two rows, with c.f.g standing for *classifier-free guidance* (Equation 14) and c.g. standing for *classifier-guidance* (Equation 12). Each result is an average computed over six different runs (seeds). Test scores are min-max normalised with respect to the smallest and largest oracle scores in the *full* dataset, i.e. any scores greater than 1 are *greater* than any score observed in the full dataset. Design Bench results are shown for illustrative purposes only, and are not directly comparable to our results due to differences in evaluation setup.