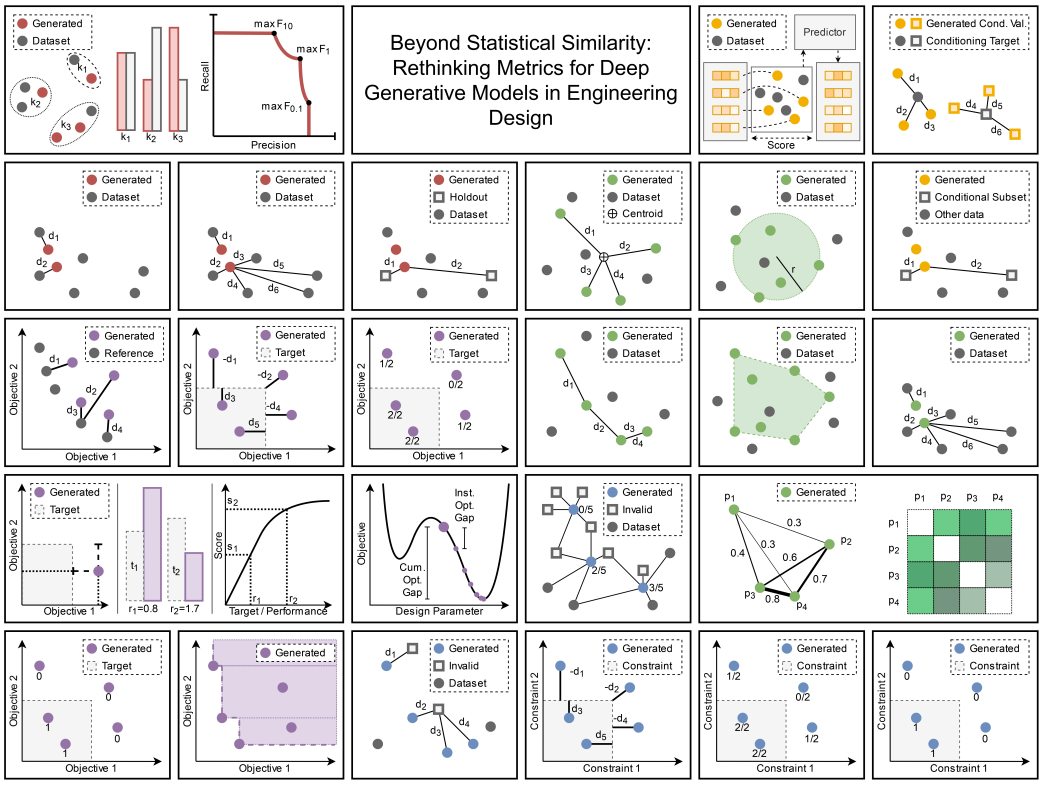
Graphical Abstract

Beyond Statistical Similarity: Rethinking Metrics for Deep Generative Models in Engineering Design

Lyle Regenwetter, Akash Srivastava, Dan Gutfreund, Faez Ahmed



Highlights

Beyond Statistical Similarity: Rethinking Metrics for Deep Generative Models in Engineering Design

Lyle Regenwetter, Akash Srivastava, Dan Gutfreund, Faez Ahmed

- Present a practical guide to evaluation metrics for deep generative models in design.
- Discuss 25+ metrics measuring similarity, diversity, performance, and validity.
- Train and evaluate six deep generative models on easy-to-visualize 2D problems.
- Evaluate state-of-the-art models on bike frame and optimal topology design problems.
- Release all datasets, models, metrics, scoring utilities, and visualization code publicly.

Beyond Statistical Similarity: Rethinking Metrics for Deep Generative Models in Engineering Design

Lyle Regenwetter[a], Akash Srivastava[b], Dan Gutfreund[b], Faez Ahmed[a]

[a]*Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, 02139, MA, United States*
[b]*MIT-IBM Watson AI Lab, 314 Main St, Cambridge, 02142, MA, United States*

Abstract

Deep generative models such as Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), Diffusion Models, and Transformers, have shown great promise in a variety of applications, including image and speech synthesis, natural language processing, and drug discovery. However, when applied to engineering design problems, evaluating the performance of these models can be challenging, as traditional statistical metrics based on likelihood may not fully capture the requirements of engineering applications. This paper doubles as a review and practical guide to evaluation metrics for deep generative models (DGMs) in engineering design. We first summarize the well-accepted 'classic' evaluation metrics for deep generative models grounded in machine learning theory and typical computer science applications. Using case studies, we then highlight why these metrics seldom translate well to design problems but see frequent use due to the lack of established alternatives. Next, we curate a set of design-specific metrics which have been proposed across different research communities and can be used for evaluating deep generative models. These metrics focus on unique requirements in design and engineering, such as constraint satisfaction, functional performance, novelty, and conditioning. Since data-driven design problems are hugely varied in nature, we discuss which evaluation metrics are best suited to different design objectives and representation schemes. We structure our review and discussion as a set of practical selection criteria and usage guidelines. Throughout our discussion, we apply the metrics to models trained on simple-to-visualize 2-dimensional example problems. Finally, to illustrate the selection process and classic usage of the presented metrics, we evaluate three deep generative models on a multifaceted bicycle frame design problem

considering performance target achievement, design novelty, and geometric constraints. We publicly release the code for the datasets, models, and metrics used throughout the paper at `decode.mit.edu/projects/metrics/`.

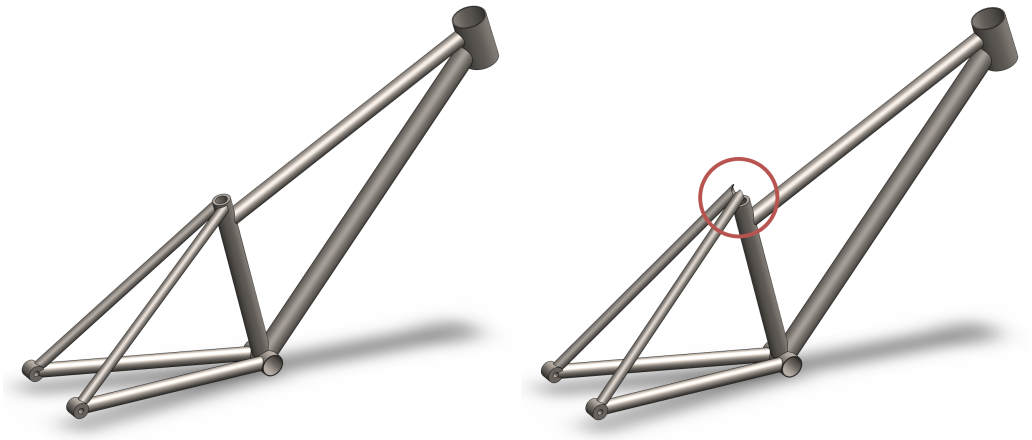
---

1. Introduction

Deep generative models (DGMs) have seen explosive growth across engineering design disciplines in recent years. DGMs like Generative Adversarial Networks (GAN) [1] and Variational Autoencoders (VAE)[2] have dominated image generation problems since 2014, but only bridged the gap to the design community in 2016 [3]. DGMs have since been applied across design domains to problems such as optimal topology generation, airfoil synthesis, and metamaterial design. As promising new methods for image synthesis like diffusion models [4, 5] are introduced in other machine learning fields, researchers in design adapt them to solve challenging design problems [6]. Similarly, transformers, a leading class of generative models for sequences, have dominated natural language generation for years [7, 8, 9], and have seen extensive use in the textual generation of design concepts [10, 11].

DGMs are powerful learners, boasting an unparalleled ability to process and understand complex data distributions and mimic them through batches of synthetic data. In the context of data-driven design, these 'data distributions' are often comprised of a collection of existing designs that lie in some multidimensional design manifold in the same way that a collection of points would form a density distribution in a Euclidean space. From this perspective, it's clear why DGMs are promising data-driven designers. They can study collections of existing designs, understand their distribution, and generate new ones that should belong in the same manifold but do not yet exist. This ability of a DGM to learn and match a distribution is often measured using statistical similarity (i.e, how similar is the distribution of generated designs to the dataset?).

While DGMs' amazing distribution-matching ability is often beneficial, it is limited because, in many design problems, we desire designs to be significantly unique or distinct from existing designs. Additionally, even if distribution-matching is desirable, it is often secondary to meeting problem constraints and achieving functional performance targets. As a result, relying solely on similarity as an objective can lead to misguided design efforts, since very similar designs can have drastically different performance, as illustrated in Figure 1. Historically, design researchers have failed to account for this gap when selecting evaluation metrics, often opting for the classic similarity-based metrics that are prominent in other machine learning domains. In this paper, we provide an exposition of evaluation metrics for DGMs, which we show are important for design practitioners and design



(a) Deflection: $0.05\,cm$ (b) Deflection: $10.46\,cm$

Figure 1: Two very similar bike frames adapted from [12] with drastically different structural performance. By most distance metrics, these bike frames would be the most similar designs among a dataset of thousands. Yet, due to the disconnected geometry highlighted, they experience deflections that differ by over two orders of magnitude when subjected to the in-plane loading scenario in [12].

automation researchers. We intend for the metrics presented to apply to a variety of DGMs for design, and we do not go into detail about specific model architectures. For an introduction to DGMs and an overview of design engineering research using DGMs to date, we refer readers to a review paper on DGMs in engineering design [3].

We broadly divide metrics into five categories which each address an important facet of engineering design. The first of these is similarity, the class of metrics commonly used to evaluate DGMs. Though often overused, similarity is still important, and we present several perspectives and metrics to consider various aspects of similarity. The second category is design exploration, the idea that generated designs are often desired to be unique and varied. The third category is constraint satisfaction, the idea that generated designs must often adhere to a set of explicit or implicit constraints in order to be considered a valid design. The fourth is design quality, the idea that generated designs often have associated functional performance attributes that designers want to optimize. Designers may also want generated designs to meet specific performance targets, necessitating a related set of metrics pertaining to performance target achievement. Finally, the last category is conditioning, the idea that conditional generative models should respect and

adhere to their conditioning information. Additionally, although we discuss a wide variety of evaluation criteria within the five focus areas, this is not an exhaustive review. Many important considerations, such as those listed in the Appendix in 9, are not discussed in detail.

2. Relevant Reviews of Evaluation Metrics

Several papers have discussed and contrasted methods for evaluating DGMs, which are briefly discussed here. Many of these are dedicated review papers discussing metrics in adjacent fields to design or other application domains for DGMs. For example, many of the research and metrics for engineering optimization reviewed in [13] are largely relevant for evaluating DGMs in design. Within design, many research papers partially touch upon metrics for DGMs in specific design sub-disciplines. Certain code repositories have also been introduced, such as the Maryland Inverse Design Benchmark [1] To our knowledge, however, the existing body of research lacks a dedicated discussion of applicable metrics for DGMs which generally apply across design disciplines. We will organize our discussion of relevant work according to several focus areas of this paper: statistical similarity, design exploration, constraint satisfaction, and design quality.

Existing Literature in Similarity Metrics. Image generation [1, 2, 14, 15, 16] and natural language generation [7, 8, 9, 17] remain dominant research thrusts for DGMs. In these fields, the overwhelming majoring of metrics focus on statistical similarity. As such, similarity-related metrics for generative models in these domains have received much attention and careful consideration. Borji [18, 19] provides two complementary reviews which outline a few evaluation metrics for GANs, though most of these metrics generalize to other generative models in computer vision problems. Gatt & Krahmer [20] and Dong *et al.* [21] review the state of natural language generation, including detailed discussions of similarity-based evaluation metrics.

Design domains heavily focused on image-based data have also seen reviews of metrics focusing on statistical similarity. For example, Shah *et al.* [22] review evaluation methods for generative models in synthetic microstructure images. While the metrics discussed in the mentioned papers focus on statistical similarity, this paper argues that thinking beyond statistical similarity

[1] https://pypi.org/project/midbench/

to include factors such as performance, diversity, and constraints is crucial for design and engineering.

Existing Literature in Design Exploration Metrics. The ability of a model to explore is often captured through diversity and novelty. Diversity and novelty are prevalent concepts in design ideation and can be challenging to evaluate even in other context besides DGMs. Mueller & Ochsendorf [23] analyze numerous metrics for design diversity quantification. While useful, these design metrics are often set up to evaluate "generic" designs and may not be sufficient to measure "AI" generated designs. In particular, many metrics struggle with the extreme non-convexity and high dimensionality of the complicated design spaces typically learned by DGMs. This makes their use as an evaluation metric challenging.

Diversity metrics are also found in optimization literature, and are sometimes even built into optimization algorithms as objectives, such as in NSGA-II [24]. However, many metrics from optimization, such as the hypervolume metric, simultaneously evaluate diversity and functional performance. Specialized metrics introduced for DGMs may be better able to evaluate diversity and novelty for complex design distributions [25] and decouple diversity from other performance considerations.

Existing Literature in Constraint Satisfaction Metrics. Because design constraints differ widely by domain, constraint satisfaction has largely been addressed within individual design subdisciplines. For example, Bilodeau *et al.* [26] present several interesting constraint-related metrics for DGMs in their review of generative models for molecular discovery[2]. In some design subdisciplines, similarity-based metrics are being used to indirectly infer constraint satisfaction [22]. In this paper, we will present a set of generalized domain-agnostic constraint satisfaction metrics that extend beyond similarity.

Existing Literature in Design Quality Metrics. Design quality (functional performance) is a ubiquitous consideration across design disciplines. Many relevant functional performance metrics can be found in design optimization literature, as optimization is heavily focused on maximizing functional performance. Metrics in this domain are often used to compare different

[2]This paper also discusses other sorts of metrics, such as rediscovery, which we discuss in this paper

optimization algorithms or to quantify the strength of a set of optimized solutions. However, they can be adapted to measure the relative strength of design sets generated by different DGMs. Riquelme *et al.* [13] review evaluation metrics for multi-objective optimization, documenting their popularity in optimization research and classifying them by type. In this paper, we show that many functional performance evaluation metrics developed for single and multi-objective optimization algorithms are nonetheless relevant to DGMs and should be adopted by researchers. To better handle inverse design problems, Regenwetter & Ahmed [27] propose several evaluation metrics focused on evaluating functional performance in design problems where a performance target is given.

Contributions. In the context of existing work, this paper has the following contributions:

1. We provide a structured review and practical guide on metrics used to evaluate deep generative models on engineering design problems.
2. We highlight the fallacies of statistical similarity and propose a suite of metrics to evaluate design exploration, design constraints, design quality, and conditioning requirements. For each category, we propose multiple metrics, many of which originate in other disciplines, such as multi-objective optimization, natural image generation, and molecule synthesis.
3. We train numerous deep generative models, including GANs, VAEs, MO-PaDGANs, DTAI-GANs, cVAEs, and cGANs, on two-dimensional examples to demonstrate how deep generative models can be evaluated in different circumstances.
4. We present a design case study on bike frame synthesis focusing on performance, diversity, and constraints and discuss how to select and apply appropriate metrics.
5. We present a second design case study on optimal topology generation, evaluating and contrasting state-of-the-art models using the evaluation metrics discussed.

3. Background

Many data-driven design practitioners are intimately familiar with the challenges of model selection. Say a designer wants to train a model to gen-

erate a set of optimal structural topologies that meet several manufacturability constraints and performance targets. The designer trains a BigGAN [16] and a StyleGAN2 [15] and generates thousands of design options using each model. How should the designer select which model to use? The go-to evaluation method for many practitioners using these types of GANs would be the Fréchet Inception Distance (FID) [28], a metric that attempts to quantify similarity to the training dataset. FID can indicate which model generates sets of designs that are most similar to the dataset, but it doesn't provide information on how well they meet specific design constraints, performance targets, or diversity goals. Additionally, FID is calibrated for natural image datasets, so it may not accurately measure similarity for other types of data or structures.

For design researchers looking to train deep generative models (DGMs), commonly used evaluation metrics should be revisited. In this paper, we present a structured guide to help researchers select better evaluation metrics for design problems, focusing on five main facets: similarity, diversity, constraint satisfaction, functional performance, and conditioning. However, before introducing individual metrics in Section 4, we will discuss terminology and several background concepts. Notably, we will discuss common requirements and prerequisites needed to apply the metrics as well as classifications to categorize metrics. We include a tabular summary of the metrics, their requirements, and their categorizations in Table 1.

3.1. A Note on Terminology

In this paper, we broadly use the term 'metrics' to loosely refer to 'evaluation criteria,' not specifically distance metrics. To avoid loss of generality, we broadly use the term 'samples' to refer to the output of a generative model. We also refer to the original data entries as 'datapoints.' Though in generative design problems, 'datapoints' are often existing designs and 'samples' are often generated designs, this may not be universally true. In Section 6, we also refer to constraint-violating data entries as 'invalid datapoints,' which typically correspond to existing design concepts that fail to meet some set of constraints or design requirements.

3.2. Calculating Distance Between Designs

Many distance-based metrics presented in this paper require the ability to calculate distances amongst and in between datapoints and samples. We

Table 1: Overview of metrics discussed in the paper. For each metric, we discuss: 1. The requirements and auxiliary computational cost necessary to evaluate the metric (key: **Aux** – Auxiliary predictive task & training, **CL** – Clustering method, **CFC** – Closed-Form Constraints, **Cond** – Condition parameter calculation, **Const** – Constraint violation test, **Dist** – Distance metric, **DP** – Differentiable method to calculate design performance, **Emb** – Vector embedding, **Inv** – Dataset of invalid designs), **Perf** – Method to calculate design performance. 2. Whether the metric characterizes **points** or **sets** of generated samples. 3. Whether the metric uses a reference set (**binary**) or not (**unary**). 4. Relevant **spaces** to which the metric can reasonably apply. 5. Whether the metric depends on **hyperparameters**. 6. The metric's **bounds**. 7. The optimal **direction** of the metric. Notes: *Assuming L2 Norm (Euclidean) Distance Kernel. Distance Kernel impacts hyperparameters, bounds, and objective. For example, switching to an RBF kernel would result in bounds of $[0, 1]$ ($[-1, 1]$ for signed distances), flipping the direction of the objective, as well as adding a hyperparameter to tune. ** Depending on the clustering method used, a distance metric may suffice instead of a full embedding. *** Problem parameters such as reference points or sets, objective weights and DTAI priority parameters are not considered to be hyperparameters.

Category	Metric:	Requirements/ Eval Costs:	Point/ Set:	Unary/ Binary:	Space:	Hyperparameters***:	Bounds/ Values:	Direction:
Similarity and Distribution Matching	Statistical Distance/Divergence	Depends	Set	Binary	Both	Depends	$[0, \infty]$	Minimize
	Precision-Recall Curves	Emb**, CL	Set	Binary	Both	Yes	N/A	N/A
	Precision ($F_{\beta<<1}$)	Emb**, CL	Set	Binary	Both	Yes	$[0-1]$	Maximize
	Nearest Datapoint*	Dist	Point	Binary	Both	No	$[0, \infty]$	Minimize
	Recall ($F_{\beta>>1}$)	Emb**, CL	Set	Binary	Both	Yes	$[0-1]$	Maximize
	Nearest Generated Sample*	Dist	Set	Binary	Both	Yes	$[0, \infty]$	Minimize
	Rediscovery*	Dist	Set	Binary	Both	Yes	$[0, \infty]$	Minimize
	ML Efficacy	Aux	Set	Binary	Both	Yes	Varies	Varies
Novelty & Diversity	Inter-Sample Distance*	Dist	Point	Unary	Both	No	$[0, \infty]$	Maximize
	Nearest Datapoint*	Dist	Point	Binary	Both	No	$[0, \infty]$	Maximize
	Distance to Centroid*	Emb	Point	Unary	Both	No	$[0, \infty]$	Maximize
	Entropy	Depends	Set	Unary	Both	Depends	$[0, \infty]$	Minimize
	DPP Diversity Score	Dist	Set	Unary	Both	Yes	$[0, \infty]$	Minimize
	Smallest Enclosing Hypersphere	EVAEmb	Set	Unary	Both	No	$[0, \infty]$	Maximize
	Convex Hull	Emb	Set	Unary	Both	No	$[0, \infty]$	Maximize
Design Constraints	Constraint Satisfaction	Const	Point	Unary	N/A	No	{0,1}	Maximize
	Constraint Satisfaction Rate	Const	Point	Unary	N/A	No	$[0-1]$	Maximize
	Signed Distance to Constraint Boundary*	CFC	Point	Unary	Design	No	$[-\infty, \infty]$	Maximize
	Predicted Constraint Satisfaction	Inv, Aux	Point	Unary	Both	Yes	$[0-1]$	Maximize
	Nearest Invalid Datapoint*	Inv, Dist	Point	Binary	Design	No	$[0, \infty]$	Maximize
Performance and Target Achievement	Hypervolume	Perf	Set	Unary	Perf.	No	0-k	Maximize
	Target Achievement	Perf	Point	Unary	Perf.	No	{0,1}	Maximize
	Target Achievement Rate	Perf	Point	Unary	Perf.	No	$[0-1]$	Maximize
	Signed Distance to Target*	Perf	Point	Unary	Perf.	No	$[-\infty, \infty]$	Maximize
	Design Target Achievement Index	Perf	Point	Unary	Perf.	No	0-1	Maximize
	Generational Distance*	Perf	Point	Binary	Perf.	No	$[0, \infty]$	Minimize
	Inst/Cum. Optimality Gap*	DP	Point	Unary	Perf.	No	$[0, \infty]$	Minimize
Conditioning	Conditioning Adherence*	Cond	Point	Unary	Both	No	$[0, \infty]$	Minimize
	Conditioning Reconstruction	Aux	Point	Binary	Both	Yes	$[0, \infty]$	Minimize

indicate which metrics require distance calculations in the requirements column of Tab. 1. Depending on the design representation used, calculating distances between designs can be a significant challenge. We present some strategies to calculate distances in different representation schemes in the appendix. In general, practitioners can choose to directly calculate distances in the original space and modality of the data, or can instead compute an embedding over which to calculate a distance.

3.3. Hyperparameters

Most metrics depend on some parameters, which must be decided before using the metric. This can be viewed as an opportunity for practitioners to carefully consider the parameters they use and to report them clearly when evaluating models. This allows for fair and consistent comparison of models. Additionally, by standardizing parameter values, practitioners can ensure that the evaluation metrics are being used in a fair and unbiased manner. We indicate which metrics depend on hyperparameters in Table 1.

3.4. Point vs. Set Metrics

Certain metrics, such as statistical distance metrics, measure the properties of a set while other metrics measure the properties of an individual point (though point metrics can also be aggregated over a set of points to describe the set). This distinction is particularly significant in design problems where designers may desire to select a single 'finalist' or an elite group of 'finalists' from a set of designs generated by a DGM. Only a subset of the metrics presented will be valuable in evaluating and contrasting individual designs. Hence, we classify the metrics as point metrics or set metrics in Table 1.

3.5. Unary vs. Binary Metrics

We also distinguish between unary and binary metrics. Unary metrics, such as hypervolume, generate a score based on a single design or a set of designs. In contrast, binary metrics such as statistical divergence utilize an additional reference set in generating a score. Often, this reference set is the dataset itself. Hence, we classify the metrics as unary or binary in Table 1.

3.6. Design Spaces and Performance Spaces

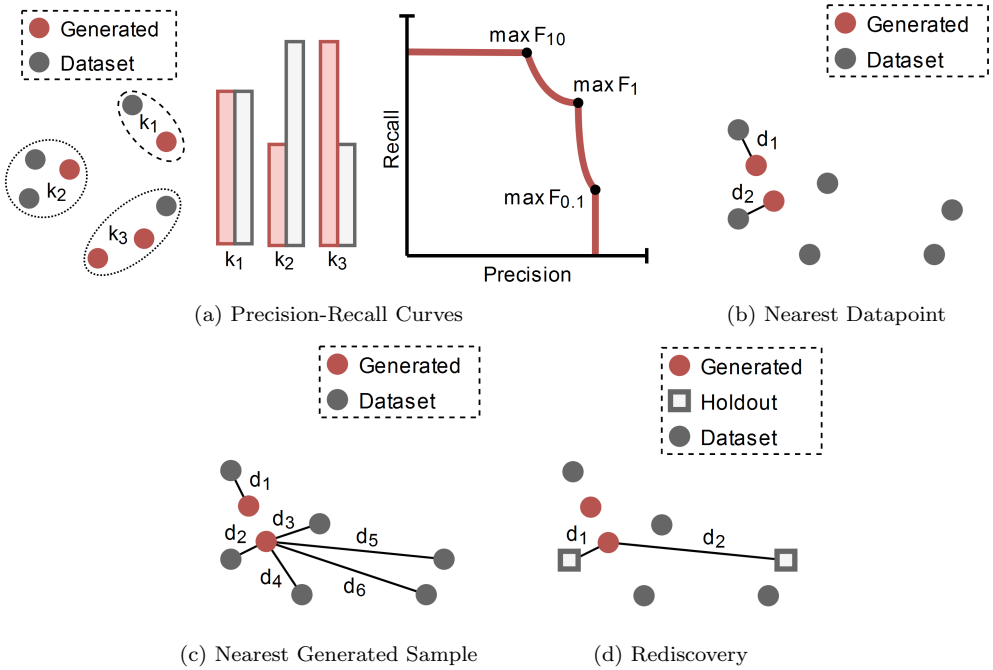
In design, we often frame a design as a point in some design space, without loss of generality across data modalities. However, we also often care about the functional performance of generated designs. We can similarly frame a

design's multi-objective performance as a point in some multi-dimensional performance space. In Table 1, we indicate which metrics make sense to apply to the performance space and which make sense to apply to the design space.

4. Evaluating Statistical Similarity

Having touched on some broad classifications of metrics, we begin our detailed discussion with the first of our five main categories: Similarity. In classic machine learning theory and many classic DGM tasks, such as image generation and natural language generation, deep generative models have a single overarching objective: To generate convincing samples that are new, but generally indistinguishable from the dataset. We typically frame this as a distribution-matching problem, i.e. is the distribution over generated samples identical to the distribution over the dataset? Accordingly, the dominant evaluation metrics in both image synthesis (FID [28], IS [29], KID [30], etc.) and natural language generation (ROUGE [31], BLEU [32], METEOR [33], etc.) have focused on similarity. Historically, similarity has also been the central objective of deep generative models in design tasks [3], since it enforces that generated designs have a general resemblance to the designs used to train the model. For example, if a hypothetical design practitioner trains a model on a dataset of centrifugal pumps, similarity should enforce that their model generates new centrifugal pump designs, rather than reciprocating pumps (or nonsensical designs). However, in practice, the designer probably doesn't want to generate just any pump, but rather a novel and functional one that meets a variety of constraints and performance objectives. Broadly speaking, mimicking the dataset is often not the only desirable objective in design. Accordingly, designers must decide how to trade off similarity against other design objectives, depending on the unique requirements of their problem. In this section, we discuss metrics and tools to evaluate similarity and reserve discussion about other objectives for later sections.

Statistical Divergence/Distance Metrics. Quantifying the discrepancy between two random variables is one of the central themes in machine learning, particularly in ML-based generative modeling [2, 1, 34, 35, 36, 37]. Within the field of deep generative modeling, the two classes of statistical discrepancy measures have been popular, namely ϕ-divergences and Integral Probability



(a) Precision-Recall Curves

(b) Nearest Datapoint

(c) Nearest Generated Sample

(d) Rediscovery

Figure 2: Illustrations of select similarity-related metrics.

Metrics (IPMs). Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability distributions on a measurable space M, such that $\mathbb{P} << \mathbb{Q}$. Then, $\phi-$divergence is defined as,

$$D_\phi(\mathbb{P}, \mathbb{Q}) = \int_M \phi\Big(\frac{d\mathbb{P}}{d\mathbb{Q}}\Big) d\mathbb{Q}. \tag{1}$$

Here, $\phi : \mathbb{R}^+ \mapsto \mathbb{R}$ is a convex function such that $\phi(1) = 0$. A popular example of this class is the Kullbeck-Leibler (KL) divergence, where $\phi(t) = t\log(t)$. Similarly, IPMs can be defined as

$$D_\mathcal{F}(\mathbb{P}, \mathbb{Q}) = \sup_{f \in \mathcal{F}} \Big| \int_M f d\mathbb{P} - \int_M f d\mathbb{Q} \Big|. \tag{2}$$

A popular member of this class is the Maximum Mean Discrepancy (MMD), where $\mathcal{F}$ is set to be the Reproducing Kernel Hilbert Space (RKHS). An intuition behind these two classes of metrics is that the first one tries to measure the ratio (where a ratio of one corresponds to identical sets), while the second one measures the distance between two distributions (where a distance of zero corresponds to identical sets).

Naturally, statistical distances are the optimal choice to measure the similarity of the generated sample distribution to the true data-generating distribution. However, for most high-dimensional problems of interest, computation of such statistical similarity measures is intractable as it requires estimation of the densities induced by the distributions. This has led to the development of estimators of these measures [38, 39, 40, 41, 42]. A popular class of plug-in estimators includes pre-training a classifier [41, 42, 40] to first estimate the log-ratio of the densities and then taking its Monte-Carlo expectation. Another class of estimators relies on neural density estimation [43], where individual densities are estimated using highly non-linear bijective functions [44, 45] and then used to estimate the discrepancy. Similar projection-based approaches also exist for IPMs [38, 46]. However, the efficacy of such estimators is dependent on the modality and the dimensionality of the problem's data. Therefore domain-agnostic estimators of statistical similarity remain largely an open problem.

Consequently, many of the leading similarity metrics are domain-specific, leveraging certain advantages of the domain to calculate. In computer vision, countless domain-specific metrics have seen widespread adoption [18]. Fréchet Inception distance (FID), for example, uses a pre-trained Inception network to calculate vector embeddings for images, assumes a Gaussian distribution over this embedding space, then trivially calculates Wasserstein-2

Distance (Fréchet distance under the Gaussian assumption) between the generated and dataset distributions in this embedding space [28][3]. In text generation methods, the commonly used perplexity metric is also a close relative of statistical distance. Perplexity is defined as the exponential of cross-entropy, which itself is the entropy of the data distribution plus the KL divergence between the generated distribution and the data distribution.

FID and perplexity have been empirically found to correlate well with the human perceptual evaluation of image and text realism. Since human perceptual evaluation is relatively uniform in computer vision and natural language, FID and perplexity are among the most commonly used metrics for model evaluation in their respective fields. In other fields, like design, the perception of realism is much more varied. This non-uniformity, alongside other challenges like data modality, may preclude any generalizable design-specific statistical distance metric from rising to prominence.

Instead, practitioners should evaluate the viability of applying domain-specific methods to design problems on a case-by-case basis. For example, when evaluating similarity in a structural topology generation problem, one may consider using image-based metrics like FID, KID, and IS, which utilize a pre-trained image classifier. These pre-trained image classifiers are often trained on ImageNet, a large computer vision dataset [47]. Accordingly, these metrics are highly biased towards ImageNet [48]. For example, even when evaluating models trained on CIFAR-10, a very similar natural image dataset, researchers have found that inception score significantly misrepresented model performance [49]. At the beginning of Section 3, we shared an anecdote about a practitioner considering whether to evaluate generated structural topologies using FID. In a case like this, it is uncertain whether the low-dimensional latent representations extracted from a network trained on animal or food images in ImageNet would contain any useful information about structural topologies, making metrics like FID a dubious choice for this type of problem. Therefore, practitioners must be cautious about applying image-based metrics in other domains and carefully evaluate their suitability.

In general, statistical distance metrics are excellent tools to evaluate similarity and ensure that generated designs are similar to the training data, but

[3]Other popular metrics calculate different IPMs, such as Maximum Mean Discrepancy in Kernel Inception Distance (KID) [30]. Other variants like Inception Score (IS) use the statistical distance between marginal and true label distributions [29].

are often challenging to estimate in high-dimensional problems. When design data is similar to natural image datasets or natural text corpora, off-the-shelf variants of statistical divergence methods can be effectively used. However, when reliably estimating statistical distance is infeasible or when practitioners desire more nuance in evaluating similarity, they can instead turn to a variety of other methods, which we present in the following subsections.

4.1. Decoupling 'Realism' and 'Coverage'

Researchers often point to a key shortcoming of statistical distance metrics, namely their inability to decouple two separate ideas in distribution matching: 'realism' and 'coverage.' Realism is the idea that generated designs should resemble the dataset[4]. Coverage is the idea that the entire spread of the dataset should be represented by generated designs. While any model that achieves perfect (zero) statistical distance must achieve both perfect realism and coverage, an imperfect model can suffer from an unknown balance of imperfect realism or coverage, which is difficult to diagnose with only a single score. To combat this, methods like precision-recall curves analyze generative models with an entire precision-recall tradeoff front between these two factors.

Precision-Recall Curves(Fig. 2a). Precision-recall curves are borrowed concepts from supervised classification but have been adapted as metrics for generative models. The concept was originally proposed by Lucic *et al.* [50] as a method to benchmark generative models on datasets with known data manifolds. It was then extended to data with unknown distributions by Sajjadi *et al.* [51], allowing for practical evaluation of DGMs on 'real' datasets. In Sajjadi *et al.*'s framework, generated data and original data are pooled, then clustered using k-means. A precision-recall (PR) curve is then calculated by comparing the proportion of generated versus original data within each discrete cluster over a sweep of a weighting parameter. Other methods have also been proposed to generate PR curves from arbitrary distributions, bypassing the need for discrete binning of the data [52]. From the curve, summary scores like Area-Under-the-Curve (AUC) and a maximum F_1 score can also be derived. We refer the reader to [51] for mathematical reasoning and visual examples behind the PR curves for generative models.

[4]The term 'realism' can be misleading since it implies that the dataset reflects reality (i.e., covers the entire space of 'realistic' data), which is typically untrue.

4.2. Similarity of Generated Data to Dataset ('Realism')

Although precision-recall curves demonstrate how a DGM model tends to balance realism and coverage, in some cases, we may care more for one or the other. In problems where we particularly care about realism, we can calculate a 'biased' version of the F_1 score to estimate precision or use a simple distance-to-dataset metric.

Precision. Precision for generative models captures the fraction of generated designs falling within the support of the dataset. Extracting singular precision values from PR curves is challenging since it's unclear which value to select, or how to average values. Instead, Sajjadi *et al.* [51] propose using the maximum F_β score for some $\beta << 1$ over all precision-recall pairs in the PR curve as a proxy for precision.

Nearest Datapoint (Figure 2b). For a simple estimate of 'realism,' practitioners can calculate the distance to the nearest datapoint for every generated sample. Despite its simplicity, the score is an effective method to capture local proximity to the dataset. Nearest datapoint can also be averaged over a set of samples to estimate the 'realism' performance of a DGM. This metric is particularly important for applications of DGMs in data augmentation, as it demonstrates that generated synthetic designs (samples) are similar to a dataset of existing designs (datapoints).

4.3. Dataset Coverage

In some design applications, there might be a need to focus more on design space coverage to ensure that all key modalities of the design space are reflected in generated designs. In such cases, practitioners can instead estimate recall or use other metrics to quantify coverage.

Recall. Like precision, recall is a metric introduced by Sajjadi *et al.* [51]. Though extracting singular recall values from PR curves is similarly challenging, practitioners can select some maximum F_β score for some $\beta >> 1$ to estimate recall.

Nearest Generated Sample (Figure 2c). For a simple approach to capture dataset coverage, practitioners can calculate the distance to the nearest generated sample for every original datapoint. This score can be averaged over the dataset to evaluate a set of generated samples. However, this score depends on the size of the generated sample set, necessitating the selection of a set size tuning parameter for standardization.

Rediscovery (Figure 2d). Rediscovery is an evaluation technique that evaluates an algorithm's ability to rediscover datapoints that were withheld during training. It has been used in the molecule synthesis domain [26], to check whether a DGM could rediscover a known molecule that was withheld from the training data. However, we hypothesize that rediscovery can be a useful metric for many other design problems. While rediscovery is originally used in discrete problems where the rediscovery rate can be calculated as the exact proportion of withheld designs rediscovered in a generated set [22], the metric can easily be relaxed to support other data modalities. Instead of calculating a proportion of datapoints exactly rediscovered, practitioners can instead calculate the distance from a withheld datapoint to the nearest sample in a generated sample set. Effectively, this performs a nearest generated sample evaluation over the set of withheld designs [5]. Though it requires both a tuning parameter (holdout size) and the foresight to remove a split of the dataset before training, rediscovery is an elegant extension of pure 'coverage' metrics that further quantifies simple generalization capabilities of the model. As such, we feel that this metric deserves consideration in other design disciplines beyond molecular design.

4.4. Evaluating Effective use of Generated Data in Downstream Tasks

A common approach to measuring the similarity of a generated sample set to the training dataset is to check whether the generated set serves as an effective stand-in for some auxiliary task. If a practitioner is generating design data with a specific downstream task in mind, it may be viable to directly evaluate generated samples on this task as a metric. In other scenarios, artificially constructed tasks can serve as an effective method to evaluate generated samples.

ML Efficacy. Auxiliary machine learning tasks, such as classification, are often used to evaluate generated sample sets in a process known as ML efficacy testing. Several variations of these scores have been proposed, as in [53, 54]. Often a supervised machine learning model is trained on the generated dataset and then tested on the original data. Ideally, the performance on the original data should be as strong as possible. Alternatively, one model is first trained on the original dataset and tested on a split of the original

[5] We also note that other metrics can also be evaluated on a "test set," though we feel that evaluating coverage on a holdout set is a particularly insightful choice.

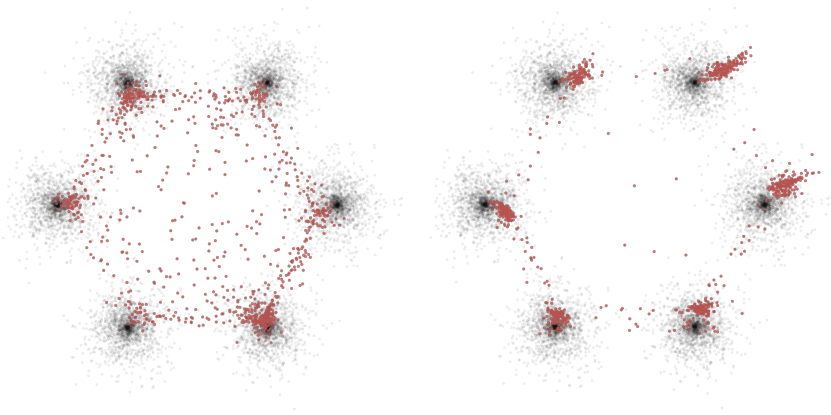
data. Then, an equivalent model is trained on a set of generated samples but still evaluated on the same split of original data. In this case, the difference in performance constitutes the ML efficacy score, which is minimized in the ideal case where the two models have comparable performance.

4.5. Demonstration of Statistical Similarity Metrics

Thus far, we have introduced a variety of metrics to quantify a model's ability to generate distributions of designs that match the training dataset. To showcase the use of these metrics, we evaluate two classic generative models, a Variational Autoencoder (VAE) and a Generative Adversarial Network (GAN)[6] on a synthetic dataset which challenges models to learn six non-overlapping data modes. Details on model architecture, metrics settings, and training are included in the appendix. Plots of the generated data are shown in Figure 3. Gray points denote training data, and red points denote samples. Scores for a few selected metrics for these two models are shown in Table 2. If a practitioner was using a single distribution-matching metric, such as MMD, to compare the two models, they would think that the GAN is the stronger performer among the two. However, when looking at an array of metrics, one can note that the GAN is the stronger performer in accuracy-based metrics such as F10 and nearest datapoint (better performance is highlighted by bold text). In contrast, the VAE outperforms the GAN in many coverage-based metrics such as nearest generated sample, rediscovery, and F0.1. In overall distribution-matching metrics, results are mixed, with the GAN outperforming in MMD and falling behind in F1 and AUC. The GAN also performed significantly better in machine learning efficacy on this dataset. Compared to a single metric, this suite presents a more nuanced picture, and the final model selection could vary based on the end goal. If coverage is more important, then the VAE may be preferred, while the GAN may be preferred for realism[7]. However, even the best coverage of the design space does not guarantee the novelty or diversity of generated designs. Therefore, in the following section, we introduce and discuss metrics to evaluate the capabilities of DGMs to generate diverse and novel designs.

[6]The goal of this example is to compare any two DGM models; hence, the state-of-art models and strong instantiations were intentionally not chosen as they performed very well on a two-dimensional case.

[7]We note that these results need not generalize to other datasets or architectures.

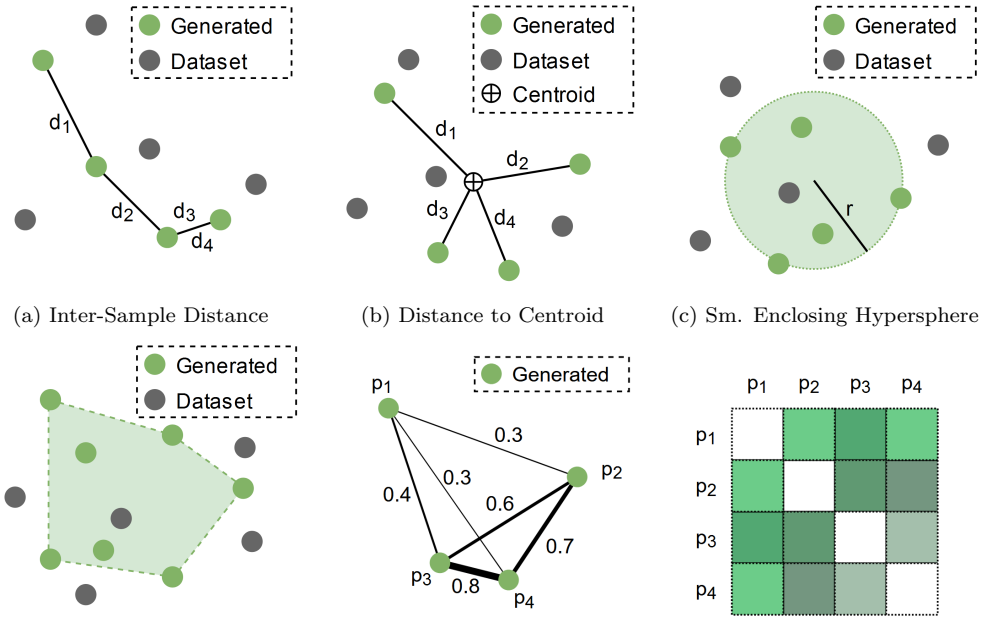


(a) VAE (b) GAN

Figure 3: Distributions generated by a Variational Autoencoder and Generative Adversarial Network (red) overlaid over training data (gray). The GAN dominates in accuracy-related metrics while the VAE outperforms in coverage.

Table 2: Distribution-matching scores for models in Figure 3. The GAN dominates in accuracy-related metrics while the VAE outperforms in coverage. Point metrics are averaged over the generated set.

(Avg.) Scores:	VAE	GAN
Nearest Datapoint	0.043	**0.018**
Nearest Generated Sample	**0.090**	0.100
Rediscovery	**0.093**	0.099
Precision-Recall Curve F1	**0.475**	0.386
Precision-Recall Curve F10	0.907	**0.934**
Precision-Recall Curve F0.1	**0.804**	0.753
Precision-Recall Curve AUC	**0.468**	0.379
Maximum Mean Discrepancy	0.045	**0.013**
Machine Learning Efficacy	0.675	**0.765**



(a) Inter-Sample Distance (b) Distance to Centroid (c) Sm. Enclosing Hypersphere

(d) Convex Hull (e) DPP Diversity Score

Figure 4: Illustrations of select diversity and novelty metrics.

5. Evaluating Design Exploration

Design exploration can be an important consideration in many fields, such as product design, architecture, and engineering, where new and innovative solutions are often sought after. A model that successfully explores the design space will typically generate diverse sets of novel designs. Diversity often goes hand-in-hand with generalizability of a model and diversity-aware DGMs have even been shown to avoid common generalizability pitfalls such as mode collapse [55, 25]. Design novelty refers to the degree to which a design is new or unique compared to existing designs. Achieving novelty is particularly difficult for data-driven generative models since it is somewhat contrary to its default objective of learning and mimicking a dataset. In design methodology literature, novelty is typically considered a point metric, whereas diversity (or variety) is typically considered a set metric. In this section, we discuss several metrics with which to quantify the novelty of generated samples and the overall diversity of generated sample sets.

5.1. Novelty

The design methodology community has a rich body of work classifying and defining novelty. A common distinction is 'psychological novelty' (P-novelty) versus 'historical novelty' (H-novelty). A design is considered P-novel if the idea is new for the person who generated it. In contrast, a design is H-novel if an idea has never appeared in history before [56]. If we treat our generative model as the 'designer' and consider the dataset as the set of designs that the designer has seen, we argue that the P-novelty can be estimated for a DGM. However, since we can't assume that our dataset is comprised of every design ever conceptualized, estimating H-novelty is challenging, even if we use the dataset as a reference distribution.

Inter-Sample Distance (Figure 4a). A simple metric to measure the P-novelty of a generated sample is the distance to the nearest other generated sample. In this simple form, the metric is a strong assessment of local novelty. However, the metric has also been relaxed to use the distance to the n^{th} nearest neighbor [57].

Nearest Datapoint (Figure 2b). The nearest datapoint metric (previously introduced as a similarity metric in Section 4.2) is a P-novelty metric, measuring how far from the nearest datapoint a generated sample is. Whereas the metric was best minimized as a metric for accuracy, when used to evaluate novelty, a larger score is preferable. Nearest datapoint is a very insightful metric when checking for data copying, where the DGM will overfit and memorize individual datapoints to replicate while sampling. One limitation of the metric is its large sensitivity to individual datapoints. Nevertheless, the nearest datapoint metric has been used in design literature [25] to demonstrate the capability of deep generative models to create novel designs.

Distance to Centroid (Figure 4b). Instead of using the distance to the nearest samples, practitioners can instead quantify the novelty of a generated sample using the distance from the sample to a single point which summarizes a set of generated samples, such as the centroid or the geometric median. As Brown & Mueller [58] note, the choice of centroid or median can yield very different results. Though simple and cheap to compute, the metric may carry implicit assumptions about convexity. For example, in a torus-like distribution, the centroid may in fact be quite novel. Mueller & Ochsendorf [23] propose a variant to adapt the metric into a diversity score using the maximum distance

to the centroid or median. We discuss diversity in more detail in the next subsection.

5.2. Diversity

Design diversity is closely related to the concept of "entropy" in information theory and refers to the variety or range of different solutions or designs that are generated for a given problem. In the context of design problems, it can refer to the degree to which a set of solutions to a problem encompasses different styles, forms, or variations. While novelty is a point metric, diversity measures a property of a set of designs, though many averaged novelty metrics may often closely correlate with diversity.

Diversity can be broken down into two components: uniformity and spread. Uniformity measures the relative distance between designs. Spread, also known as 'extent' in multi-objective optimization literature, measures the range of designs within the generated distribution [13]. Imagine the design space as a balloon with small balls inside it. The spread can be thought of as the diameter of the balloon. A bigger balloon will have a higher extent diversity score. However, two balloons with the same diameter may have different uniformity diversity scores, for example, if one balloon has all the balls evenly distributed, while the other has most of the balls stuck in one corner. Many diversity metrics, such as entropy, combine uniformity and spread into a single value. Below, we discuss a few of these diversity metrics[8].

Entropy. In general, entropy scores are a metric for design diversity when evaluated on a set of generated designs. Some popular metrics include the Shannon entropy index, Herfindahl–Hirschman Index (HHI) [59], Gini-Simpson index, and inverse Simpson index [58]. When practitioners have discrete data, they can directly calculate the entropy. When working with continuous representations, however, practitioners must estimate entropy from samples, a well-studied statistical problem [60]. Entropy captures both uniformity and spread, and its properties are well-grounded in mathematics and information theory.

[8]Averaged novelty metrics are also often more focused on either spread or uniformity. For example, averaged inter-sample distance and averaged nearest datapoint are often strong measures of uniformity, whereas averaged distance to centroid is more focused on spread.

Smallest Enclosing Hypersphere (Figure 4c). The smallest enclosing hypersphere metric is a purely spread-focused metric that identifies the hypervolume of the smallest hypersphere that encloses all generated samples. Originally proposed for novelty measurement [61], the calculation is nontrivial for high-dimensional data and is often approximated to reduce cost [58]. Smallest enclosing hypersphere is highly sensitive to relative scaling between parameters. It also makes a convexity assumption and is sensitive to outliers.

Convex Hull (Figure 4d). The convex hull is another spread-focused metric defined as the smallest convex set that includes a set of generated samples. The total hypervolume enclosed within the generated set's convex hull is a common metric for diversity [62]. Brown & Mueller [58] found the convex hull to agree with human assessments of novelty in small low-dimensional design problems, compared to competing metrics. Like the smallest enclosing hypersphere, the convex hull makes a convexity assumption and is often sensitive to outliers.

DPP Diversity Score (Figure 4e). Determinantal Point Processes (DPP) can be used in conjunction with distance metrics to evaluate a diversity score of a generated sample set. DPPs calculate a score based on the eigenvalues of a matrix constructed using distances between points from a generated sample set [63]. Like entropy, DPP diversity captures both uniformity and spread. The benefit of DPP is its ease of calculation for high-dimensional data, as it only requires a positive-semidefinite kernel as an input. However, it is sensitive to design duplicates, as the determinant collapses to zero when any of the eigenvalues is zero. We note that DPP diversity is highly nonlinear and small changes in score may imply sizeable changes in diversity

5.3. Demonstration of Design Exploration Metrics

In this section, we introduced a variety of diversity and novelty-related metrics. To demonstrate their performance, we use the same representative synthetic data problem introduced in Figure 3[9]. Scores are shown in Table 3. The VAE dominates in average novelty metrics (nearest datapoint and inter-sample distance), owing largely to the fact that the generated samples are more spaced apart and span regions of the space outside of the dataset. However, the GAN outperforms the VAE in diversity metrics focused on

[9]Point metrics (novelty) are averaged over generated samples.

Table 3: Diversity and novelty scores for generated distributions from Figure 3. The VAE dominates in average novelty, while the GAN dominates in spread-focused metrics. Point metrics are averaged over the generated set.

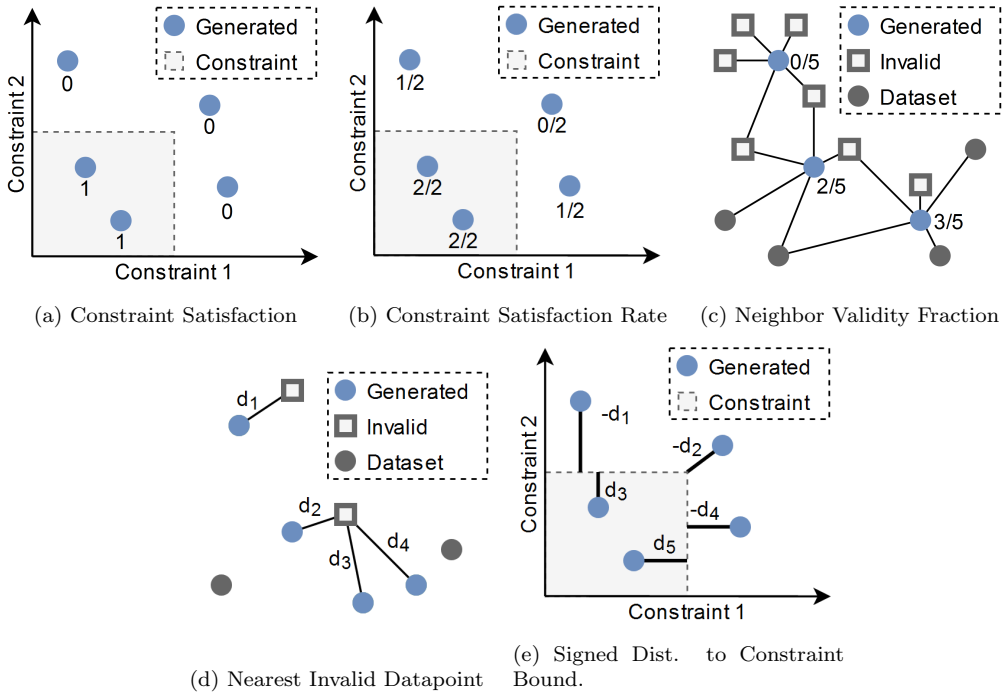
(Avg.) Scores:	VAE	GAN
Nearest Datapoint	**0.043**	0.018
Inter-Sample Distance	**0.030**	0.019
Convex Hull	5.651	**7.173**
DPP Diversity	**14.398**	14.688
Distance to Centroid	1.120	**1.391**

spread (convex hull and distance to centroid) because samples generated by the GAN span a larger convex space. This illustrates a shortcoming with certain diversity metrics where the potentially diverse samples in the center of the space generated by the VAE do not contribute to the score. The DPP diversity score, which takes into account both uniformity and spread, favors the VAE when using a Euclidean radial basis function (RBF) kernel.

From this two-dimensional example, we would like to highlight two points. First, average novelty metrics are easily confused for diversity metrics, but as this example illustrates, they do not always agree with diversity scores. Second, practitioners need to be aware of convexity assumptions and consider both uniformity and spread when evaluating a model's diversity.

6. Evaluating Design Constraints

Many design problems have constraints, which are limitations placed on the possible designs within a given problem or task. These constraints can be physical, such as geometric constraints, or functional, such as performance or cost requirements. They are used to guide the design process and ensure that the final solution meets certain requirements and is feasible to implement. Constraints are commonly driven by materials or manufacturing limitations, industry standards, or nonnegotiable safety requirements. Unlike performance targets, which we discuss in Section 7.2, we *must* satisfy constraints for the design to be valid. Appropriately handling constraints is extremely important for generative models in design, and ensuring that DGMs can respect design constraints is a critical consideration in evaluating their performance. In this section, we propose several techniques for quantifying constraint satisfaction in DGMs. Much like functional performance



(a) Constraint Satisfaction (b) Constraint Satisfaction Rate (c) Neighbor Validity Fraction

(d) Nearest Invalid Datapoint (e) Signed Dist. to Constraint Bound.

Figure 5: Illustrations of select constraint satisfaction metrics.

metrics, constraint evaluation metrics are highly dependent on known information about the problem. Therefore, when little or no information about constraints is available, it is possible that no metric will be practically viable for a particular problem.

6.1. Leveraging Constraint Violation Tests

In many design problems, there exist known methods (using analytical equations, rules, simulations, or physics knowledge) to test whether a design is valid with respect to each of the problem constraints. Since closed form constraint definitions are rare in design problems practioners may instead have to turn to such methods to evaluate constraint satisfaction. Below, we list a few methods that leverage such constraint violation tests, when available.

Constraint Satisfaction (Figure 5a). Provided that practitioners have access to some black-box constraint satisfaction test, perhaps the simplest possible constraint violation metric is a simple binary constraint satisfaction score

(i.e., does a generated sample simultaneously meet all constraints?). When averaged over a generated sample set, this can effectively serve as an indicator for the proportion of generated samples that are valid. In practice, more versatile scores are often more informative, especially in problems with multiple constraints.

Constraint Satisfaction Rate (Figure 5b). A variant of the simple constraint satisfaction score is the constraint satisfaction rate, which quantifies the proportion of all constraints met by a single generated sample. If different constraints have different priority weightings, this score can be weighted by the priority of the various constraints.

6.2. Metrics that Leverage Datasets of Invalid Designs

Unfortunately, it is common not to have even a black box constraint evaluator or methods allowing the direct estimation of the distance to the constraint boundary. In such cases, practitioners may have access to or may be able to procedurally generate a large collection of infeasible or invalid designs. These datasets of constraint-violating (invalid) designs provide a tool to approximate the constraint adherence of generated designs.

Predicted Constraint Satisfaction. When practitioners have access to a reference set of constraint-violating datapoints, they can use a classifier to predict the constraint satisfaction of their generated samples. This classifier can be something complex like a neural network, or something simple and robust like k-nearest neighbors. In the case of k-nearest neighbors, the fraction of neighbors that are valid provides a likelihood that a generated sample satisfies constraints, which serves as the score, as shown in Figure 5c.

Nearest Invalid Datapoint (Figure 5d). When practitioners have access to a reference set of constraint-violating datapoints, they can calculate the distance from each generated sample to the nearest known invalid datapoint. The underlying assumption is that samples near constraint-violating datapoints are also likely to be constraint-violating. This gives a rough approximation of the distance to the constraint boundary.

6.3. Leveraging Mathematically-Defined Constraint Boundaries

Though rare, practitioners may sometimes have access to a closed-form mathematically-defined constraint boundary. They can usually then calculate distances from generated design samples to the constraint boundary, which can be highly informative.

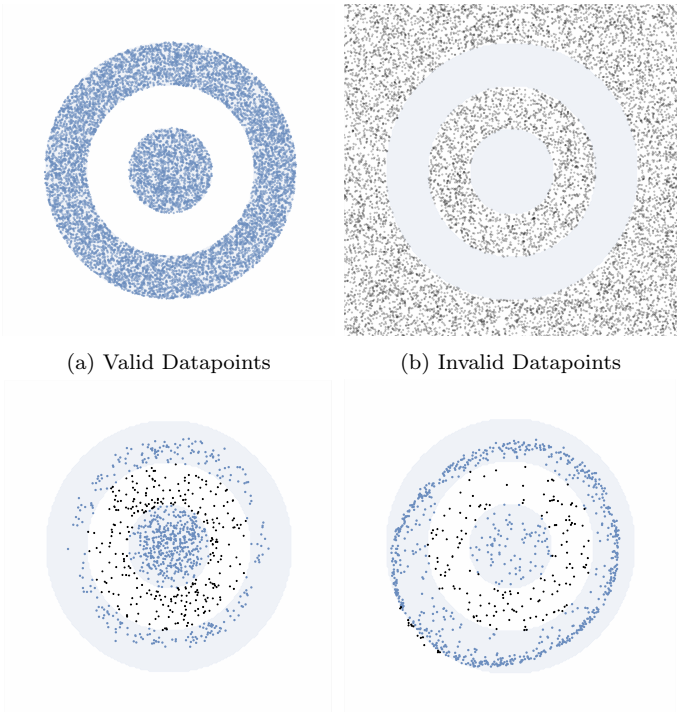
Signed Distance to Constraint Boundary: (Figure 5e). The distance to constraint boundary metric is most informative as a signed distance field (SDF), with generated samples satisfying the constraints having positive distances and samples violating the constraints having negative distances. Indicating by what margin samples are satisfying or violating constraints can be significantly more informative than a simple binary criterion or proportion of constraints met.

6.4. Demonstration of Design Constraint Metrics

Having presented a variety of metrics for evaluating constraint satisfaction, we again showcase these metrics on a simple 2-dimensional problem. To demonstrate the constraint adherence task, we test a GAN and a VAE on a variant of a concentric ring problem that we created for this task. In this problem, we create a non-convex feasible design space consisting of a concentric circle and ring, separated and surrounded by two infeasible regions, respectively (Figure 6a). To support the nearest invalid datapoint metric, we also include a dataset of invalid datapoints, shown in Figure 6b, though these invalid datapoints are not used during training. Both the GAN and the VAE struggle to avoid the invalid area of the design space, as seen by points overlapping with the infeasible regions in Figures 6c and 6d. However, as shown in Table 4, the GAN outperforms the VAE in every metric, indicating that it is better suited to generate feasible designs in this problem. Note that the metrics presented use differing amounts of problem information. In cases when these metrics don't agree, giving the more informed metric precedence is recommended (e.g. if available, use constraint satisfaction over predicted constraint satisfaction). With this, we conclude our discussion about constraint adherence metrics and move on to consider metrics that evaluate design performance.

7. Evaluating Design Quality

In problems like image synthesis, 'quality' and 'realism' are almost synonymous. The more realistic the generated images, the higher their 'quality' and the higher the performance of the generator. This makes sense for problems like human face generation, where if a generated image looks like an existing human face, it might be considered a more realistic and higher-quality image. In design, 'quality' is typically not associated with similarity to existing designs. Instead, the quality of designs is often governed by an



(a) Valid Datapoints (b) Invalid Datapoints

(c) VAE (d) GAN

Figure 6: Distributions generated by a Variational Autoencoder and Generative Adversarial Network. Both the VAE and GAN struggle to observe the non-convex constraint dividing the two valid regions of the design space. Generated samples that violate constraints are shown in black, while valid generated samples are shown in blue. The GAN demonstrates generally higher constraint satisfaction performance.

Table 4: Constraint adherence scores averaged over the generated distributions in Figure 6. The GAN demonstrates generally higher constraint satisfaction performance. Point metrics are averaged over the generated set.

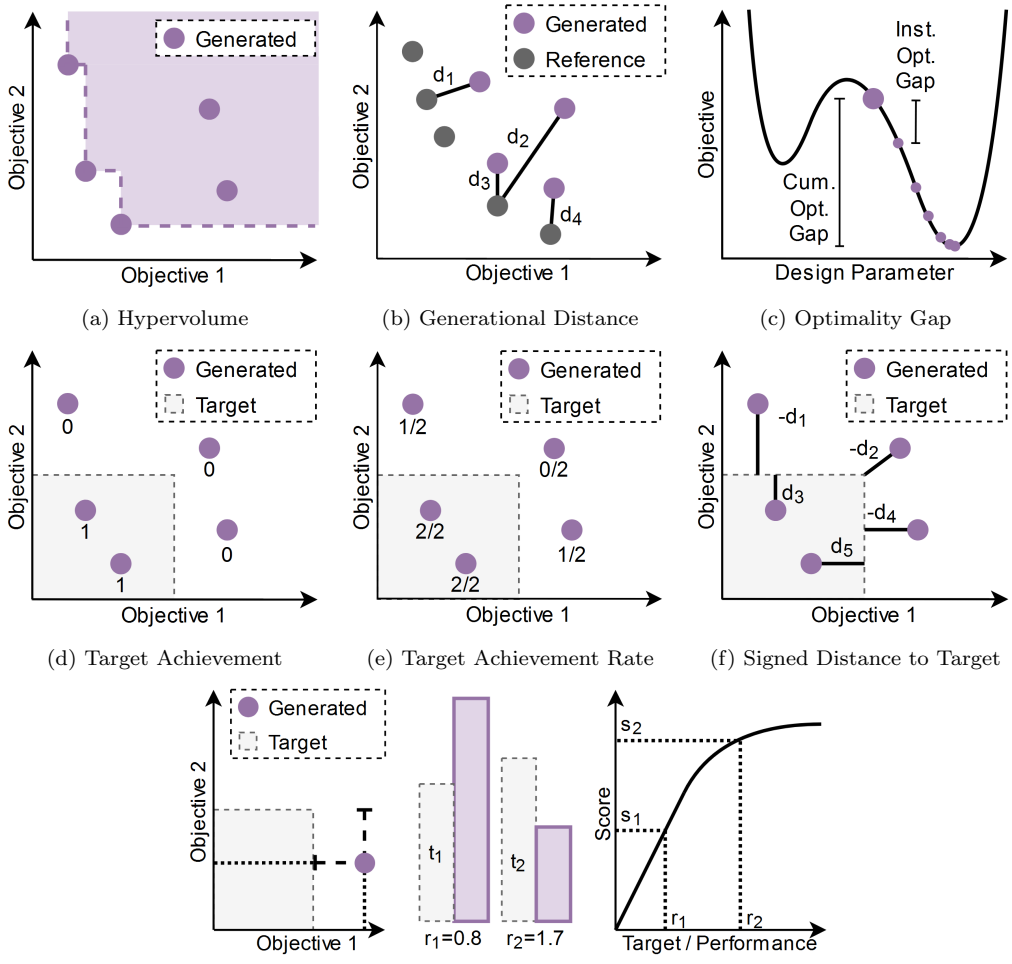
Average scores:	VAE	GAN
Constraint Satisfaction	0.713	**0.833**
Constraint Satisfaction Rate	0.857	**0.917**
Predicted Constraint Satisfaction	0.698	**0.823**
Nearest Invalid Datapoint	0.17	**0.173**

associated set of functional performance characteristics, often modeled as a mapping from the design space to some performance space. Let us consider a bicycle design problem. As demonstrated in Fig. 1 earlier, two frames that look visually similar in the pixel space still have orders of magnitude different deflection values. Quality characteristics in design may include factors such as cost, weight, efficiency, etc., and are highly problem dependent. We'd like to note that a generated design's innate performance attributes can also serve as simple but effective evaluation metrics for the design and the method that generated it.

7.1. Performance Optimality

In many engineering problems, designers want to find a distributed set of solutions that are performant across multiple objectives. This is especially common in multi-objective optimization problems, where designers often seek to identify a Pareto-optimal set of points. Pareto-optimal designs have the property that any performance improvement in some objective must come at the expense of performance in some other objective. This means that Pareto-optimal designs cannot be 'dominated' by any other design, i.e., there exists no design that has superior performance in every objective. Identifying a strong approximation set for the Pareto-optimal front can be extremely valuable in design problems because it effectively provides an optimal design given any arbitrary choice of objective priorities by the designer. If practitioners seek to generate a diverse set of near-optimal designs, they may seek to quantify how close a set of generated designs is to the true Pareto-optimal front. Below, we discuss a few metrics that could be adopted by design researchers to quantify performance optimality and refer readers to more detailed reviews, such as [13] for other metrics.

Hypervolume Metric (Figure 7a). The hypervolume metric, often simply referred to as 'hypervolume,' is a staple metric in the multi-objective optimization field [13], which estimates the proximity of a set of generated samples to the (often unknown) Pareto-optimal front. In simple terms, the metric calculates the hypervolume comprised of all points that are dominated by some point in the generated set but simultaneously dominate some fixed reference point. In any problem where a diverse near-Pareto-optimal set of samples is desired, hypervolume could be used to quantify the performance of generated samples. For example, the metric was used in [55] to compare the performance of different GAN models for airfoil synthesis.



(a) Hypervolume

(b) Generational Distance

(c) Optimality Gap

(d) Target Achievement

(e) Target Achievement Rate

(f) Signed Distance to Target

(g) Design Target Achievement Index

Figure 7: Illustrations of select design quality and target achievement metrics.

Generational Distance (Figure 7b). When a set of 'optimal' reference designs is known, generational distance could be used to measure design optimality. Generational distance, another staple of the multi-objective optimization community [13], measures the distance from a generated sample to the nearest point on the 'optimal' reference set [64]. This reference set may not consist of truly optimal designs but is typically taken as a reliable approximation for a true Pareto-optimal design set. Generational distance is not widely adopted as a metric for deep generative models but can serve as an excellent metric for evaluating the quality of generated designs in cases when an optimality frontier is known or could be constructed from the training data using non-dominated ranking methods.

Optimality Gap (Figure 7c). When working on well-defined problems, practitioners may be able perform gradient-based optimization on their problem. In these cases, they can estimate the distance to an optimal design using 'optimality gap' metrics. The distance from the generated design to the local minimum discovered by the optimizer is often referred to as the 'cumulative optimality gap' or simply the 'optimality gap.' A variant, the instantaneous optimality gap, has also been proposed to measure the distance to the modified design after the first step of gradient descent [65].

7.2. Target Achievement

While generating an entire set of Pareto-optimal designs can be helpful when exact design goals are not yet decided, practitioners may also need to apply generative models to design problems where performance targets are specified. These types of problems necessitate a suite of metrics that quantify a model's ability to achieve performance targets. We would like to clarify that, in contrast to hard design constraints, performance targets are intended to be negotiable. They are also different from soft constraints, as exceeding a target by a larger margin is often desirable, which is not the case with constraints. While many of the metrics for constraint satisfaction can be modified to quantify target achievement, the handling of design targets can call for more nuanced metrics which reflect their flexibility.

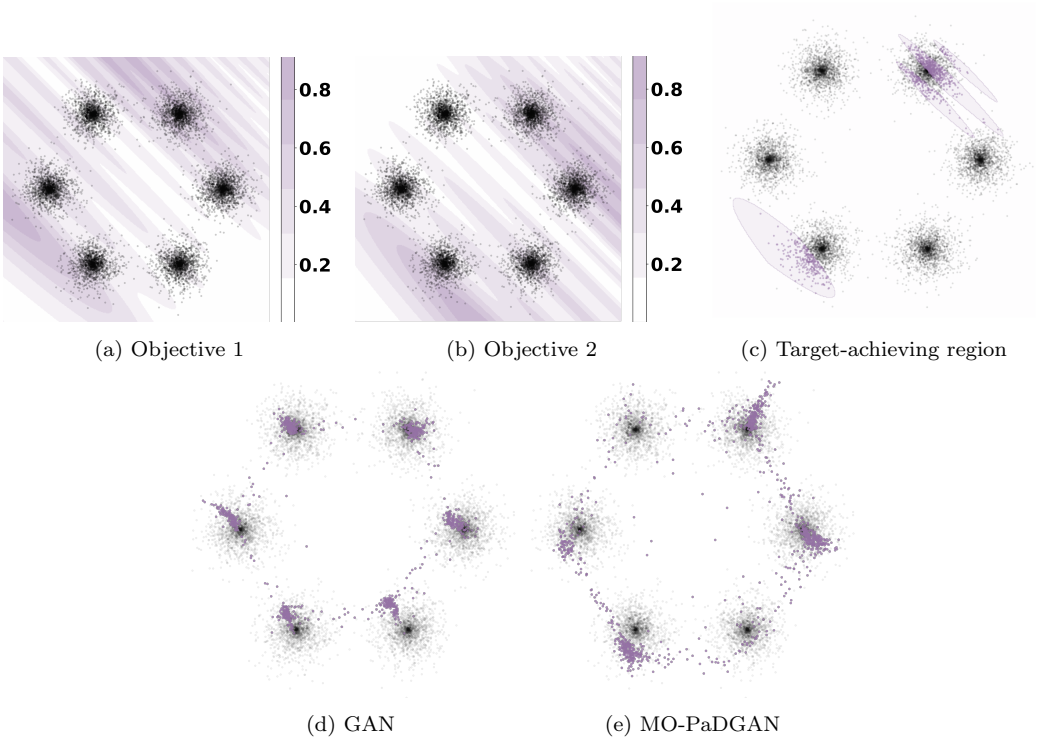
Target Achievement Scores. Reformulating several of the previously discussed constraint satisfaction metrics, we can define analogs for the target achievement case. Target achievement (Figure 7d), much like the constraint satisfaction score, measures whether a design simultaneously meets all performance

targets across objectives. However, since simultaneously achieving all the targets in a given problem may be difficult, more nuanced scores are typically more informative in quantifying proximity to the target. Target achievement rate (Figure 7e) is analogous to the constraint satisfaction rate, quantifying the weighted proportion of multi-objective design targets met by design. When practitioners have a well-defined target criterion, they can calculate a signed distance to target (Figure 7f), indicating the degree to which designs are outperforming or underperforming the set of multi-objective design targets.

Design Target Achievement Index (Figure 7g). In problems with particularly nuanced design goals and targets, practitioners may want an even more flexible metric than the signed distance to the target. Regenwetter et. al. [27] proposed the design target achievement index (DTAI), which considers the relative importance of targets and the value of continued optimization beyond the target. The key idea is to aggregate weighted soft penalties when a target in any of the objectives is not achieved and combine them with reward functions when the target is exceeded. DTAI is also bounded and differentiable, making it viable as a training loss in generative design problems where design performance values are provided in the dataset.

7.3. Demonstration of Design Quality Metrics

Having introduced a variety of performance and target achievement metrics for DGMs in design, we once again present a 2-dimensional example demonstrating the use of these metrics. We select the KNO1 test problem as the objective function [66] and add it to the 2-dimensional example discussed previously. We test a standard GAN, which ignores performance, and a Multi-Objective Performance-augmented Diverse GAN (MO-PaDGAN) [55], which attempts to generate a diverse set of high-performing designs. For target-oriented metrics, we select a fairly demanding performance target of $(0.5, 0.5)$, which is only attainable in small regions of the design space. The model and metric parameters are included in the appendix, and the results are plotted and tabulated in Figure 8 and Table 5. The MO-PaDGAN largely ignores the two low-performance modes and outperforms the standard GAN in every performance metric tested. It achieves a stronger Pareto-optimal set as indicated by the hypervolume metric. It generates designs that are, in general, closer to the known Pareto-optimal reference set, as indicated by the generational distance metric. Its designs achieve a greater fraction of



(a) Objective 1 (b) Objective 2 (c) Target-achieving region

(d) GAN (e) MO-PaDGAN

Figure 8: Visual demonstration of a Generative Adversarial Network and Multi-Objective Performance-augmented Diverse GAN on the 2-dimensional KNO1 objective. The MO-PaDGAN, which considers functional performance samples predominantly from higher-performing modes.

the performance targets, lie closer to the target boundary, and adhere more closely to the target overall, as indicated by the weighted target achievement rate, signed distance to target, and design target achievement index, respectively. These metrics match our intuition and provide a way to quantify differences between different models: A standard GAN underperforms in all metrics, as it only maximizes distribution similarity, without any consideration for other factors such as objectives and targets. Note that the GAN would dominate the MO-PaDGAN in almost every statistical similarity metric. In design problems, higher functional performance often comes at the expense of statistical similarity.

Table 5: Performance and target achievement scores for distributions in Figure 8.The MO-PaDGAN, which considers functional performance, samples predominantly from higher-performing modes. Point metrics are averaged over the generated set.

(Avg.) scores:	GAN	MO-PaDGAN
Hypervolume	0.460	**0.571**
Generational Distance	0.215	**0.315**
Weighted Target Ach. Rate	0.162	**0.430**
Signed Distance to Target	-0.275	**-0.154**
Design Target Ach. Index	0.324	**0.433**

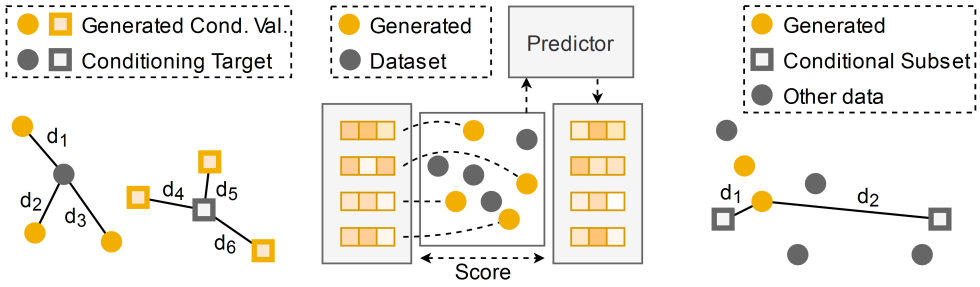
8. Evaluating Conditioning Requirements

In DGMs, conditioning refers to the process of incorporating additional information, such as labels or attributes, into the model when generating new data. For example, in image generation, a deep generative model can be conditioned on class labels, such as 'dog' or 'cat', in order to generate images of specific animals, rather than randomly picking cats or dogs (or training separate generators for each class of image). Conditional DGMs are often more robust and data efficient than training many individual DGMs since they allow designers to reuse a single model for many variants of a particular design problem. This also differs from traditional optimization approaches where a change in problem parameters usually necessitates re-optimization.

Conditional variants of many classic generative models have been proposed [67, 68, 69], but are most commonly applied to class-conditional problems. In design, many tasks necessitate continuous conditioning, and specialized models have been proposed [70, 71]. Conditioning information varies from problem to problem. It is commonly used to encode constraints, functional performance targets, or parameters used to distinguish one version of a problem from another. For example, DGMs for structural topologies are typically conditioned on boundary conditions (normally thought of as constraints), volume fraction (sometimes thought of as a functional performance attribute), and load locations (used to distinguish different loading problems) [72, 6].

8.1. Conditioning Adherence (Figure 9a)

When working with conditional models, practitioners may want to quantify the degree to which their model respects the information on which it is



(a) Conditioning Adherence (b) Conditioning Reconstruction (c) Cond. Nearest Gen. Sample

Figure 9: Illustrations of select conditioning metrics.

conditioned. In some problems, practitioners are conditioning on information that they can directly calculate for generated samples. For example, if conditioning a structural topology DGM on volume fraction, the volume fraction of generated samples can be calculated as the fraction of filled pixels to total pixels. In this case, the difference in conditioned versus actual volume fraction serves as a conditioning adherence metric. Broadly speaking, the distance between condition and recalculated condition is a viable approach to quantify conditioning adherence.

8.2. Conditioning Reconstruction (Figure 9b)

In the absence of a method to directly recalculate conditioning information from generated samples, practitioners can instead use a predictive model to reconstruct estimates of the condition. A reconstruction loss serves as the score. The accuracy of this metric relies on the accuracy of the predictor, since any prediction error incurred will factor into the score.

8.3. Adapting Unary Metrics to Conditional Problems

Many of the metrics presented in previous sections must be adapted for conditional problems. We first discuss strategies to adapt unary metrics to conditional settings and discuss the more challenging task of adapting binary metrics in the following section. Unary metrics must be calculated and averaged over numerous conditions. For a fair comparison, these conditions must be identical or equivalent when evaluating different models on a particular problem. For class-conditional problems, this may mean reporting a score for each class or reporting a weighted average across classes. For continuous conditioning problems, this may mean averaging over many conditions sampled from some repeatable parametric statistical distribution.

8.4. Adapting Binary Metrics to Conditional Problems

In unconditional problems, binary metrics typically compare some generated posterior distribution to a reference distribution (often the dataset, which we call the prior). In a conditional setting, our posterior is instead a conditional distribution. In this case, we typically have two options, each with its own strengths, which we discuss below. In either case, much like for unary metrics, we typically average scores over a sweep of conditions to yield an overall condition-agnostic score, or simply report metrics under several individual conditions.

Comparing Conditionally Posterior to a Conditional Prior. The 'standard'[10] option to evaluate binary metrics in conditional settings requires us to compare the conditional posterior distribution to a conditional reference distribution (usually the conditional prior–the subset of the dataset which meets the condition). We have illustrated the conditional nearest generated sample metric in Figure 9c, but the principle applies broadly to many binary metrics. However, identifying a conditional reference is often nontrivial, especially when the reference distribution is something other than the dataset[11]. Even when we are simply using the dataset as a reference distribution, if we have a continuous conditioning space, we can't select a discrete subset of the dataset which exactly respects the condition. Instead, we can approximate by selecting a subset of the dataset whose conditions are proximal to the condition used to create the generated distribution. However, this necessitates calculating some distance in the conditioning space, such as the vicinal loss defined in literature [70, 74]. Calculating these vicinity-based values may again be nontrivial in problems with high-dimensional or multimodal conditioning information.

Comparing Conditionally Generated Distributions to a Marginal Distribution. Alternatively, we can compare the conditional posterior distribution to the full marginal reference distribution (usually the prior). Though less intuitive, this can be the next best option when calculating a conditional reference is intractable. But in certain cases, comparing a conditional posterior

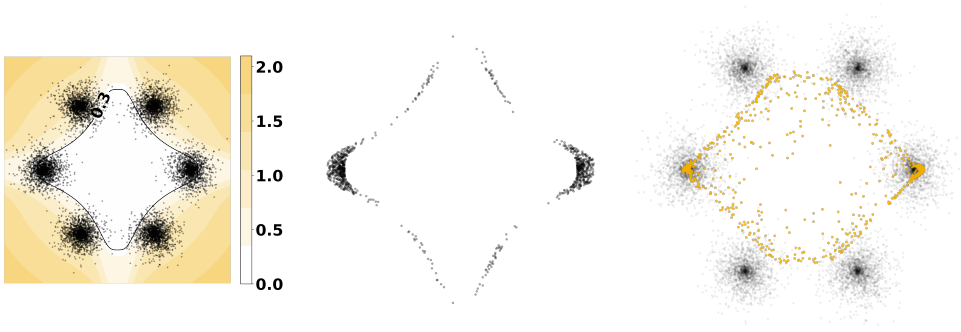
[10]This is the standard formulation for class-conditional variants of popular metrics like class-conditional FID and IS [73].

[11]For example the reference distribution in generational distance is a pareto-optimal set. The conditional subset of this refence may no longer be pareto-optimal, defeating the purpose of the metric

to a marginal prior may actually be preferable. This is particularly desirable if we want new designs that meet some specific condition but resemble designs from the dataset that don't meet this condition. Consider a DGM that was trained on a dataset of heavy mountain bikes and light road bikes and is generating bike frames conditioned on frame mass. If this DGM discovers how to generate lightweight mountain bikes, it will score poorly on similarity metrics evaluated against the conditional prior, since the only lightweight bikes in the dataset are road bikes. In contrast, this model would score well on similarity metrics evaluated against the marginal prior. The comparison to the marginal or conditional prior effectively balances a tradeoff between two possible training objectives for the model. On the one hand, lightweight mountain bikes are 'unrealistic', according to the data. On the other hand, discovering adaptations of designs to meet the specified condition may be innovative and desirable.

8.5. Demonstration of Conditioning Metrics

To demonstrate the conditioning of generative models, we construct a fairly challenging continuous conditioning problem on the previously used dataset. Each of the datapoints are labeled according to a highly nonlinear conditioning function. The goal is to generate samples that have a condition value of exactly 0.3. Though we would typically care about performance across a variety of condition values, we will only examine this particular conditioning value in this case study. In practice, we would likely perform a weighted average across the conditioning variable to get a sense of the overall conditional generational performance of the models. We train a conditional VAE and a conditional GAN, the results of which are plotted in Figure 10 and tabulated in 6. Details about the model architecture, training, and metric settings are included in the appendix. We take the conditional prior to be the 10% of the dataset that most closely matches the condition, as shown in 10b. Overall, the cGAN focuses heavily on the leftmost and rightmost modes, struggles to accurately capture other areas of the distribution, and ignores sparse areas. In contrast, the cVAE under-represents the two main modes, but much more faithfully captures the remainder of the distribution. Overall, the cVAE significantly outperforms the GAN in conditioning reconstruction and adherence, indicating that it generated samples that, on average, much more closely match the condition. In both conditional and marginal F10 and F0.1, the cVAE and cGAN repeat trends from their unconditional counterparts in Section 4.5. Due to its focus on the main modes, the GAN



(a) Original data (b) Conditional Prior (c) cVAE vs. marginal prior

(d) cVAE vs. conditional prior (e) cGAN vs. marginal prior (f) cGAN vs. conditional prior

Figure 10: Visual demonstration of a conditional Variational Autoencoder and conditional GAN on a continuous conditioning problem. Generated distributions are overlaid over both the original dataset (marginal prior), as well as the nearest 10% of the data to the condition (conditional prior).The cVAE better matches the marginal prior, while the cGAN better matches the conditional prior.

.

Table 6: Performance and target achievement scores for distributions in Figure 10. The cVAE better matches the marginal prior, while the cGAN better matches the conditional prior on this dataset. Point metrics are averaged over the generated set.

(Avg.) scores:	cVAE	cGAN
Conditioning Reconstruction	**0.007**	0.026
Conditioning Adherence	**0.008**	0.026
Conditional F10	**0.909**	0.826
Conditional F0.1	0.879	**0.960**
Conditional MMD	0.035	**0.021**
Marginal F10	**0.949**	0.933
Marginal F0.1	0.447	**0.554**
Marginal MMD	**0.068**	0.097

significantly outperforms in maximum mean discrepancy when comparing to the subset of the dataset that most closely match the condition. However, when comparing to the marginal prior, the cVAE is the clear leader in MMD, generating more samples that lie closer to the four underrepresented modes. As illustrated, binary metrics can tell very different stories when compared against conditional or marginal priors. Next, we move on to consider miscellaneous DGM evaluation metrics.

9. Other Metrics for Evaluating Deep Generative Models

Thus far, we have discussed metrics to evaluate similarity, diversity, constraints, performance, and conditioning. Below, we briefly summarize a few more types of metrics that may be important to evaluate the performance of DGMs in certain problems. A detailed, but not necessarily comprehensive list of other considerations for DGMs in design is included in the Appendix in 15.1.

9.1. Latent Disentanglement

Many DGMs synthesize designs by taking randomized inputs from some (usually multidimensional) latent variable space. Many design researchers have attempted to link latent variables with the physical properties of the generated designs [75, 76], which, ideally, would serve as a tool for human designers to manually select or tune generated designs. If the latent space is

disentangled, the generation process becomes more interpretable. Tasks like creating new functionally graded material or design space exploration could also be enhanced by disentangled representations, as moving in the latent space along certain dimensions corresponds to expected changes. Disentanglement metrics generally fall into one of three categories: Intervention-based metrics such as Z-diff [77], predictor-based metrics such as attribute predictability score (SAP) [78], and information-based metrics such as mutual information gap (MIG) [79]. We refer readers to reviews like [80] and [81] for more detailed discussions on disentanglement metrics.

9.2. Human Evaluation

Though automated evaluation metrics are often the most practical, they seldom provide as valuable of an analysis as people. Human evaluation approaches can roughly be divided into crowdsourced evaluation frameworks and expert evaluation frameworks, where the primary tradeoff is cost versus evaluation quality.

Crowdsourced Evaluation. Crowdsourced evaluation frameworks are common in computer vision fields since untrained humans typically suffice for determining the 'realism' of images in computer vision problems. Metrics like Human Eye Perceptual Evaluation (HYPE) [82], for example, quantify how easily humans can discern between real and fake samples. In contrast, designs may be difficult to properly evaluate since they are not always represented in a visual medium and may have infeasibilities or inefficiencies that require engineering expertise to discern.

Expert Evaluation. Expert evaluation methods are often used in various applications of deep generative models to assess the quality of the generated samples. Domain experts or linguists evaluate the samples based on specific criteria such as realism, coherence, and relevance. The scores are then averaged over a set of generated samples to compare different models. This approach is considered to be a gold standard for evaluating the quality of generated samples, but is time-consuming and costly.

10. Application Study: Exploratory Bicycle Frame Design with Constraints and Performance Targets

Throughout the paper, we have applied metrics to simple 2-dimensional problems to visually demonstrate how the presented metrics function. How-

ever, real design problems are typically higher-dimensional, often have more constraints and objectives, and are generally highly non-convex. As such, we feel that practical case studies on real engineering design problems may be insightful. Therefore, in this section, we present the first of two case studies. In this problem, we want to design novel, diverse, and high-performing bicycle frames that meet a set of ten structural performance targets and adhere to an unknown set of implicit design constraints. We seek a DGM that consistently generates designs meeting performance targets and constraints, but generates a wide enough variety to offer a broad selection of design candidates.

10.1. Dataset

The dataset we use [12] features roughly 4000 constraint-satisfying designs and roughly 300 constraint-violating designs. Each frame design is inspired by a real bicycle design [83] and is parameterized over 37 parameters. Each constraint-satisfying design is 'labeled' with a vector of 10 structural performance values, such as weight, safety factors, and deflections under various loading conditions. This dataset was previously used as a benchmark for target-seeking deep generative models in [27] and we adapt the models tested for our demonstration. We also adopt the objective weights and DTAI parameters from [27].

10.2. Models

For our case study, we train and test three Generative Adversarial Network (GAN) variants. The first is a 'vanilla' GAN, a type of network that adversarially trains two networks, one which generates new designs and one which discriminates between existing designs and generated designs. The vanilla GAN is blind to design performance, only implicitly observes constraints, and does not promote exploration beyond the convex region of the dataset. The second is a Multi-Objective Performance-augmented Diverse GAN (MO-PaDGAN) which uses a performance-weighted DPP kernel to augment the loss function of the GAN in an effort to simultaneously encourage higher performance and greater diversity among generated designs. MO-PaDGAN generally encourages diverse, higher-performing designs, but does not consider specific performance targets or explicit constraint handling. The third is a DTAI-GAN which further augments the MO-PaDGAN with a weighted target achievement loss and classifier guidance to avoid constraint-violating designs. In this sense, DTAI-GAN is the only model that 'explicitly'

considers constraints, while the other models 'implicitly' consider constraints by training only on constraint-satisfying designs.

10.3. Selecting Metrics

We'd like to evaluate diversity, performance, target achievement, and constraint satisfaction, alongside distribution matching. We discuss how we select metrics for this problem and justify these choices.

Statistical Similarity. While the goal is not just to mimic existing designs, measuring similarity is important to make sure that we are still generating bicycle frames. We would also ideally like our generated designs to span as much of the design space as possible, thereby representing as many key types of designs as possible. As such, we care about both 'realism' and coverage, as well as general similarity. For this, we select nearest datapoint (NDP), nearest generated sample (NGS), and F1 to capture realism, coverage, and similarity, respectively. Since we primarily care about these metrics in the design space, we evaluate all three in the design space, rather than the performance space.

Design Exploration. As one of the stated objectives of the problem, we want to generate a diverse set of novel designs. This is common in scenarios where a final design will be selected by experts from a diverse set. While we could look to maximize nearest datapoint as a novelty metric, we still want our designs to resemble designs in the dataset. We instead calculate nearest datapoint in the performance space, instead of the design space. Put plainly, while we want generated designs to resemble existing designs, we would like them to have different performance values. Since we also want generated designs to be diverse, we select inter-sample distance and DPP diversity as two diversity metrics of choice, which we evaluate in the design space.

Design Constraints. The FRAMED problem provides a method to evaluate constraint adherence for generated samples, making the constraint satisfaction metric a natural choice. Since closed-form constraint definitions are not available, more informative choices such as signed distance to constraint boundary are not feasible in this problem.

Design Quality and Target Achievement. In FRAMED, we do not have a known set of Pareto-optimal designs. While we could infer an approximate set from the dataset, we have no basis for assuming that dataset designs are

near-optimal, in part due to a randomization step in the FRAMED dataset generation pipeline [12]. Therefore, we can feasibly expect to generate designs that dominate the previously non-dominated designs in the dataset, making generational distance a poor choice. We also do not have a differentiable solver which we could query to calculate the optimality gap. As such, we use hypervolume to capture the overall optimality of generated designs. Since we are given performance targets with priority weights, we use a weighted target achievement rate as a simple metric, but we can also use signed distance to target since we have access to simple closed-form target criteria. Finally, we use the design target achievement index (DTAI) to evaluate overall target achievement performance.

10.4. Results

We demonstrate the performance of the three DGMs and their scores on selected metrics in Figure 11.

Statistical Similarity. Across the board, the vanilla GAN achieved superior distribution-matching performance to the GAN variants focused on design performance. As indicated by the demonstrated Principal Component Analysis embeddings, MO-PaDGAN identified a higher-performing region of the design space and consistently generated many designs away from the original design distribution. Similarly, DTAI-GAN identified a region of the design space where designs were likely to meet performance targets and sampled heavily from this region. Since there were a handful of dataset samples near this region, DTAI-GAN outperformed MO-PaDGAN in nearest datapoint. However, since it very consistently generated designs far away from the center of the dataset distribution, it scored much worse in nearest generated sample. Both models achieved very low F1 scores. Though the distribution-matching scores indicate that the performance-aware models deviated significantly from the dataset, we don't necessarily find this concerning since we primarily care about finding novel high-performing designs.

Design Exploration. The DTAI-GAN scored highest in performance-space novelty, generating many designs whose performance differs greatly from those in the dataset. These models also achieved much higher diversity exploring more of the space and generating more varied samples.

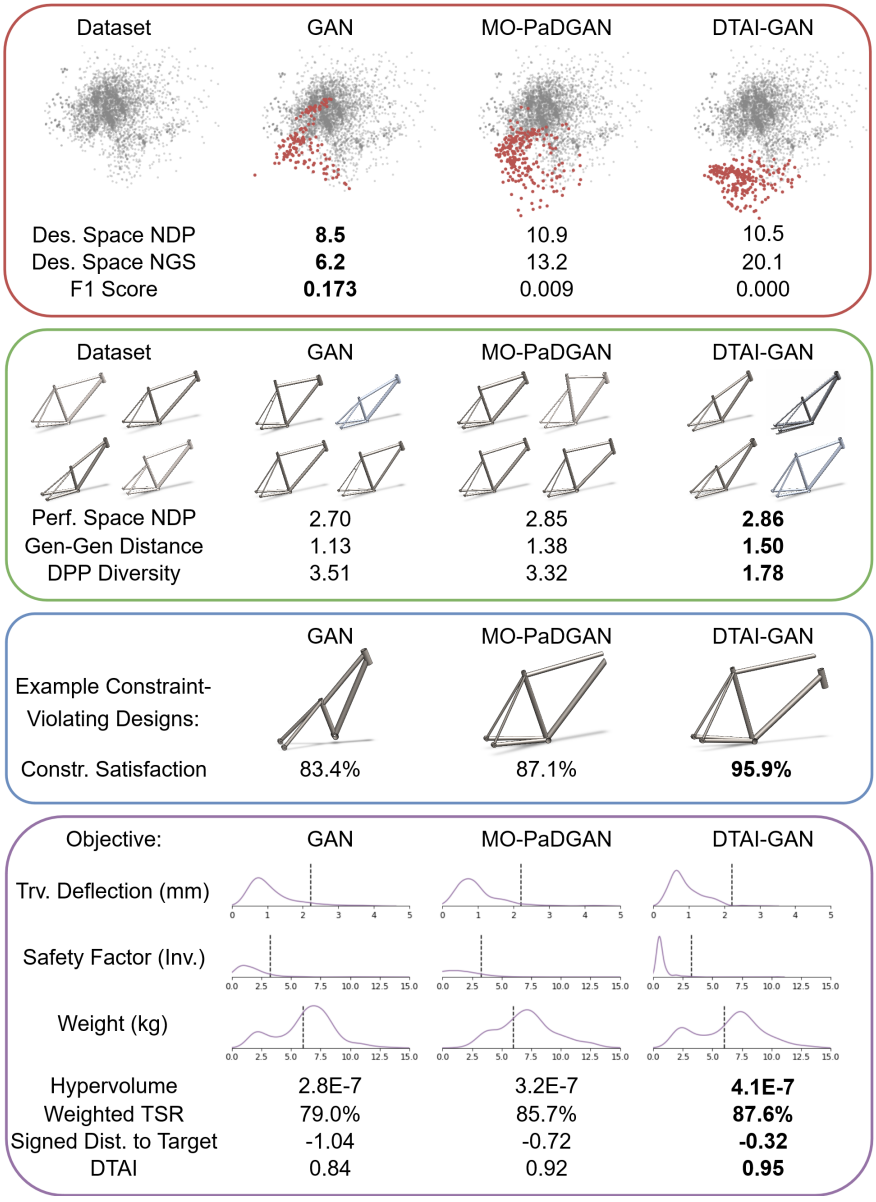


Figure 11: Evaluation of three models on FRAMED dataset. The first box, focused on similarity shows a 2-D Principal Component Analysis embedding of generated designs overlaid over the dataset. The second box, focused on diversity shows select generated bike frames. The third box, focused on constraint satisfaction shows select invalid bike frames. The last box, focused on functional performance shows kernel density plots of structural performance scores of generated bikes (design target shown as the dotted line; smaller is better).

Design Constraints. Since we don't have access to closed-form constraint tests, we are not able to evaluate distance to constraint boundary or other more insightful constraint-related metrics. Instead, we simply know that, as expected, the constraint-aware DTAI-GAN is a significantly stronger performer at constraint satisfaction than the two models that only implicitly considered constraints.

Design Quality and Target Achievement. Across all performance and target achievement metrics, DTAI-GAN scores the highest, and the vanilla GAN scores the lowest. Kernel density plots for three of the ten performance objectives are shown in the last panel of 11, and in each, DTAI-GAN's distribution is most favorable. Interestingly, the DTAI-GAN's average target success rate was barely higher than MO-PaDGAN's. However, its average signed distance to the target was much higher, indicating that DTAI-GAN's designs that exceeded the target did so more drastically and designs that missed the target did so by a smaller margin.

10.5. Analysis

The metrics strongly demonstrate that the DTAI-GAN achieves its stated goal of generating a diverse and novel set of designs that achievement performance targets and satisfy constraints. The enhanced functional performance and constraint satisfaction of generated designs detracted from the DTAI-GAN's ability to match the training dataset. However, this behavior was largely expected and encouraged in order to discover higher-performing regions of the design space than were previously represented in the dataset. We now move on to our second case study, exploring optimal topology generation.

11. Application Study: Optimal Topology Generation using Conditional Deep Generative Models

In this section, we demonstrate the appropriate selection of metrics for structural topology generation problems. Recent work [72, 84, 6, 85] in the field has focused on training DGMs to circumvent the reliable but slower Topology Optimization (TO) solvers, such as Solid Isotropic Material with Penalization (SIMP) [86, 87]. These DGMs train on a dataset of topologies generated by SIMP, typically taking volume fraction and loading information as conditioning. Then, for various loading cases and volume fractions,

they generate topologies that they predict SIMP would generate. Typically, the single functional performance objective is to minimize compliance of the generated topologies.

11.1. Metric Selection

We first discuss existing metrics commonly used for evaluating DGMs in optimal topology generation, then propose several additional metrics that may be valuable.

Existing Metrics. Mazé & Ahmed [6] propose a set of four evaluation metrics for DGMs in optimal topology generation which have been adopted in later works [85]: Compliance error, volume fraction error, load violation, and floating material. We break down what these metrics mean, and which metrics from this paper they correspond to:

1. Compliance Error: Compliance error describes the percent difference between a generated topology's compliance and the compliance of a SIMP-generated topology under the same loading condition. This metric is a variant of the signed distance to target metric (Sec. 10.3). Compliance is treated as a functional performance objective in TO. For each conditional input, a target compliance value is selected to be the compliance of the topology generated by SIMP. The compliance error is then simply given as the normalized signed distance to this target. Specifically, outperforming SIMP is rewarded with a negative compliance error, while exceeding the target compliance is assigned a positive compliance error.
2. Volume Fraction Error: Volume fraction error quantifies the percent error between a generated topology's volume fraction and the target volume fraction given to the generator. This metric is conditioning adherence (Sec. 8.1) since volume fraction is provided as a conditional input to the model and calculated for generated topologies. The error is calculated as the ratio of the difference to the conditional input.
3. Load violation and floating material: Load violation and floating material quantify whether the generated topology has material at the point of load application, and disconnected material, respectively. These are constraint satisfaction scores.

All in all, this is a strong set of evaluation metrics capturing functional performance, constraint satisfaction, and conditioning. We note that each of

these metrics is a point metric and can be summarized across a generated sample set as practitioners see fit. Papers commonly report mean, median, or proportion above some threshold.

11.2. Other Metrics

Though the previously discussed metrics quantify many important facets of DGM performance in optimal topology generation, we feel that several other metrics could also be informative. Specifically, we propose three additional metrics focused on diversity, novelty, and more nuanced constraint satisfaction scoring.

1. Proportion of Floating Material: While load violation is a binary constraint, floating material can be quantified using a scalar measuring proportion of pixels that are disconnected from the topology. This continuous score may be more informative than a binary metric. Since the constraint boundary is at zero, this proportion is simply the distance to constraint boundary metric.
2. Distance to SIMP: Quantifying the novelty of generated topologies relative to their SIMP-generated counterparts can yield insight as to whether the model is memorizing SIMP-generated solutions or learning general structural optimization skills which may yield different results from SIMP. Models which are able to generate novel topologies may be able to seed SIMP to find superior solutions[12] (or in rare cases find stronger solutions outright) even if their average compliance error is relatively high. In contrast, a model that mimics SIMP will seldom be able to find superior topologies even if it has perfect compliance error. In many TO datasets, the only datapoint available for each discrete condition is the SIMP-generated solution, meaning that distance to SIMP is simply the conditional nearest datapoint metric. To calculate this score, we need a distance metric that works on topologies. As discussed earlier, embedding models trained on natural images may not offer meaningful distances, so we instead use pixel-wise distance, though Structural Similarity Index Metric [88] could be a valid choice.
3. Topology Diversity: Just as we may like to understand how novel our DGM-generated topologies are from SIMP's topologies, we may seek to

[12] A known limitation of SIMP is its susceptibility to local minima caused by nonconvexity.

quantify how diverse these topologies are with respect to one another. This can tell us whether our model is capable of finding several families of strong solutions, increasing the likelihood that a strong design candidate is present in a large sample. We can use pixel distance to calculate a diversity matrix over a batch of samples generated under a single condition. We can then calculate DPP diversity as a score for the diversity of the generator.

11.3. Models

We benchmark two versions of the state-of-the-art DGM for topology optimization, TopoDiff [6]. TopoDiff is a Denoising Diffusion Probabilistic Model (DDPM) that trains using a similarity objective to mimic topologies generated by SIMP. Simultaneously, it uses classifier guidance during sampling to 'push' otherwise invalid topologies away across constraint boundaries and into feasible regions. TopoDiff is conditioned on 2D stress field images that indicate the loading condition and assist in its training. Classically, these stress fields are calculated using Finite Element Analysis (FEA) to simulate stresses when the loads are applied to a solid material block. However, running FEA to generate stress fields on every input problem is time-consuming, so we train an alternative version of TopoDiff conditioned on kernel-based stress field estimates, as proposed in [85].

11.4. Results

The two variants of TopoDiff are benchmarked on the four original metrics from the paper, as well as the additional three that we propose. Models are tested on the first 300 conditions in TopoDiff's level 1 test data. We diffuse all samples over 100 steps. Unlike [85] and [6], we generate 10 samples per condition per model instead of one, in order to score diversity. This adds additional complexity to the averaging process. Both point and set metrics must both be averaged over conditions but point metrics must additionally be averaged over samples generated for a single condition. The scores are tabulated in Table 7. The reported averaging technique for point metrics ('mean' or 'median') is used to average over samples for a single condition, while mean averaging is used for all metrics (both point and set) over all conditions. Reported results are expected to differ slightly from [85] and [6] since different instantiations of models were tested and only a subset of the test set was evaluated. Inference time, as reported in [85], is also

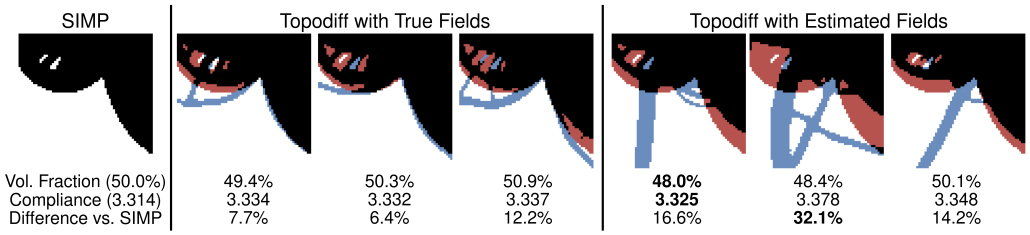


Figure 12: Sample topologies generated by two versions of TopoDiff compared against SIMP-generated topology. Three samples are generated using each model. Additional material is shown in blue while removed material is shown in red (relative to SIMP). SIMP's compliance and volume fraction are shown in parentheses. TopoDiff tends to generate more novel and diverse results when conditioned on kernel estimates of the stress fields. In this test problem, it generates the lightest and least compliant topology, all while being fairly novel relative to SIMP.

included [13]. For qualitative examination, Figure 12 demonstrates several generated topologies using both approaches, emphasizing the difference between the generated topology and the SIMP-generated solution.

11.5. Discussion

Since the FEA-generated strain fields are more accurate and informative than the kernel-estimated strain fields, we generally expect the FEA-generated fields to yield superior results. Indeed, TopoDiff demonstrates superior compliance error (target achievement), constraint achievement (load validity, floating material), and volume fraction error (conditioning adherence) when conditioned on the FEA fields. Additionally, as the new floating material quantity metric indicates, generated designs are also closer to being constraint-satisfying when TopoDiff is conditioned on FEA fields.

In addition to vastly improved inference time, a notable strength of the variant trained on the kernel-estimated strain fields is the higher novelty versus SIMP-generated results and the higher diversity among generated topologies. This difference in inference time is emphasized in [85], however, the authors don't showcase the vastly improved diversity and novelty benefits of their proposed approach to condition TopoDiff on kernel estimates of the stress fields. As we illustrate with auxiliary metrics, the kernel-estimated fields also result in greater novelty and diversity. In particular, the solutions

[13] We do not recalculate inference time and instead use published values from [85].

Table 7: Scores of two variants of TopoDiff. While the variant conditioned on true stress fields demonstrates superior target achievement, constraint satisfaction, and conditioning adherence, the variant conditioned on estimated fields exhibits higher novelty and diversity of generated solutions, in addition to being significantly faster during inference.

Metric	Topodiff - **FEA Fields**	Topodiff - **Kernel Fields**
Median Compliance Error (Distance to Target Boundary)	**1.99%**	9.61%
Mean Volume Fraction Error (Conditioning Adherence)	**1.49%**	1.69%
Mean Load Validity (Binary Constraint Satisfaction 1)	**0.00%**	0.13%
Mean Floating Material (Binary Constraint Satisfaction 2)	**6.60**%	8.97%
Mean Floating Material Quantity (Distance to Constraint Boundary 2)	**0.022%**	0.031%
Mean Distance to SIMP (Mean Novelty)	11.6%	**17.3%**
Topology Diversity (Diversity)	7.144	**5.87**
Mean Inference Time (sec.)	5.87	**2.35**

generated differ from the SIMP solutions by almost 50% more pixels than samples from the FEA field-conditioned model. Coupled with the significantly higher diversity, they have a much higher chance of re-seeding SIMP to find a stronger solution. This largely explains the significant performance improvements found by [85], using generated samples to seed further re-optimization in SIMP. The kernel-estimated fields may generate not only faster samples but better samples for this task compared to the FEA fields.

Through the selection of new metrics for DGMs in optimal topology generation, we have enabled insightful comparisons between models regarding the novelty and diversity of the samples they generate. These comparisons are highly relevant for many practical tasks, such as re-seeding TO solvers like SIMP to find better local optima. With this, we conclude our case study and proceed to some final discussion on implications for engineering design.

12. Implications for Engineering Design

Since the introduction of deep generative models in engineering design research, the overwhelming majority of proposed models have been trained

exclusively to optimize for statistical similarity [3]. Since most of these models were adopted from natural image synthesis problems, it's no surprise that design researchers have similarly adopted statistical similarity, the commonplace class of metrics in the image synthesis domain. DGMs have shown immense promise in numerous design fields, such as topology optimization, airfoil design, photonic/phononic materials, molecular design, metamaterials, and many more [3]. However, as design researchers continue to adapt and enhance these models for design tasks, a similar need for modifications and new additions to evaluation metrics grows more pressing by the day.

12.1. The Pitfall of Statistical Similarity

Early in this paper, we attempted to illustrate why relying solely upon statistical similarity as a design evaluation metric can be misguided. In short, statistical similarity is simply insufficient for evaluating most designs, since it ignores design constraints, novelty, diversity, and performance, which are often primary concerns in a design setting. This limitation has been raised as a primary critique of DGMs in engineering. For example, Woldseth *et al.* [89] demonstrate how DGMs applied to topology optimization fail in very simple cases when neglecting to account for performance or constraints due to the massive performance difference between some statistically similar designs. Furthermore, over-optimizing for statistical similarity can even be counterproductive and restrict the ability of a model to discover truly useful designs that exhibit superior properties to designs in the dataset. Since statistical similarity is still an important facet of DGM performance, we do not advocate for universal abstinence from similarity metrics. Instead, we encourage researchers to explore the nuance of similarity and think critically about their choice of metrics. Simultaneously, we hope that design researchers will look beyond statistical similarity to consider the wealth of other metrics at their disposal.

12.2. Selecting Appropriate Metrics

What should practitioners consider beyond similarity? This typically requires a careful examination of the design problem at hand and depends on how practitioners seek to expand upon existing designs in the dataset. Some common goals are a greater level of diversity in design candidates, a set of designs that meet a challenging set of constraints, or a collection of designs that attain superior performance attributes to existing designs. We will briefly discuss several of the popular application areas and which types

of metrics may be relevant for these problems. However, the exact choice of metrics often depends on the representation scheme, the use case, the cost of evaluation of metrics, and the DGMs used.

Molecular Design & Drug Discovery. Molecular design problems typically require that designs adhere to a rigorous set of design constraints based on fundamental chemistry principles. Constraint satisfaction is, therefore, central to molecular design. Nonetheless, diversity and functional performance are typically important as well, as many practitioners seek to explore a wide range of designs that achieve desirable functional properties [26]. As the goal is often "discovery" of something useful and new, it is imperative to consider metrics related to novelty and performance. Not surprisingly, the "rediscovery" metric, was also reported in molecule design literature to assess models.

Topology Optimization, Structural Design, and Metamaterials. As we highlighted in the case study in Section 11, TO and other related structural design problems tend to feature functional performance goals, constraints, and conditioning, which should each be quantified when evaluating DGMs. For image representation of topologies, structures, or metamaterials, pixel similarity is often used as the loss function. Researchers have shown that adding goals of performance (e.g., compliance) and constraints (e.g., manufacturability) significantly improves the usefulness of the generated designs. Diversity metrics could also be useful to discover new structures, including the problems where DGM-generated topologies are used to warm-start classic topology optimization methods.

Material and Microstructure Design. Generative models are often used to generate datasets of synthetic microstructure images to assist in microstructure characterization and reconstruction tasks [90]. This can, in turn, help establish process-structure-property links, which enable a variety of benefits, including inverse design of processes to attain desired material properties. Since realism is a primary goal in synthetic dataset generation, similarity metrics are typically favored. However, popular metrics such as FID and IS may incur heavy bias due to the domain gap between natural image datasets and microstructure scans.

Product Design and Inverse Product Design. Though product design is a fairly abstract categorization, inverse design involves 'reverse engineering' a

design given a set of desired properties and characteristics. Since many inverse product design problems feature functional performance goals, both target achievement and general functional performance metrics are usually essential. Additionally, inverse design problems lend themselves to conditional models, making conditioning metrics particularly important. Creativity tends to be an important consideration in product design. It is often defined as a combination of novelty and quality or performance of design. As a result, novelty metrics are used in many of the real-world product design problems tackled by DGMs in the current literature [91, 92, 27].

12.3. Responsible use of Evaluation Metrics

Having access to a wealth of evaluation metrics provides researchers with a great deal of flexibility in showcasing the performance of their models. However, in such a young field with few established common practices, this flexibility may also cause confusion. For the good of the community, we encourage researchers to leverage as many relevant metrics as possible to showcase both the strengths and weaknesses of their models and provide a more comprehensive picture. Finally, we encourage researchers to be diligent in publishing any hyperparameters or evaluation metric settings and share their training datasets. To properly compare models and establish clear progression in the field, evaluation metric settings and training datasets must be fully transparent in order to be replicated.

Finally, though this paper has focused on presenting metrics for the effective evaluation of DGMs, some of the proposed metrics can also be incorporated into the training of DGMs with little to no modification, as with the design target achievement index in [27] and DPP diversity metric in [55]. When directly used as training objectives, validating performance using other metrics is important to ensure the model hasn't exploited its objectives.

13. Conclusion

As deep generative models continue to expand their reach in engineering design communities, the increasing need for effective and standardized metrics grows more pronounced. This paper explored evaluation metrics for deep generative models in engineering design and presented a curated set of design-focused metrics to address this growing demand. We presented a wide variety of evaluation metrics focusing on realism, coverage, novelty, constraint satisfaction, performance, target achievement, and conditioning. We discussed

which metrics to select when evaluating different facets of model performance and demonstrated the application of these metrics on easy-to-visualize problems. Finally, we presented a practical application of the methods discussed on a challenging bicycle frame design problem. We detailed how and why we selected appropriate metrics and described the performance of three deep generative models applied to the problem.

In writing this paper, our overarching goal has been to call attention to the fallacies of statistical similarity metrics in engineering design problems, inspire practitioners to consider the many alternatives that we discuss and provide them with the requisite knowledge to effectively apply these metrics to their design problems. We publicly release our datasets, models, and implementation code for the metrics used in our case studies at `decode.mit.edu/projects/metrics/` in order to facilitate easy adoption in different domains. We sincerely hope that researchers will use these resources to better understand their models and greatly improve their capabilities in engineering design tasks.

14. Acknowledgements

The authors would like to acknowledge the MIT-IBM Watson AI Lab in supporting this research. They would also like to acknowledge Giorgio Giannone for his assistance with TopoDiff training.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, MIT Press, Cambridge, MA, USA, 2014, p. 2672–2680.

[2] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013).

[3] L. Regenwetter, A. H. Nobari, F. Ahmed, Deep generative models in engineering design: A review, Journal of Mechanical Design 144 (7) (2022) 071704.

[4] F.-A. Croitoru, V. Hondru, R. T. Ionescu, M. Shah, Diffusion models in vision: A survey, arXiv preprint arXiv:2209.04747 (2022).

[5] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, Y. Shao, W. Zhang, B. Cui, M.-H. Yang, Diffusion models: A comprehensive survey of methods and applications, arXiv preprint arXiv:2209.00796 (2022).

[6] F. Mazé, F. Ahmed, Topodiff: A performance and constraint-guided diffusion model for topology optimization, arXiv preprint arXiv:2208.09591 (2022).

[7] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[8] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, et al., Exploring the limits of transfer learning with a unified text-to-text transformer., J. Mach. Learn. Res. 21 (140) (2020) 1–67.

[9] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.

[10] Q. Zhu, J. Luo, Generative transformers for design concept generation, Journal of Computing and Information Science in Engineering 23 (4) (2023) 041003.

[11] L. Siddharth, L. Blessing, J. Luo, Natural language processing in-and-for design research, Design Science 8 (2022) e21.

[12] L. Regenwetter, C. Weaver, F. Ahmed, Framed: An automl approach for structural performance prediction of bicycle frames, Computer-Aided Design (2022) 103446.

[13] N. Riquelme, C. Von Lücken, B. Baran, Performance metrics in multi-objective optimization, in: 2015 Latin American Computing Conference (CLEI), 2015, pp. 1–11. doi:10.1109/CLEI.2015.7360024.

[14] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: 2019 IEEE/CVF Conference on

Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4396–4405. doi:10.1109/CVPR.2019.00453.

[15] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, T. Aila, Analyzing and improving the image quality of stylegan, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 8107–8116. doi:10.1109/CVPR42600.2020.00813.

[16] A. Brock, J. Donahue, K. Simonyan, Large scale gan training for high fidelity natural image synthesis, arXiv preprint arXiv:1809.11096 (2018).

[17] OpenAI, Gpt-4 technical report (2023). arXiv:2303.08774.

[18] A. Borji, Pros and cons of gan evaluation measures, Computer Vision and Image Understanding 179 (2019) 41–65.

[19] A. Borji, Pros and cons of gan evaluation measures: New developments, Computer Vision and Image Understanding 215 (2022) 103329.

[20] A. Gatt, E. Krahmer, Survey of the state of the art in natural language generation: Core tasks, applications and evaluation, Journal of Artificial Intelligence Research 61 (2018) 65–170.

[21] C. Dong, Y. Li, H. Gong, M. Chen, J. Li, Y. Shen, M. Yang, A survey of natural language generation, ACM Computing Surveys 55 (8) (2022) 1–38.

[22] D. Shah, A. Suresh, A. Admasu, D. Upadhyay, K. Deb, A survey on evaluation metrics for synthetic material micro-structure images from generative models, arXiv preprint arXiv:2211.09727 (2022).

[23] C. T. Mueller, J. A. Ochsendorf, Combining structural performance and designer preferences in evolutionary design space exploration, Automation in Construction 52 (2015) 70–82.

[24] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE transactions on evolutionary computation 6 (2) (2002) 182–197.

[25] W. Chen, F. Ahmed, Padgan: Learning to generate high-quality novel designs, Journal of Mechanical Design 143 (3) (2021) 031703.

[26] C. Bilodeau, W. Jin, T. Jaakkola, R. Barzilay, K. F. Jensen, Generative models for molecular discovery: Recent advances and challenges, Wiley Interdisciplinary Reviews: Computational Molecular Science (2022) e1608.

[27] L. Regenwetter, F. Ahmed, Design target achievement index: A differentiable metric to enhance deep generative models in multi-objective inverse design, arXiv preprint arXiv:2205.03005 (2022).

[28] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, Gans trained by a two time-scale update rule converge to a local nash equilibrium, Advances in neural information processing systems 30 (2017).

[29] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training gans, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, Curran Associates Inc., Red Hook, NY, USA, 2016, p. 2234–2242.

[30] M. Bińkowski, D. J. Sutherland, M. Arbel, A. Gretton, Demystifying mmd gans, arXiv preprint arXiv:1801.01401 (2018).

[31] C.-Y. Lin, Rouge: A package for automatic evaluation of summaries, in: Text summarization branches out, 2004, pp. 74–81.

[32] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 311–318.

[33] S. Banerjee, A. Lavie, Meteor: An automatic metric for mt evaluation with improved correlation with human judgments, in: Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, 2005, pp. 65–72.

[34] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, JMLR.org, 2017, p. 214–223.

[35] M. Arjovsky, L. Bottou, Towards principled methods for training generative adversarial networks, arXiv preprint arXiv:1701.04862 (2017).

[36] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models (2020). arXiv:2006.11239.

[37] P. Dhariwal, A. Nichol, Diffusion models beat gans on image synthesis (2021). arXiv:2105.05233.

[38] Z. Goldfeld, K. Greenewald, Sliced mutual information: A scalable measure of statistical dependence, Advances in Neural Information Processing Systems 34 (2021) 17567–17578.

[39] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, R. D. Hjelm, Mine: mutual information neural estimation, arXiv preprint arXiv:1801.04062 (2018).

[40] B. Rhodes, K. Xu, M. U. Gutmann, Telescoping density-ratio estimation, Advances in neural information processing systems 33 (2020) 4905–4916.

[41] M. Sugiyama, T. Suzuki, T. Kanamori, Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation, Annals of the Institute of Statistical Mathematics 64 (5) (2012) 1009–1044.

[42] M. Gutmann, A. Hyvärinen, Noise-contrastive estimation: A new estimation principle for unnormalized statistical models, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 297–304.

[43] J. Alsing, T. Charnock, S. Feeney, B. Wandelt, Fast likelihood-free cosmology with neural density estimators and active learning, Monthly Notices of the Royal Astronomical Society 488 (3) (2019) 4440–4458.

[44] D. P. Kingma, P. Dhariwal, Glow: Generative flow with invertible 1x1 convolutions, Advances in neural information processing systems 31 (2018).

[45] I. Kobyzev, S. J. Prince, M. A. Brubaker, Normalizing flows: An introduction and review of current methods, IEEE transactions on pattern analysis and machine intelligence 43 (11) (2020) 3964–3979.

[46] I. Deshpande, Y.-T. Hu, R. Sun, A. Pyrros, N. Siddiqui, S. Koyejo, Z. Zhao, D. Forsyth, A. G. Schwing, Max-sliced wasserstein distance and its use for gans, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 10648–10656.

[47] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[48] S. Barratt, R. Sharma, A note on the inception score, arXiv preprint arXiv:1801.01973 (2018).

[49] M. Rosca, B. Lakshminarayanan, D. Warde-Farley, S. Mohamed, Variational approaches for auto-encoding generative adversarial networks, arXiv preprint arXiv:1706.04987 (2017).

[50] M. Lucic, K. Kurach, M. Michalski, S. Gelly, O. Bousquet, Are gans created equal? a large-scale study, Advances in neural information processing systems 31 (2018).

[51] M. S. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, S. Gelly, Assessing generative models via precision and recall, Advances in neural information processing systems 31 (2018).

[52] L. Simon, R. Webster, J. Rabin, Revisiting precision recall definition for generative modeling, in: International Conference on Machine Learning, PMLR, 2019, pp. 5799–5808.

[53] S. Ravuri, O. Vinyals, Classification accuracy score for conditional generative models, Advances in neural information processing systems 32 (2019).

[54] L. Xu, M. Skoularidou, A. Cuesta-Infante, K. Veeramachaneni, Modeling tabular data using conditional gan, Advances in Neural Information Processing Systems 32 (2019).

[55] W. Chen, F. Ahmed, Mo-padgan: Reparameterizing engineering designs for augmented multi-objective optimization, Applied Soft Computing 113 (2021) 107909.

[56] M. A. Boden, Computer models of creativity, AI Magazine 30 (3) (2009) 23–23.

[57] J. Lehman, K. O. Stanley, Abandoning objectives: Evolution through the search for novelty alone, Evolutionary computation 19 (2) (2011) 189–223.

[58] N. C. Brown, C. T. Mueller, Quantifying diversity in parametric design: a comparison of possible metrics, AI EDAM 33 (1) (2019) 40–53.

[59] F. Ahmed, S. K. Ramachandran, M. Fuge, S. Hunter, S. Miller, Measuring and optimizing design variety using herfindahl index, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 59278, American Society of Mechanical Engineers, 2019, p. V007T06A007.

[60] L. F. Kozachenko, N. N. Leonenko, Sample estimate of the entropy of a random vector, Problemy Peredachi Informatsii 23 (2) (1987) 9–16.

[61] S. Pavoine, S. Ollier, D. Pontier, Measuring diversity from dissimilarities with rao's quadratic entropy: are any dissimilarities suitable?, Theoretical population biology 67 (4) (2005) 231–239.

[62] J. Podani, Convex hulls, habitat filtering, and functional diversity: mathematical elegance versus ecological interpretability, Community Ecology 10 (2) (2009) 244–250.

[63] A. Kulesza, B. Taskar, et al., Determinantal point processes for machine learning, Foundations and Trends® in Machine Learning 5 (2–3) (2012) 123–286.

[64] D. A. Van Veldhuizen, Multiobjective evolutionary algorithms: classifications, analyses, and new innovations, Air Force Institute of Technology, 1999.

[65] Q. Chen, J. Wang, P. Pope, M. Fuge, et al., Inverse design of two-dimensional airfoils using conditional generative models and surrogate log-likelihoods, Journal of Mechanical Design 144 (2) (2022).

[66] J. Knowles, Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems, IEEE Transactions on Evolutionary Computation 10 (1) (2006) 50–66.

[67] M. Mirza, S. Osindero, Conditional generative adversarial nets, arXiv preprint arXiv:1411.1784 (2014).

[68] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel, Infogan: Interpretable representation learning by information maximizing generative adversarial nets, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 2180–2188.

[69] K. Sohn, X. Yan, H. Lee, Learning structured output representation using deep conditional generative models, in: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2, NIPS'15, MIT Press, Cambridge, MA, USA, 2015, p. 3483–3491.

[70] A. Heyrani Nobari, W. Chen, F. Ahmed, Pcdgan: A continuous conditional diverse generative adversarial network for inverse design, Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (Aug 2021). doi:10.1145/3447548.3467414.
URL http://dx.doi.org/10.1145/3447548.3467414

[71] A. Heyrani Nobari, W. W. Chen, F. Ahmed, RANGE-GAN: Design Synthesis Under Constraints Using Conditional Generative Adversarial Networks, Journal of Mechanical Design (2021) 1–16arXiv: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/doi/10.1115/1.4052442/6756741/md-21-1243.pdf, doi:10.1115/1.4052442.
URL https://doi.org/10.1115/1.4052442

[72] Z. Nie, T. Lin, H. Jiang, L. B. Kara, Topologygan: Topology optimization using generative adversarial networks based on physical fields over the initial domain, Journal of Mechanical Design 143 (3) (2021) 031715.

[73] Y. Benny, T. Galanti, S. Benaim, L. Wolf, Evaluation metrics for conditional image generation, International Journal of Computer Vision 129 (5) (2021) 1712–1731.

[74] X. Ding, Y. Wang, Z. Xu, W. J. Welch, Z. J. Wang, Ccgan: Continuous conditional generative adversarial networks for image generation, in: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021, OpenReview.net, 2021.
URL https://openreview.net/forum?id=PrzjugOsDeE

[75] L. Wang, Y.-C. Chan, F. Ahmed, Z. Liu, P. Zhu, W. Chen, Deep generative modeling for mechanistic-based learning and design of metamaterial systems, Computer Methods in Applied Mechanics and Engineering 372 (2020) 113377.

[76] W. Chen, M. Fuge, Synthesizing designs with interpart dependencies using hierarchical generative adversarial networks, Journal of Mechanical Design 141 (11) (2019) 111403.

[77] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, beta-vae: Learning basic visual concepts with a constrained variational framework, in: International conference on learning representations, 2017.

[78] A. Kumar, P. Sattigeri, A. Balakrishnan, Variational inference of disentangled latent concepts from unlabeled observations, arXiv preprint arXiv:1711.00848 (2017).

[79] R. T. Chen, X. Li, R. B. Grosse, D. K. Duvenaud, Isolating sources of disentanglement in variational autoencoders, Advances in neural information processing systems 31 (2018).

[80] J. Zaidi, J. Boilard, G. Gagnon, M.-A. Carbonneau, Measuring disentanglement: A review of metrics, arXiv preprint arXiv:2012.09276 (2020).

[81] K. Do, T. Tran, Theory and evaluation metrics for learning disentangled representations, arXiv preprint arXiv:1908.09961 (2019).

[82] S. Zhou, M. Gordon, R. Krishna, A. Narcomey, L. F. Fei-Fei, M. Bernstein, Hype: A benchmark for human eye perceptual evaluation of generative models, Advances in neural information processing systems 32 (2019).

[83] L. Regenwetter, B. Curry, F. Ahmed, Biked: A dataset for computational bicycle design with machine learning benchmarks, Journal of Mechanical Design 144 (3) (2022).

[84] M. M. Behzadi, H. T. Ilieş, Gantl: Toward practical and real-time topology optimization with conditional generative adversarial networks and transfer learning, Journal of Mechanical Design 144 (2) (2022).

[85] G. Giannone, F. Ahmed, Diffusing the optimal topology: A generative optimization approach, arXiv preprint arXiv:2303.09760 (2023).

[86] M. P. Bendsøe, N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, Computer methods in applied mechanics and engineering 71 (2) (1988) 197–224.

[87] G. I. Rozvany, M. Zhou, T. Birker, Generalized shape optimization without homogenization, Structural optimization 4 (1992) 250–252.

[88] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE transactions on image processing 13 (4) (2004) 600–612.

[89] R. V. Woldseth, N. Aage, J. A. Bærentzen, O. Sigmund, On the use of artificial neural networks in topology optimisation, Structural and Multidisciplinary Optimization 65 (10) (2022) 294.

[90] R. Bostanabad, Y. Zhang, X. Li, T. Kearney, L. C. Brinson, D. W. Apley, W. K. Liu, W. Chen, Computational microstructure characterization and reconstruction: Review of the state-of-the-art techniques, Progress in Materials Science 95 (2018) 1–41.

[91] S. Yoo, S. Lee, S. Kim, K. H. Hwang, J. H. Park, N. Kang, Integrating deep learning into cad/cae system: generative design and evaluation of 3d conceptual wheel, Structural and multidisciplinary optimization 64 (4) (2021) 2725–2747.

[92] S. Oh, Y. Jung, S. Kim, I. Lee, N. Kang, Deep generative design: Integration of topology optimization and generative models, Journal of Mechanical Design 141 (11) (2019).

[93] T. Nguyen, Q.-H. Pham, T. Le, T. Pham, N. Ho, B.-S. Hua, Point-set distances for learning representations of 3d point clouds, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 10478–10487.

[94] M. Tantardini, F. Ieva, L. Tajoli, C. Piccardi, Comparing methods for comparing networks, Scientific reports 9 (1) (2019) 1–19.

[95] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever, Learning transferable visual models from natural language supervision (2021). arXiv:2103.00020.

[96] V. Khrulkov, L. Mirvakhabova, E. Ustinova, I. Oseledets, V. Lempitsky, Hyperbolic image embeddings, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6418–6428.

[97] F. Faghri, D. J. Fleet, J. R. Kiros, S. Fidler, Vse++: Improving visual-semantic embeddings with hard negatives, arXiv preprint arXiv:1707.05612 (2017).

[98] G. Dai, J. Xie, Y. Fang, Siamese cnn-bilstm architecture for 3d shape representation learning., in: IJCAI, 2018, pp. 670–676.

[99] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al., Universal sentence encoder, arXiv preprint arXiv:1803.11175 (2018).

[100] H. Cai, V. W. Zheng, K. C.-C. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, IEEE Transactions on Knowledge and Data Engineering 30 (9) (2018) 1616–1637.

[101] S. Morozov, A. Voynov, A. Babenko, On self-supervised image representations for gan evaluation, in: International Conference on Learning Representations, 2020.

[102] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

15. Appendix

15.1. Miscellaneous DGM evaluation areas

There are several broader areas of DGM evaluation that we deem out of scope and do not directly discuss. These areas include but are not limited to:

1. Cost: Computational, time, memory, and energy costs of training, inference and deployment.
2. Robustness: The model's resilience against noise, adversarial attacks, or perturbations in input data.
3. Transferability and data efficiency: The capability of the model to generalize and perform well on unseen tasks with limited original data or fine-tuning.
4. Stability and consistency: The degree to which the model produces similar results when provided with similar inputs and its ability to avoid issues like mode collapse or gradient vanishing/exploding during training.
5. Bias, and fairness: The resilience of the model to biased data.
6. Privacy: The ease with which revealing information about training data can be reconstructed from a trained model.

15.2. Calculating Distances between Designs

Many evaluation metrics necessitate the ability to calculate the distance between designs. This task is often nontrivial. Below, we provide some methods to calculate distances in various data modalities, as well as some strategies to leverage embeddings to calculate distances.

1. **Continuous variables**: Common distance metrics include Euclidean distance and Manhattan distance. These distances are sensitive to the relative scaling between parameters. As such, care must be taken to appropriately scale data. Cosine similarity is another common approach to measuring 'distance' between variables.
2. **Ordinal Discrete Variables**: For ordinal discrete variables (such as the number of teeth on a gear), L_1, L_2, and L_∞ are all valid metrics.
3. **Categorical Variables**: For discrete variables with no sequential significance (i.e., categorical data), one-hot encoding can transform the data into boolean data, to which the aforementioned distance methods apply. When directly working in categorical spaces, Hamming distance is a widely used metric.

4. **Pixels**: Calculating distances directly on images is challenging, largely due to their high dimensionality. Depending on the nature of the images, directly calculating pixel distance using a simple method like L2 may be a valid approach. For example, this may work on topologies represented as images, while on photographs of vehicles, pixel distance is unlikely to reflect similarity in the content of images. As a strong alternative, distances between images can be calculated using Structural Similarity (SSIM) [88]. Recently, calculating distances between images is most commonly done using learned embeddings.
5. **3D geometry**: For many common 3D geometry representation methods like point clouds, meshes, and rasterized curves, point-set distances like Chamfer and Hausdorff distance are commonly used. We refer the reader to papers like [93] for further details on metrics used in computer graphics.
6. **Text**: Directly calculating distances between pieces of textual data is challenging. While simple categorical distances can be calculated on tokenized text data using cosine distance, learned text embeddings are usually the preferred approach to calculating distances on text data.
7. **Graphs**: Measuring distance between graphs (networks) is a common problem known as the 'network comparison' problem. Typically methods depend on the type of graph (directed vs. undirected, weighted vs. unweighted, known vs. unknown nodes, etc.). We refer readers interested in calculating distances between graphs to [94].
8. **Multimodal Data** Calculating distances between designs represented using multimodal data is nontrivial and inherits the challenges of the individual modalities. Though this problem requires further research, there are a few strategies that researchers can select from. A simple approach takes the weighted sum of distances across different modalities but requires tuning of weights. Another approach constructs an embedding from the multimodal data over which to measure distances [95].

While calculating distances in the original data modality is often viable, a common alternative approach is to find features of the data and then calculate distances between those features. These features are often learned using a machine learning model instead of being hand-picked. These approaches are very common when working with images [96, 97], 3D geometry [98], text [7, 99], and graphs [100]. However, learned embeddings suffer from a potential risk of poor generalizability. For example, a breadth of literature

has documented how image embeddings struggle to generalize even across similar natural image datasets [101, 49, 48, 82]. In general, great care must be taken when using learned embeddings to calculate distances, regardless of the modality of the original data.

When working with multimodal data, calculating distances can be challenging. Shared embeddings are a viable approach to calculating distances between datapoints of different modalities. For example, Contrastive Language-Image Pretraining (CLIP) [95] embeddings map images and text into a common embedding space in which distances can be calculated, as showcased in Figure 13.

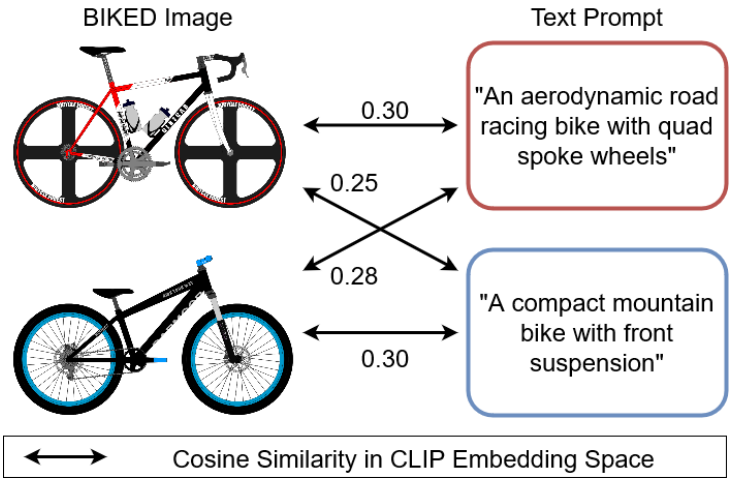


Figure 13: Cosine similarity between bicycle images from [83] and text prompts in CLIP embedding space. Shared embeddings allow for distance calculation between datapoints of different modalities.

15.3. Model and Hyperparameter Settings for Evaluation Experiments

We present the training settings for all models tested on the synthetic datasets and all evaluation metrics used.

GAN and cGAN Models.

1. Discriminator and generator models are 2 hidden-layer neural networks with 100 parameters with leaky ReLU activation with slope coefficient $\alpha = 0.2$.
2. The latent vectors are four-dimensional

3. Discriminator and generator optimizers are Adam [102] with learning rate $1e-3$.
4. The models are trained over 5000 randomized batches of 32 samples.

VAE Model.

1. Encoder and decoder models are 3 hidden-layer neural networks with 100 parameters with ReLU activation.
2. The latent dimension is 4.
3. The model is trained for 100 epochs of batches of 100.
4. the model's optimizer is Adam with learning rate $1e-3$.

MO-PaDGAN Model.

1. Discriminator an generator architectures and optimizers are the same as in the GAN model. latent dimension and training epochs are also unchanged.
2. All training parameters are identical to [55] except $\gamma_0 = 5$ and $\gamma_1 = 2$.

Experiment 1: Evaluating statistical similarity on 2D data (Sec. 4.5.

1. Nearest datapoint, nearest generated sample, and rediscovery use Euclidean distance.
2. F1, F10, F0.1, and AUC-PR use 20 clusters, angle resolution of 1000, and 10 clustering runs.
3. ML efficacy uses an auxiliary regression task of predicting the objectives used in Sec. 7.3. A K-nearest-neighbors regressor with $K = 5$ is used and evaluated using coefficient of determination (R^2).

Experiment 2: Evaluating design exploration on 2D data (Sec. 5.3.

1. Nearest datapoint and inter-sample distance uses Euclidean distance.
2. DPP Diversity uses a subset size of 10 and an exponentiation parameter of 0.1.

Experiment 3: Evaluating design constraints on 2D data (Sec. 6.4.

1. Predicted constraint satisfaction uses K-nearest-neighbors with $K = 5$.
2. Nearest Invalid Datapoint uses Euclidean Distance.

Experiment 4: Evaluating design quality on 2D data (Sec. 7.3.

1. The hypervolume reference point is selected as the 99th quantile performance values from the dataset.
2. The Pareto-optimal set for KNO1 is given as $x + y = .4705$.
3. The target for all target-related scores is set as (0.5,0.5). Objectives are weighted equally.
4. DTAI parameters are $\alpha = \beta = [1, 1]$.

Experiment 5: Evaluating conditioning adherence on 2D data (Sec. 8.5.

1. cVAE and cGAN architectures are identical to unconditional counterparts, except for
2. Both marginal and conditional F0.1 and F10 use 20 clusters, and angle resolution of 1000, and 10 clustering runs.
3. Conditional scores used the nearest 10% of the dataset as the constraint-satisfying subset.
4. Condition adherence and reconstruction use mean squared error.
5. Condition reconstruction uses K-nearest-neighbors regression with $K = 5$.

15.4. Model and Hyperparameter Settings for Bicycle Frame Design Problem

Evaluation Metric Settings.

1. Nearest datapoint, nearest generated sample, and inter-sample distance use Euclidean distance.
2. F1 score uses 20 clusters, angle resolution of 1000, and 10 clustering runs.
3. DPP Diversity uses a subset size of 10 and an exponentiation parameter of 0.1.
4. The target for all target-related scores is set as 0.75 quantile performance value in each objective across the FRAMED dataset
5. DTAI parameters are $\alpha = [1, 1, 1, 1, 2, 1, 1, 4, 4, 3], \beta = [4, 4, 4, 4, 4, 4, 4, 4, 4, 2]$.

Model Parameters.

1. Discriminator and generator models for the GAN, MO-PaDGAN, and DTAI-GAN are 2 and 3 hidden-layer neural networks respectively with 128 parameters and leaky ReLU activation with slope coefficient $\alpha = 0.2$.

2. All PaDGAN and DTAI-GAN training parameters are identical to [55] except $\gamma_0 = 5$ and $\gamma_1 = 0.5$
3. The latent vectors are four-dimensional
4. Discriminator and generator optimizers are Adam [102] with learning rate $1e-4$.
5. The models are trained over 50000 randomized batches of 8 samples.
6. MO-PaDGAN and DTAI-GAN used differentiable performance surrogate regressors for performance evaluations of generated designs. DTAI-GAN used a differentiable constraint satisfaction surrogate classifier for constraint satisfaction likelihoods. Performance prediction and constraint satisfaction prediction networks were optimal neural networks identified through Bayesian optimization in [12].

15.5. Public Code and Evaluation Metrics Package

In our publicly released codebase we have provided the following:

1. 6 model architectures
2. 23+ evaluation metrics
3. 14+ synthetic datasets
4. Code to generate score reports with confidence bounds
5. Code to generate training convergence plots
6. Code to generate plots of generated samples on 2D data, as shown in paper
7. Code to generate animations of generated samples throughout training process on 2D data

These features are organized as an easy-to-use package that can be quickly used to evaluate other researchers models.