Learning from Invalid Data: On Constraint Satisfaction in Generative Models

Giorgio Giannone*
Massachusetts Institute of Technology
Technical University of Denmark
ggiorgio@mit.edu

Lyle Regenwetter*
Massachusetts Institute of Technology
regenwet@mit.edu

Akash Srivastava*
MIT-IBM Watson AI Lab
akashsri@mit.edu

Dan Gutfreund
MIT-IBM Watson AI Lab
dgutfre@us.ibm.com

Faez Ahmed
Massachusetts Institute of Technology
faez@mit.edu

Abstract

Generative models have demonstrated impressive results in vision, language, and speech. However, even with massive datasets, they struggle with precision, generating physically invalid or factually incorrect data. This is particularly problematic when the generated data must satisfy constraints, for example, to meet product specifications in engineering design or to adhere to the laws of physics in a natural scene. To improve precision while preserving diversity and fidelity, we propose a novel training mechanism that leverages datasets of constraint-violating data points, which we consider invalid. Our approach minimizes the divergence between the generative distribution and the valid prior while maximizing the divergence with the invalid distribution. We demonstrate how generative models like Generative Adversarial Networks and Denoising Diffusion Probabilistic Models that we augment to train with invalid data vastly outperform their standard counterparts which solely train on valid data points. For example, our training procedure generates up to 98% fewer invalid samples on 2D densities, improves connectivity and stability four-fold on a stacking block problem, and improves constraint satisfaction by 15% on a structural topology optimization benchmark in engineering design. We also analyze how the quality of the invalid data affects the learning procedure and the generalization properties of models. Finally, we demonstrate significant improvements in sample efficiency, showing that a tenfold increase in valid samples leads to a negligible difference in constraint satisfaction, while <10% invalid samples lead to a tenfold improvement. Our proposed mechanism offers a promising solution for improving precision in generative models while preserving diversity and fidelity, particularly in domains where constraint satisfaction is critical and data is limited, such as engineering design, robotics, and medicine.

1 Introduction

Generative models have shown remarkable qualitative and quantitative results in vision, language, and speech. However, these models often struggle with precision and notoriously generate an abun-

*Equal Contribution.

Preprint.

dance of physically or factually inaccurate image and text content. While this lack of precision is a nuisance in image or text synthesis, it becomes a paramount concern in domains like engineering design with high-stakes (including safety-critical) constraints. Constraints in the design of products such as cars and airplanes are ubiquitous and varied, but common examples include geometric constraints (such as disconnected or colliding components), functional requirements (such as maximum weight or drag coefficient), industry standards, and manufacturing constraints. Classic deep generative models are trained only on valid samples, i.e., samples that fulfill the constraints prescribed by the training distribution. In the case of engineering product design, these would be existing functional products. Models struggle to accurately learn the boundary of the valid space using only valid data. Since this task is equivalent to learning a classification problem with only one class present in the data, this is no surprise. To address this issue, we propose a simple mechanism to leverage datasets of constraint-violating data points (invalid samples) while training deep generative models. This invalid data helps generative models accurately estimate the density of valid samples, as indicated in Figure 1. Across several test problems, we determine that models trained using invalid data can achieve orders of magnitude better constraint satisfaction. Drawing parallels with classification problems, where underrepresented class samples can provide more meaningful information, invalid data can prove to be significantly more informative than valid data in generative tasks. Through our research, we have substantiated this concept, demonstrating that including invalid data, even when just a fraction of the dataset, can dramatically enhance constraint satisfaction - often improving it by multiple orders of magnitude. This finding implies that models trained using invalid data often showcase better sample efficiency than conventional models. This revelation is particularly significant, given that in most engineering contexts, collecting invalid data is generally less costly than acquiring valid design data.

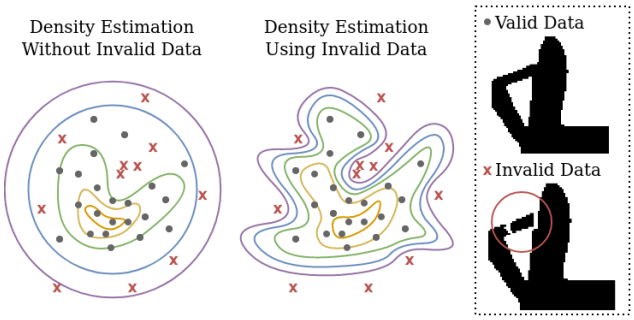


Figure 1: Many real-world data distributions have gaps in their support caused by constraints. Generative Models classically estimate these distributions using constraint-satisfying (valid) samples. We propose a new training method for generative models that leverages constraint-violating (invalid) data to more accurately estimate the density of in-distribution (valid) data. For example, by examining structural topologies that violate geometric continuity constraints, a model can better learn to generate geometrically valid topologies. For more visualizations, see Figure 9 in the Appendix.

Contributions.

- **(i)** We present *Constraint-Satisfying Generative Models*, a method to leverage invalid samples to inject constraint information into generative models. Our method is grounded in Density Ratio Estimation, providing a principled way to deal with situations where precision in generative models is important.
- **(ii)** We showcase our method on intuitive examples, including complex synthetic 2d continuous densities and a stacking block problem. We demonstrate that incorporating invalid data can achieve overwhelmingly superior constraint satisfaction and distribution matching while improving the sample efficiency of generative models by orders of magnitude.
- **(iii)** We provide preliminary experimental results on a challenging engineering design problem, structural topology optimization, showing how invalid designs effectively improve performance and constraint satisfaction.

2 Background

Below, we discuss types of constraints commonly used in engineering literature, common ways of constraint handling using generative models, and their limitations. We also provide preliminaries on some deep generative models.

Types of Constraint Information. Constraint handling has a rich history in engineering design, spanning some of the literature in deep generative models and extending to adjacent fields like topology optimization and multi-objective engineering optimization. Generally, we distinguish four types of constraint handling in design problems [66]. *(i) No Constraint Information*: no information about constraints is given or can be collected, and learning constraints is typically unfeasible or extremely challenging in a finite data regime. *(ii) Dataset of Invalid Designs*: a collection of constraint-violating invalid designs may be available or can be generated. Our method leverages such invalid data to learn a constraint-satisfying generative model. The value of negative samples is largely governed by their relative difficulty. Hard invalids fall near the valid data manifold, while easy invalids lie deep within constrain-violating regions. *(iii) Constraint Check*: we have access to a check that determines whether a design satisfies constraints. This may be as intractable as a black-box 'oracle' or simulator, such as a 3D parametric model that only builds when a design is geometrically valid. This check may also be computationally expensive, limiting its use. *(iv) Closed-form Constraints*: a constraint check may also be completely closed-form. In such scenarios, direct optimization is often favored over generative models in design problems. In other cases, constraint-enforcing rules can be built into the model structure, an approach used in some generative models for molecular design [20, 34]. We note that each level of constraint information is strictly more informative than the previous. In this paper, we focus on the scenario in which a small dataset of invalid samples is available or can be generated, which is common in applications such as structural design, mobility design (e.g., cars, bikes, ships, airplanes), and material synthesis.

Active Constraint Handling in Generative Models. Constraint satisfaction is an explicit goal in many design generation problems [5, 13, 19, 20, 52, 56, 75]. Despite this, the overwhelming majority of generative models in design do not actively consider constraints [65, 66], even though opportunities for active constraint satisfaction are often available. Several engineering design datasets [28, 45, 67] feature constraint-violating designs, and many others have documented checks for validity [89, 90], allowing datasets of invalid designs to be generated. One common approach for active constraint satisfaction involves pretraining a supervised model to predict constraint satisfaction and querying this model during model training with an associated loss over the prediction. Often this model directly predicts constraint satisfaction likelihood [28, 45, 63], though it can also predict physical properties which are combined in a more complex constraint check [87]. Structured formulations can be employed in scenarios where explicit constraint information is available. One approach to incorporating explicit constraints is through conditioning, which is popular in many design generation problems [3, 32, 44, 45, 52]. In one variant, conditional modeling, the generative model conditions on the constraints denoted as $\mathbf{c}$ and learn a conditional distribution, $p(\mathbf{x}|\mathbf{c})$, where $\mathbf{x}$ represents the generated output. Another variant is conditional constraint generation, which involves a two-step process. Firstly, a generative model generates unconstrained designs or primitives, denoted as $p(\mathbf{p})$. Then, a separate constraint model, $p(\mathbf{c}|\mathbf{p})$, is trained to learn suitable constraints for a given primitive. Finally, a solver, denoted as $S(\mathbf{p}, \mathbf{c})$, obtains the final design $\mathbf{x}$ by incorporating both the generated primitives and their corresponding constraints. This approach is commonly used in Computer-Aided Design (CAD) generation, where simple 2D primitives and geometric constraints are jointly modeled in a generative model [59, 72].

Limitations. While these approaches can be effective in scenarios where constraints are well-defined, as we show later, they may not be easily applicable in general cases where constraints are not readily available or easily specified. Therefore, alternative implicit approaches, where constraints are learned from demonstrations, such as leveraging invalid data or utilizing unsupervised learning methods, can provide valuable solutions in scenarios where explicit constraint information is not accessible.

Generative Adversarial Networks. Generative Adversarial Networks (GANs [1, 29, 47, 55, 81]) have emerged as a powerful framework for generating realistic and diverse data samples. GANs have two main components: a generator p_θ and a discriminator f_ϕ. The generator learns to generate synthetic data samples by transforming random noise into meaningful outputs, while the discriminator aims to distinguish between real and generated samples. The standard GAN loss can be written as:

$$L(\theta, \phi) = \mathbb{E}_{p_x(\mathbf{x})}[\log f_\phi(\mathbf{x})] + \mathbb{E}_{p_\theta(\mathbf{x})}[1 - \log(f_\phi(\mathbf{x}_\theta))], \tag{1}$$

where f_ϕ is the output of a binary classifier. This loss can be seen as a special version of a general f-divergence formulation [55]. Training a GAN consists in iterating over $\min_\theta \max_\phi L(\theta, \phi)$. In practice, for a given set $\bar{\theta}$, we optimize $\min_\phi -L(\bar{\theta}, \phi)$, and for a set $\bar{\phi}$, we optimize $\min_\theta L(\theta, \bar{\phi})$. GANs can also be interpreted in terms of estimating the density ratio [30, 81] between the data and

the generated distribution, $r(\mathbf{x}) = p_x(\mathbf{x})/p_\theta(\mathbf{x})$. Such ratio can be estimated by a discriminative model as $r_\phi = f_\phi(\mathbf{x})/(1 - f_\phi(\mathbf{x}))$ and $r_\phi = 1$ gives us $p_\theta = p_x$.

Classifier Guidance in Diffusion Models. Diffusion Models are powerful generative models that learn the data distribution denoising a multiscale noisy process defined by an approximate posterior q using a model p_θ. Let $\mathbf{x}_0$ denote the observed data $\mathbf{x}_0 \in \mathbb{R}^D$. Let $\mathbf{x}_1, ..., \mathbf{x}_T$ denote T latent variables in $\mathbb{R}^D$. In particular, the generative model for a latent step t can be written as:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}, \mathbf{g}) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, \mathbf{c}) + \sum_{p=1}^{P} \mathbf{g}_p,\ \sigma), \tag{2}$$

where $\mathbf{c}$ is a conditioning term and $\mathbf{g}$ is a guidance term, containing information to guide the sampling process toward regions with low invalid configurations. We can then adapt the loss as $L_{t-1,\epsilon}(\mathbf{x}, \mathbf{c}) = \mathbb{E}_{q(\epsilon)}\left[w_t \|\epsilon_\theta(\mathbf{x}_t(\mathbf{x}_0, \epsilon), \mathbf{c}) - \epsilon\|_2^2\right]$, where conditioning $\mathbf{c}$ is applied during training, the guidance mechanism $\mathbf{g}$ is applied only at inference time, merging the noisy gradients of the classifier with the unconditional score of a diffusion model to learn a conditional score. Interestingly, Diffusion Models have a built-in technique that can be leveraged for constraint satisfaction. We can leverage classifier guidance [25] to discriminate between valid and invalid samples, helping a generative model to learn the data distribution and move toward regions with high validity and far from low-validity regions. Notice that for low levels of noise, the discriminator is approximately estimating the ratio between p_x and the invalid distribution; where for high levels of noise, we estimate the ratio between p_θ and the invalid distribution.

3 Method

We present a method that assimilates knowledge from *invalid* samples into a probabilistic generative framework, utilizing divergence minimization and density ratio estimation techniques. By *invalid* samples, we refer to data instances that fail to satisfy the specified constraints. Our approach purposefully integrates these constraint-violating instances into the learning process, alongside the valid samples, aiming to augment the model's capabilities to adhere to the constraints more accurately. The underlying hypothesis is that by exposing the model to a blend of valid and invalid samples, it gains a nuanced understanding of the patterns correlating with constraint satisfaction. In this section, we initially discuss why the conventional divergence minimization-based approach to deep generative modeling falls short in addressing constraint satisfaction. Following that, we explain how the current method can be adjusted to exploit invalid samples to improve constraint satisfaction. Lastly, we propose a practical learning algorithm that utilizes density ratio estimators to train highly precise deep generative models.

Divergence Minimization with Valid Samples. Let p_x be data distribution and p_θ the generative model. Given N samples from p_x, $\mathcal{D} = \{x_i\}_{i=1}^N$, the objective of generative modeling is to find a setting θ^* of θ, such that, for an appropriate choice of discrepancy measure, $p_{\theta^*} = p_x$. A common choice for such discrepancy measure is the Kullback–Leibler or KL divergence, which allows us to find θ^* as the solution to the following optimization problem,

$$\theta^* = \arg\min_\theta \mathbb{KL}[p_\theta \| p_x] = \arg\min_\theta \int p_\theta(\mathbf{x}) \left[\log \frac{p_\theta(\mathbf{x})}{p_x(\mathbf{x})}\right] d\mathbf{x}. \tag{3}$$

KL-divergence belongs to the f-divergence family of discrepancy measures over probability distributions and requires that p_θ is absolutely continuous with respect to p_x, denoted by $p_\theta << p_x$. While our proposal can be easily extended to Integral Probability Measures (IPM) [50] and other divergences, for the remainder of this discourse, we will primarily focus on delineating our technique in the context of KL-divergence. In practice, direct optimization of (3) is often intractable. As such, it is common in deep generative modeling, to learn θ by using either a tractable lower-bound to a slightly different variant of (3) [8, 33, 37] or by using plug-in or direct estimators of the divergence measure [10, 29, 30, 60, 80, 81, 82]. In both of these cases, under certain conditions, as $N \to \infty$, theoretically, it holds that, $\theta \to \theta^*$. However, in practice, N is typically not very large, especially in high-dimensional problems where N needs to scale exponentially in the number of dimensions. This implies that there remains a finite discrepancy between the model and data distributions. This mismatch often manifests in p_θ allocating high probability mass in regions where p_x may not have significant empirical support i.e. allocates very small probability mass. In domains such as engineering design, where the invalid designs tend to be very close to the valid designs, this leads to the

generation of invalid designs with high probability. This lack of precision underpins the relatively limited success of deep generative models in the engineering design domain [66].

Divergence Minimization with Valid and Invalid Samples. Let n_x denote the *invalid distribution* i.e., the distribution of invalid designs. Assuming mutual absolute continuity [2] of p_x, p_θ and n_x, we can now re-write (3) as,

$$\begin{aligned}&\arg\min_\theta \int p_\theta(\mathbf{x}) \left[\log p_\theta(\mathbf{x}) - \log p_x(\mathbf{x}) + \left(\log \frac{n_x}{n_x}\right)\right] d\mathbf{x} \\ &= \arg\min_\theta \int p_\theta(\mathbf{x}) \left[\log \frac{p_\theta(\mathbf{x})}{n_x(\mathbf{x})} - \log \frac{p_x(\mathbf{x})}{n_x(\mathbf{x})}\right] d\,\mathbf{x}. \end{aligned} \tag{4}$$

While the solution for (4) is the same as the solution for (3) i.e. $p_{\theta^*} = p_x$, the model is now directly incentivized to allocate the same amount of probability mass to the samples from n_x as does the data distribution p_x. This ensures that when trained using finite N, the model does not allocate high probability mass to invalid samples. In other words, training under (4) encourages the model to minimize its discrepancy with respect to p_x such that its own discrepancy with respect to n_x matches exactly that of p_x and n_x. Another important benefit of the reformulation in (4) is that in cases where sampling from n_x is inexpensive (such as in the engineering design domain), the sample efficiency of the model with respect to samples from p_x improves as shown in the next section.

Density Ratio Estimation and Generator Training. Let p and q be two densities, such that p and q are absolutely continuous with respect to each other. Then, their ratio is defined as $r(\mathbf{x}) = p(\mathbf{x})/q(\mathbf{x})$ over their support. We can estimate this density ratio $r(\mathbf{x})$ by training a binary classifier (using the binary cross-entropy loss) to distinguish between samples from p and q [30, 80, 82]. We employ a pair of such density ratio estimators (DRE) $(f_\phi^{p_\theta/n_x}, f_\gamma^{p_x/n_x})$, each parameterized by a deep neural network with shared weights in the representation layers, to estimate $p_\theta(\mathbf{x})/n_x(\mathbf{x})$ and $p_x(\mathbf{x})/n_x(\mathbf{x})$ in (4) respectively. An interesting consequence of our formulation (4) is that it avoids the saddlepoint optimization formulation of GANs, thereby not pitching the generator to fool the critic in order to match p_θ to p_x. However, in practice[3], we found that a slight variant of (4),

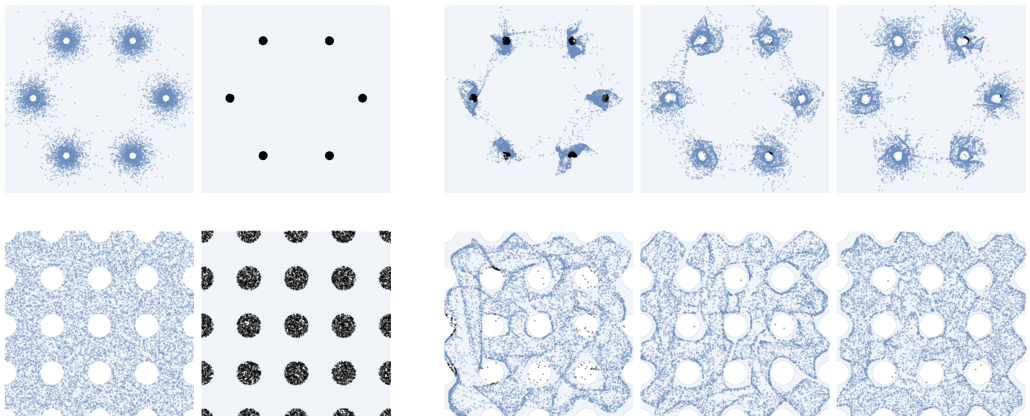
$$\arg\min_\theta \int p_\theta(\mathbf{x}) \left(\frac{1}{2}\left[\log \frac{p_\theta(\mathbf{x})}{n_x(\mathbf{x})} - \log \frac{p_x(\mathbf{x})}{n_x(\mathbf{x})}\right] + \frac{1}{2}\left[\log \frac{p_\theta(\mathbf{x})}{p_x(\mathbf{x})}\right]\right) d\,\mathbf{x}, \tag{5}$$

which does involve GAN-like adversarial training between the generator and the critic works well, especially in high-dimensional problems. Intuitively, in Eq. 6 the generative model is directly matched to the training data, minimizing the discrepancy in a KL sense, providing a strong signal for model learning over θ. At the same time, Eq. 4 gives us a principled way to inform the model about regions we do not want to explore. We also explore the use of kernel-based diversity mechanisms to improve mode coverage (see Appendix B).

Hard-vs-Easy Invalid Configurations. *Hard* invalid configurations are samples that do not fulfill the prescribed constraint by a relatively small margin, making them close to the distribution of valid samples. While *Easy* invalid configurations are easier to collect, the performance of our method significantly improves with hard invalid configurations. This is because when two densities are *well-separated* i.e. easy to distinguish, in the finite sample setting, there exist a number of perfect classification boundaries. In such cases, the classifier tends to significantly underestimate the density ratio when it fails to learn the Bayes-optimal classifier boundary. This problem has been recently characterized as the *density-chasm issue* [68, 80]. Our experiments demonstrate that even a small number of hard invalid samples is sufficient to significantly enhance constraint satisfaction. In the following, we will call models trained to leverage invalid samples **CS-GM**, for *Constraint-Satisfying Generative Model*, where the specific implementation details of our method will be adapted for different modeling choices.

[2] Unlike [77], in the domain of engineering designs, where the constraints are often non-binary, we cannot assume the supports of p_x and n_x to be disjoint.

[3] We found that for a carefully tuned value of λ, $\arg\min_\theta \int p_\theta(\mathbf{x}) \left(-\lambda \left[\log p_\theta(\mathbf{x})/n_x(\mathbf{x})\right] + \left[\log p_\theta(\mathbf{x})/p_x(\mathbf{x})\right]\right) d\mathbf{x}$ also works equally well empirically.



(a) Valid **(b)** Invalid **(c)** GAN **(d)** CS-GAN **(e)** CSD-GAN

Figure 2: Generated Distributions from several generative models on two 2D toy problems (`Problem 1` on top, `Problem 2` on the bottom). Valid data points and samples are shown in blue and invalid ones in black. The GAN models trained with invalid data (CS-GAN) generate significantly fewer invalid samples, matching the original distributions much more closely than vanilla models. When further augmenting the GAN with diversity (CSD-GAN), we see that it improves coverage, slightly improving similarity to the original densities.

4 Experiments

4.1 Complex 2D Densities with Constraints

We first showcase our approach on two easy-to-visualize 2D toy problems. Problem 1 is an adaptation of a classic multi-modal test problem made significantly more challenging with the addition of small invalid regions in the centers of each mode. Problem 2 is a simple uniform distribution with many discontinuous circular regions of invalidity in a grid pattern. 10k valid and invalid datapoints are randomly sampled, constituting the valid and invalid datasets. Architecture and training details are included in the appendix.

Evaluation Metrics. To measure performance, we calculate several scores obtained from precision-recall curves [71], including F1, and the area under the curve (abv. AUC-PR). We calculate the mean distance to the nearest dataset point for each generated sample as a simple estimate for accuracy (abv. NDP). Similarly, we calculate the mean distance to the nearest generated sample for each point in the dataset (abv. NGS). Finally, we calculate Maximum mean Discrepancy and proportion of invalid generated samples (abv. Validity)[4].

Results. Table 1 and Figure 2 present a performance comparison of a vanilla GAN and our proposed CS-GAN that leverages invalid data. We also demonstrate that CS-GAN can be further augmented with a Determinantal Point Process (DPP) diversity loss[5] (CSD-GAN) to improve distribution-matching performance. The results show that CS-GAN achieves near-perfect constraint satisfaction compared to the vanilla model with the CSD-GAN closely trailing. Not only do our proposed models achieve vastly superior constraint satisfaction, but they also attain stronger overall distribution-matching performance as indicated by MMD, F1, and AUC-PR. This dominance spans both precision (NDP) and recall (NGS). In general, the diversity-augmented CSD-GAN achieves slightly higher distribution-matching performance compared to the CS-GAN.

How Much Invalid Data is Enough? In the realm of generative models, it is theoretically possible to recover the underlying data distribution, represented by p_x, when provided with an infinite amount of valid data, model capacity, and computational resources. However, in practical scenarios where data throughput and computing are not only finite but limited, like engineering design and scientific research, simply increasing the volume of valid data is not a viable strategy to improve constraint satisfaction. In Table 2, we present empirical evidence to support our arguments. By solely increasing

[4]Percent difference for metrics that are minimized at zero are calculated as $\frac{\text{ours}-\text{vanilla}}{\text{vanilla}}$. Percent difference for metrics that are maximized at one is calculated as $\frac{\text{ours}-\text{vanilla}}{1-\text{vanilla}}$.

[5]More details on DPP diversity are included in the appendix.

Table 1: Metrics for the 2d density problems depicted in Figure 2. Percentage improvements over the vanilla model are indicated in green. Training with negative samples results in 97% fewer invalid samples and consistent improvements across distribution-matching metrics. Incorporating diversity during training further improves distribution-matching while still retaining large validity improvements. Results indicate that our models not only learn the density of valid samples better but almost perfectly learn to avoid invalid regions of the space. Extended results with other models and standard deviations are included in Table 7.

	Problem 1			Problem 2		
	GAN	**CS-GAN** (ours)	**CSD-GAN** (ours)	**GAN**	**CS-GAN** (ours)	**CSD-GAN** (ours)
↓ **Validity**	0.1433	**0.0042 (-97.1 %)**	0.0073 (-94.9 %)	0.0236	**0.0005 (-97.9 %)**	0.0007 (-97.0 %)
↓ **MMD**	0.0042	0.0017	**0.0003**	0.0009	0.0005	**0.0003**
↓ **NDP**	0.0126	**0.0110**	0.0125	0.0166	**0.0155**	**0.0155**
↓ **NGS**	0.0341	0.0163	**0.0158**	0.0271	0.0218	**0.0198**
↑ **F1**	0.7093	0.9274	**0.9479**	0.9531	0.9656	**0.9734**
↑ **AUC-PR**	0.7995	0.9835	**0.9898**	0.9932	0.9959	**0.9977**

Table 2: Comparison of **validity metric** for GAN models trained with different amounts of valid and invalid data. CS-GANs can generate many times fewer invalid samples, even when trained on orders of magnitude less data. Scores are averaged over four instantiations. Column-wise percent improvements are shown (fixing the amount of valid data and increasing the invalid data) **Lower is better.**

(a) Models		**(b)** Problem 1			**(c)** Problem 2		
	Invalid Samples	Valid Samples $1K$	$4K$	$16K$	Valid Samples $1K$	$4K$	$16K$
GAN	0	10.3%	10.0%	12.3%	2.4%	2.3%	5.9%
CS-GAN	$1K$	0.6% (-94 %)	0.3% (-97 %)	**0.3**% (-97 %)	0.8% (-66%)	0.6% (-73 %)	0.6% (-89 %)
CS-GAN	$4K$	**0.2** % (-98 %)	0.3% (-97 %)	0.4% (-96 %)	**0.2**% (-91 %)	0.3% (-86 %)	0.5% (-91 %)
CS-GAN	$16K$	**0.2** % (-98 %)	**0.1**% (-99 %)	**0.3**% (-97 %)	**0.2**% (-91 %)	**0.2**% (-91 %)	**0.1**% (-98 %)

the number of valid samples without incorporating invalid samples (first row - vanilla GAN), we observed no reduction in validity metric, despite a tenfold increase in valid samples. Conversely, when we introduce a modest proportion of invalid samples (5-10 %, second row, last column), we can achieve a significant reduction in the rate of generating invalid samples, decreasing it by one order of magnitude. Furthermore, as we continue to add more invalid samples, we generally observe improvements up to a certain threshold (per column results). These experiments serve as empirical justification for showing that even a small amount of invalid samples in generative models enhances constraint satisfaction.

4.2 The Stacked Blocks Problem

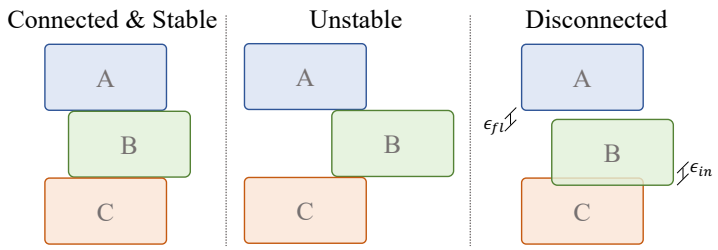


Figure 3: Stacked Blocks. Our goal is to generate valid stacks of blocks (on the left) that are (I) connected and (II) stable. To achieve such a goal we leverage sets of invalid samples, stacks that are either unstable (center), disconnected (right), or both. For the disconnected constraint, we consider the constraint unfulfilled in both cases of intersecting or floating blocks.

Block-stacking problems have long been studied as a case study in 'intuitive physics' [2, 69], on which many predictive and generative computational approaches have been tested [31, 79]. In this study, we address a simple block-stacking problem through the lens of constraint satisfaction.

Dataset. In generating the dataset, we define two constraints to ensure that: I) blocks are arranged in a physically plausible way and that there are no floating or unsupported blocks, and II) the center of gravity is supported. We call these constraints `connectivity` and `stability`. We designate four categories of datapoints: stable & connected (`S-C`), unstable & connected (`U-C`), stable & disconnected (`S-D`), and unstable & disconnected (`U-D`).

Table 3: Base model vs Constraint-satisfying model trained with invalid data. We test on 20 splits (1000 samples each) and evaluate metrics that quantify precision and constraint satisfaction. We consider boxes floating or intersecting if the distance between the points is larger than 0.9 units (the minimum distance between constraints in the invalid data is 1 unit). b: base block; m: middle block; t: top block. For floating $\mathbf{y}_b < \mathbf{y}_m$ and for intersecting $\mathbf{y}_b > \mathbf{y}_m$. For NDA-GAN, we select $\lambda = 0.75$ as the best-performing hyperparameter.

Metrics	GAN (w/o Invalid)	NDA-GAN (w/ Invalid)	CS-GAN (w/ Invalid)
$\downarrow$ `median`$(\lvert\mathbf{y}_b^1 - \mathbf{y}_m^0\rvert)$	$2.78 \pm 0.26\ u$	$5.12 \pm 0.37\ u$	**0.54** $\pm 0.03\ u$
$\downarrow$ `median`$(\lvert\mathbf{y}_m^1 - \mathbf{y}_t^0\rvert)$	$2.12 \pm 0.22\ u$	$1.91 \pm 0.22\ u$	**0.83** $\pm 0.05\ u$
$\downarrow$ `no-overlap`$(\mathbf{x}_b, \mathbf{x}_m)$	**0.41** ± 0.64 %	12.85 ± 3.41 %	1.82 ± 0.79 %
$\downarrow$ `no-overlap`$(\mathbf{x}_m, \mathbf{x}_t)$	**0.31** ± 0.66 %	17.89 ± 3.70 %	3.90 ± 2.01 %
$\downarrow$ `floating`$(\mathbf{y}_b, \mathbf{y}_m)$	20.44 ± 3.73 %	**14.47** ± 2.83 %	**13.78** ± 4.07 %
$\downarrow$ `floating`$(\mathbf{y}_m, \mathbf{y}_t)$	38.04 ± 4.49 %	20.73 ± 4.11 %	**13.94** ± 2.17 %
$\downarrow$ `intersect`$(\mathbf{y}_b, \mathbf{y}_m)$	64.59 ± 4.10 %	77.20 ± 3.97 %	**0.00** ± 0.00 %
$\downarrow$ `intersect`$(\mathbf{y}_m, \mathbf{y}_t)$	43.80 ± 4.47 %	54.82 ± 5.01 %	**30.79** ± 4.05 %
$\uparrow$ `connected`$(\mathbf{y}_b, \mathbf{y}_m)$	14.96 ± 3.04 %	8.32 ± 2.11 %	**86.21** ± 4.07 %
$\uparrow$ `connected`$(\mathbf{y}_m, \mathbf{y}_t)$	18.15 ± 3.06 %	24.44 ± 3.22 %	**55.26** ± 4.05 %

Fulfilling a Single Set of Constraints. For our initial experiment, we relax the `stability` constraint, allowing models to focus solely on `connectivity`. The connected data subsets (`S-C` and `U-C`) are designated as valid data, while disconnected subsets (`S-D` and `U-D`) are designated as invalid data. We train a vanilla model using only the valid data, and two models which utilize both the valid and invalid data: a NDA-GAN [77] and our proposed CS-GAN, which leverages Eq. 6. We score a variety of constraint satisfaction scores, as described in Table 3. CS-GAN outperforms the base model and the NDA-GAN by a large margin in most constraint-satisfaction scores, and in particular on global connectivity between boxes, indicating that the CS-GAN approach is effective in improving constraint satisfaction in situations where precision is important.

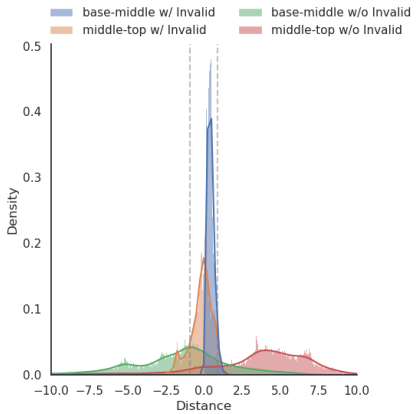


Figure 4: Distance from Constraint w/ and w/o leveraging Invalid Data for the block stack problem. The two vertical grey lines indicate the acceptable range or tolerance such that the constraints are considered satisfied. Our CS-GAN greatly reduces the distance that generated designs lie from the constraint target compared to a GAN, demonstrating its aptitude for constraint satisfaction.

However, the benefits of invalid data only extend to constraints that were included in the negative data. As shown in the upper half of Table 4, models trained on the relaxed case, including CS-GAN (fourth row) struggle to satisfy constraints not represented in the invalid dataset since they do not see representative examples of unstable configurations in invalid data.

Fulfilling multiple Sets of Constraints. Next, we consider the more challenging problem where stacks must simultaneously satisfy `connectivity` and `stablility` constraints. `S-C` is used as valid data while the `S-D`, `U-C`, and `U-D` are pooled to constitute the invalid data. Results are summarized in the lower half of Tables 4 and 8. We see that models trained using only valid data perform poorly in constraint satisfaction scores across the board. Autoregressive models (AR-GAN, [27, 46]) tend to work better than standard GAN (second row) but still fall behind compared to CS-GAN. CS-GAN connects and stabilizes an order of magnitude more configurations than the GAN (fifth row) training on only valid (connected and balanced) configurations. Interestingly, AR-GAN (sixth row) and NDA-GAN

(seventh row) can achieve high scores on connectivity, with NDA-GAN performing better than CS-GAN. However, when the model is challenged to fulfill both constraints, AR-GAN generates almost exclusively invalid configurations, and NDA-GAN satisfies all constraints less than 1/4 as frequently as the CS-GAN, even if presented with the same amount of valid and invalid data. We include more visualizations for the block stacking example in Appendix D.

Table 4: Overview of block stacking results. The upper half of the table shows results when the `stability` constraint is ignored during training (and unstable configurations are not designated as invalid data to NDA-GAN and CS-GAN). When stability is not considered during training, no generative models are able to reliably fulfill the `stability` constraint. The lower half shows results where both disconnected stacks and unstable stacks are considered invalid (and are provided to NDA-GAN and CS-GAN). CS-GAN improves constraint satisfaction by an order of magnitude over most baselines.

	Valid Data		Invalid Data		Metrics		
	Connected	Stable	Disconnected	Unstable	`Stability` ↑	`Connectivity` ↑	Both ↑
GAN	✓	✗	✗	✗	2.44 %	2.80 %	0.19 %
AR-GAN	✓	✗	✗	✗	0.085 %	13.23 %	0.00 %
NDA-GAN	✓	✗	✓	✗	0.39 %	6.39 %	0.00 %
CS-GAN (ours)	✓	✗	✓	✗	**3.75** %	**45.30** %	**1.03** %
GAN	✓	✓	✗	✗	83.20 %	4.83 %	3.28 %
AR-GAN	✓	✓	✗	✗	32.16 %	8.12 %	0.07 %
NDA-GAN	✓	✓	✓	✓	**84.65** %	12.63 %	8.85 %
CS-GAN (ours)	✓	✓	✓	✓	70.35 %	**41.10** %	**36.02** %

4.3 Guiding Generative Models with Invalid Topology Optimization Designs

With the previous experiments, we have demonstrated that CS-GAN is an effective method to improve constraint satisfaction on 2d densities and a stacked block problem. We now want to understand how well our method works on a complex engineering design problem. Additionally, we want to explore the connection between our method and alternative approaches to guide generative models like guidance in DDPMs. To do so, we consider the problem of optimal structural topology generation [76] (Fig. 8). A known solution for this problem is a powerful computational design tool called Topology optimization (TO), which determines locally-optimal configurations of a given structure, given a set of constraints. With generative models, we seek to improve topology quality and inference time compared to TO. To tackle this challenging problem, we train a TopoGAN [52] on a dataset of valid topologies obtained with a solver. Then we train the same architecture with invalid topologies, i.e. topology optimized with the wrong amount of material (Volume Fraction Error) or with low or null manufacturability (presence of Floating Material). Our goal is to learn a model, using Eq. 6, that fulfills the volume and/or manufacturability constraints better than the base model. Table 6 presents the results of this experiment, and we can see that CS-TopoGAN consistently decreases Volume Fraction Error with a negligible increase in Floating Material. Satisfying two constraints is hard, and the TO problem itself is still an open problem for modern generative models. However, this result corroborates our methodology as a powerful method to push the generative process away from invalid samples, showing promising results on an extremely challenging real-world problem. Finally, we also consider conditional DDPM variants with and without guidance wrt invalid samples (Table 5). Given that DDPM-based models can achieve very high constraint satisfaction as shown in [28, 45], we consider performance as the main metric. We perform evaluation of three different models, a conditional DDPM (cDDPM), with dense conditioning as proposed in [28] (CDM), and with physical fields conditioning [45] (TopoDiff), in terms of their

Table 5: Performance for generated topologies with and without leveraging invalid data. Performance is measured in terms of compliance error with respect to the optimal topology obtained with an iterative solver (like SIMP [4]). In TO, performance is directly related to how well the generated topologies fulfill the prescribed constraints (loads, boundary conditions). We see that classifier guidance using invalid data improves performance for the generated topologies, guiding the generation far from the region of invalidity, and corroborating the usefulness of density ratio estimation between valid and invalid data.

	In-distribution ↓		Out-of-distribution ↓	
	w/o Invalid	w/ Invalid	w/o Invalid	w/ Invalid
cDDPM [51]	17.33	**15.63**	64.40	**43.64**
CDM [28]	13.60	**7.65**	46.47	**40.03**
TopoDiff-G [45]	**5.40**	5.92	31.45	**18.34**

performance in both in-distribution and out-of-distribution constraints, with and without invalid samples. Interestingly, using classifier guidance with invalid data improves performance in most cases, corroborating the hypothesis that invalid data sampling can be leveraged to improve constraint satisfaction in generative design.

Table 6: Constraint Satisfaction for a GAN trained with valid topologies, and a CS-GAN trained with valid and invalid topologies. In this case, we consider invalid topologies as one that does not respect the prescribed volume fraction and those with floating material, i.e., that cannot be manufactured. We test our model on 9000 unseen configurations with in-distribution constraints. Topology Optimization is a particularly hard problem because different classes of semantic constraints are entangled. We can see that CS-GAN balances across constraints, improving by more than 15 % of the Volume Fraction error and maintaining a comparable Floating Material error rate. Training with CS-GAN greatly decreases the volume fraction error, making the model aware of a non-trivial invalid region. Such results align with recent literature on generative topology optimization [45].

	% Floating Material ↓	% Volume Fraction Error ↓
TopologyGAN [52]	**50.28** $\pm$ 0.49	10.36 $\pm$ 0.08
CS-TopologyGAN (ours)	**50.77** $\pm$ 0.52 (+ 0.97 %)	**8.76** $\pm$ 0.07 (-15.44 %)

5 Related Work

Density Ratio Estimation. Density Ratio Estimation (DRE [83]) is a critical technique in machine learning, particularly when evaluating distributions is not feasible or is computationally expensive [47]. DRE techniques are heavily employed for generative modeling, representation learning, mutual information estimation, and score matching[21, 29, 30, 80]. In the context of Generative Adversarial Networks (GANs [1, 29, 55]), the DRE methodology forms the underlying basis for their operation. A well-known technique for DRE is probabilistic classification [83], where a binary classifier is used. Each "dataset" from a particular distribution is assigned a pseudo label, enabling the comparison of distributions. However, accurate DRE from finite samples can be challenging, especially in high dimensions. This is because a naive construction of an estimator for the likelihood ratio may require a sample size that is exponential in the Kullback-Leibler divergence of the two densities to be accurate. To overcome this challenge, prior works have employed a divide-and-conquer approach. An example of this is the Telescoping Density Ratio Estimation (TRE) method [30, 68], which divides the problem into a sequence of easier DRE subproblems, thus simplifying the task. Despite its success, there are limitations to this approach, especially when the number of intermediate bridge distributions is increased. Noice contrastive estimator (NCE [30]) and hybrid generative models [68, 80, 81] are also based on the density ratio as underlying methodology, providing a flexible paradigm for large scale generative modeling.

Generative Models for Engineering Design. Generative models have recently seen extensive use in design generation tasks [65]. Generative Adversarial Nets, for example, have seen extensive use in many applications. In Topology Optimization, GANs [39, 52, 56, 57, 62, 74, 86, 94] are often used to create optimal topologies, bypassing the need for iterative solvers like SIMP. In computational materials design GANs [38, 42, 48, 84, 92, 95], VAEs [9, 12, 41, 43, 85, 88, 91], and other models are used to generate synthetic data to better learn process-structure-property relations [6]. A variety of generative models have been applied to 2D shape synthesis problems [17, 18, 19, 22, 40, 54, 93], such as airfoil design, and 3D shape synthesis problems [7, 54, 75, 96] such as mechanical component synthesis in engineering design. Finally, generative models have been proposed as a method to tackle various miscellaneous product and machine design tasks [23, 24, 64, 73].

Constraint Satisfaction in Machine Learning. Relative little research has been developed regarding generative models for constrained design. From a general point of view, Constraint Satisfaction Problems (CSPs) have been long studied in computer science and optimization in relation to optimal allocation, graph search, games, and path planning [70]. However, such constraints are mostly related to algorithmic complexity and memory allocation. In generative design [65, 76], constraint satisfaction has a different goal because we want to obtain a design with high performance but at the same time achieve diversity (distribution coverage) leveraging a probabilistic model. The constraints we want to fulfill are typically physical constraints imposed from the design requirement in the form of

boundary conditions, loads, and the amount of material. Recently, Neural Constraint Satisfaction [11] has been proposed to deal with objects in a scene to solve intuitive physics problems [31, 78]. In the CAD domain, structured models to handle constraints have been proposed[59, 72]. Conditional generative models have been proposed for structural topology optimization [52], leveraging physical fields [45, 52] and dense approximations [28] for high-quality candidate generation. These approaches rely on explicit constraint satisfaction. Instead, we focus on implicit constraint satisfaction, leveraging a dataset of invalid configurations to enhance the model capacity to generate valid designs.

Learning from Invalid Data. Invalid samples or invalid designs can be interpreted as special cases of negative data, where the invalid data is in-distribution from a distribution matching perspective but unfeasible from an engineering perspective because of violating constraints or poor manufacturability. Negative data has been studied in the context of retrieval, using triplet losses [35] and contrastive estimators [30] for representation learning [14, 58]. NDA [77] leverages negative out-of-distribution samples to improve fidelity and robustness in the vision domain. Another way to leverage negative data is as guidance using a classifier [45], to tell the model explicitly in which regions we do not want to generate.

6 Limitations and Conclusion

We presented *Constraint-satisfying Generative Models*, a framework to perform constraint satisfaction in generative models that leverages invalid samples without imposing inductive biases. During training, our framework pushes the generated samples far from regions of invalidity using an auxiliary discriminator jointly trained or used as guidance. Our approach is limited by the availability of a small but high-quality set of hard invalid samples, and such designs can be challenging to collect in some applications. In many domains like engineering design and scientific discovery, however, where invalid data is cheap and easily available and valid data is scarce, our approach takes a step towards expressive generative models that achieve precise adherence to constraints.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, ICML'17, pages 214–223. PMLR, JMLR.org, 2017.

[2] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.

[3] M. M. Behzadi and 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), 12 2021. 021711.

[4] M. P. Bendsøe. Optimal shape design as a material distribution problem. *Structural optimization*, 1:193–202, 1989.

[5] C. Bilodeau, W. Jin, T. Jaakkola, R. Barzilay, and K. F. Jensen. Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 12(5):e1608, 2022.

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

[7] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Context-aware content generation for virtual environments. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 50084, page V01BT02A045. American Society of Mechanical Engineers, 2016.

[8] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[9] R. Cang, H. Li, H. Yao, Y. Jiao, and Y. Ren. Improving direct physical properties prediction of heterogeneous materials from imaging data via convolutional neural network and a morphology-aware generative model. *Computational Materials Science*, 150:212–221, 2018.

[10] G. Casella and R. L. Berger. *Statistical Inference*. Duxbury, Pacific Grove, CA, 2002.

[11] M. Chang, A. L. Dayan, F. Meier, T. L. Griffiths, S. Levine, and A. Zhang. Neural constraint satisfaction: Hierarchical abstraction for combinatorial generalization in object rearrangement. *arXiv preprint arXiv:2303.11373*, 2023.

[12] H. Chen and X. Liu. Geometry enhanced generative adversarial networks for random heterogeneous material representation. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC-21*, Virtual, Online, Aug 2021. ASME.

[13] Q. Chen, J. Wang, P. Pope, W. Chen, and M. Fuge. Inverse design of two-dimensional airfoils using conditional generative models and surrogate log-likelihoods. *Journal of Mechanical Design*, 144(2):021712, 2022.

[14] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

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

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

[17] W. Chen, K. Chiu, and M. Fuge. Aerodynamic design optimization and shape exploration using generative adversarial networks. In *AIAA Scitech 2019 Forum*, page 2351, 2019.

[18] W. Chen and M. Fuge. Béziergan: Automatic generation of smooth curves from interpretable low-dimensional parameters. *arXiv preprint arXiv:1808.08871*, 2018.

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

[20] Y. Cheng, Y. Gong, Y. Liu, B. Song, and Q. Zou. Molecular design in drug discovery: a comprehensive review of deep generative models. *Briefings in bioinformatics*, 22(6):bbab344, 2021.

[21] K. Choi, C. Meng, Y. Song, and S. Ermon. Density ratio estimation via infinitesimal classification. In *International Conference on Artificial Intelligence and Statistics*, pages 2552–2573. PMLR, 2022.

[22] M. Dering, J. Cunningham, R. Desai, M. A. Yukish, T. W. Simpson, and C. S. Tucker. A physics-based virtual environment for enhancing the quality of deep generative designs. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 51753, page V02AT03A015. American Society of Mechanical Engineers, 2018.

[23] S. Deshpande and A. Purwar. Computational creativity via assisted variational synthesis of mechanisms using deep generative models. *Journal of Mechanical Design*, 141(12), 2019.

[24] S. Deshpande and A. Purwar. An Image-Based Approach to Variational Path Synthesis of Linkages. *Journal of Computing and Information Science in Engineering*, 21(2), 10 2020. 021005.

[25] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021.

[26] M. Elfeki, C. Couprie, M. Riviere, and M. Elhoseiny. Gdpp: Learning diverse generations using determinantal point processes. In *International conference on machine learning*, pages 1774–1783. PMLR, 2019.

[27] C. Esteban, S. L. Hyland, and G. Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[28] G. Giannone and F. Ahmed. Diffusing the optimal topology: A generative optimization approach. *arXiv preprint arXiv:2303.09760*, 2023.

[29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, NIPS'14, pages 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[30] M. Gutmann and 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*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[31] J. B. Hamrick, K. R. Allen, V. Bapst, T. Zhu, K. R. McKee, J. B. Tenenbaum, and P. W. Battaglia. Relational inductive bias for physical construction in humans and machines. *arXiv preprint arXiv:1806.01203*, 2018.

[32] A. Heyrani Nobari, W. W. Chen, and F. Ahmed. RANGE-GAN: Design Synthesis Under Constraints Using Conditional Generative Adversarial Networks. *Journal of Mechanical Design*, pages 1–16, 09 2021.

[33] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[34] F. Imrie, A. R. Bradley, M. van der Schaar, and C. M. Deane. Deep generative models for 3d linker design. *Journal of chemical information and modeling*, 60(4):1983–1995, 2020.

[35] M. Kaya and H. Ş. Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

[36] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[37] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[38] J.-W. Lee, N. H. Goo, W. B. Park, M. Pyo, and K.-S. Sohn. Virtual microstructure design for steels using generative adversarial networks. *Engineering Reports*, 3(1):e12274, 2021.

[39] B. Li, C. Huang, X. Li, S. Zheng, and J. Hong. Non-iterative structural topology optimization using deep learning. *Computer-Aided Design*, 115:172–180, 2019.

[40] R. Li, Y. Zhang, and H. Chen. Learning the aerodynamic design of supercritical airfoils through deep reinforcement learning. *AIAA Journal*, pages 1–14, 2021.

[41] X. Li, S. Ning, Z. Liu, Z. Yan, C. Luo, and Z. Zhuang. Designing phononic crystal with anticipated band gap through a deep learning based data-driven method. *Computer Methods in Applied Mechanics and Engineering*, 361:112737, 2020.

[42] S. Liu, Z. Zhong, A. Takbiri-Borujeni, M. Kazemi, Q. Fu, and Y. Yang. A case study on homogeneous and heterogeneous reservoir porous media reconstruction by using generative adversarial networks. *Energy Procedia*, 158:6164–6169, 2019.

[43] Z. Liu, L. Raju, D. Zhu, and W. Cai. A hybrid strategy for the discovery and design of photonic structures. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 10(1):126–135, 2020.

[44] M. Malviya. A systematic study of deep generative models for rapid topology optimization. 2020.

[45] F. Mazé and F. Ahmed. Diffusion models beat gans on topology optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, Washington, DC, 2023.

[46] O. Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[47] S. Mohamed and B. Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

[48] L. Mosser, O. Dubrule, and M. J. Blunt. Reconstruction of three-dimensional porous media using generative adversarial neural networks. *Physical Review E*, 96(4):043309, 2017.

[49] R. K. Mothilal, A. Sharma, and C. Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 607–617, 2020.

[50] A. Müller. Integral probability metrics and their generating classes of functions. *Advances in applied probability*, 29(2):429–443, 1997.

[51] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

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

[53] A. H. Nobari, W. Chen, and F. Ahmed. Pcdgan: A continuous conditional diverse generative adversarial network for inverse design. *arXiv preprint arXiv:2106.03620*, 2021.

[54] A. H. Nobari, W. Chen, and F. Ahmed. Range-constrained generative adversarial network: Design synthesis under constraints using conditional generative adversarial networks. *Journal of Mechanical Design*, 144(2), 2022.

[55] S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

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

[57] S. Oh, Y. Jung, I. Lee, and N. Kang. Design automation by integrating generative adversarial networks and topology optimization. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 51753, page V02AT03A008. American Society of Mechanical Engineers, 2018.

[58] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[59] W. Para, S. Bhat, P. Guerrero, T. Kelly, N. Mitra, L. J. Guibas, and P. Wonka. Sketchgen: Generating constrained cad sketches. *Advances in Neural Information Processing Systems*, 34:5077–5088, 2021.

[60] B. Poole, S. Ozair, A. Van Den Oord, A. Alemi, and G. Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

[61] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[62] S. Rawat and M. H. Shen. Application of adversarial networks for 3d structural topology optimization. Technical report, SAE Technical Paper, 2019.

[63] L. Regenwetter and F. Ahmed. Design target achievement index: A differentiable metric to enhance deep generative models in multi-objective inverse design. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 86236, page V03BT03A046. American Society of Mechanical Engineers, 2022.

[64] L. Regenwetter, B. Curry, and F. Ahmed. BIKED: A dataset and machine learning benchmarks for data-driven bicycle design. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC-21*, Virtual, Online, Aug 2021. ASME.

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

[66] L. Regenwetter, A. Srivastava, D. Gutfreund, and F. Ahmed. Beyond statistical similarity: Rethinking metrics for deep generative models in engineering design. *arXiv preprint arXiv:2302.02913*, 2023.

[67] L. Regenwetter, C. Weaver, and F. Ahmed. Framed: Data-driven structural performance analysis of community-designed bicycle frames, 2022.

[68] B. Rhodes, K. Xu, and M. U. Gutmann. Telescoping density-ratio estimation. *Advances in neural information processing systems*, 33:4905–4916, 2020.

[69] R. Riochet, M. Y. Castro, M. Bernard, A. Lerer, R. Fergus, V. Izard, and E. Dupoux. Intphys: A framework and benchmark for visual intuitive physics reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[70] S. J. Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

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

[72] A. Seff, W. Zhou, N. Richardson, and R. P. Adams. Vitruvion: A generative model of parametric cad sketches. *arXiv preprint arXiv:2109.14124*, 2021.

[73] S. Sharma and A. Purwar. Path synthesis of defect-free spatial 5-ss mechanisms using machine learning. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 83990, page V010T10A034. American Society of Mechanical Engineers, 2020.

[74] C. Sharpe and C. C. Seepersad. Topology design with conditional generative adversarial networks. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 59186, page V02AT03A062. American Society of Mechanical Engineers, 2019.

[75] D. Shu, J. Cunningham, G. Stump, S. W. Miller, M. A. Yukish, T. W. Simpson, and C. S. Tucker. 3d design using generative adversarial networks and physics-based validation. *Journal of Mechanical Design*, 142(7):071701, 2020.

[76] O. Sigmund and K. Maute. Topology optimization approaches: A comparative review. *Structural and Multidisciplinary Optimization*, 48(6):1031–1055, 2013.

[77] A. Sinha, K. Ayush, J. Song, B. Uzkent, H. Jin, and S. Ermon. Negative data augmentation. *arXiv preprint arXiv:2102.05113*, 2021.

[78] K. Smith, L. Mei, S. Yao, J. Wu, E. Spelke, J. Tenenbaum, and T. Ullman. Modeling expectation violation in intuitive physics with coarse probabilistic object representations. *Advances in neural information processing systems*, 32, 2019.

[79] K. A. Smith, L. Mei, S. Yao, J. Wu, E. S. Spelke, J. B. Tenenbaum, and T. D. Ullman. Modeling expectation violation in intuitive physics with coarse probabilistic object representations. In *Neural Information Processing Systems*, 2019.

[80] A. Srivastava, S. Han, K. Xu, B. Rhodes, and M. U. Gutmann. Estimating the density ratio between distributions with high discrepancy using multinomial logistic regression. *Transactions on Machine Learning Research*, 2023.

[81] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 3308–3318. Curran Associates, Inc., 2017.

[82] M. Sugiyama, T. Suzuki, and T. Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[83] M. Sugiyama, T. Suzuki, and T. Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[84] R. K. Tan, N. L. Zhang, and W. Ye. A deep learning–based method for the design of microstructural materials. *Structural and Multidisciplinary Optimization*, 61(4):1417–1438, 2020.

[85] Y. Tang, K. Kojima, T. Koike-Akino, Y. Wang, P. Wu, M. Tahersima, D. Jha, K. Parsons, and M. Qi. Generative deep learning model for a multi-level nano-optic broadband power splitter. In *2020 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3. IEEE, 2020.

[86] S. Valdez, C. Seepersad, and S. Kambampati. A framework for interactive structural design exploration. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC-21*, Virtual, Online, Aug 2021. ASME.

[87] J. Wang, W. W. Chen, D. Da, M. Fuge, and R. Rai. Ih-gan: A conditional generative model for implicit surface-based inverse design of cellular structures. *Computer Methods in Applied Mechanics and Engineering*, 396:115060, 2022.

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

[89] E. Whalen, A. Beyene, and C. Mueller. Simjeb: simulated jet engine bracket dataset. In *Computer Graphics Forum*, volume 40, pages 9–17. Wiley Online Library, 2021.

[90] P. Wollstadt, M. Bujny, S. Ramnath, J. J. Shah, D. Detwiler, and S. Menzel. Carhoods10k: An industry-grade data set for representation learning and design optimization in engineering applications. *IEEE Transactions on Evolutionary Computation*, 26(6):1221–1235, 2022.

[91] T. Xue, T. J. Wallin, Y. Menguc, S. Adriaenssens, and M. Chiaramonte. Machine learning generative models for automatic design of multi-material 3d printed composite solids. *Extreme Mechanics Letters*, 41:100992, 2020.

[92] Z. Yang, X. Li, L. Catherine Brinson, A. N. Choudhary, W. Chen, and A. Agrawal. Microstructural materials design via deep adversarial learning methodology. *Journal of Mechanical Design*, 140(11), 2018.

[93] E. Yilmaz and B. German. Conditional generative adversarial network framework for airfoil inverse design. In *AIAA aviation 2020 forum*, page 3185, 2020.

[94] Y. Yu, T. Hur, J. Jung, and I. G. Jang. Deep learning for determining a near-optimal topological design without any iteration. *Structural and Multidisciplinary Optimization*, 59(3):787–799, 2019.

[95] H. Zhang, L. Yang, C. Li, B. Wu, and W. Wang. Scaffoldgan: Synthesis of scaffold materials based on generative adversarial networks. *Computer-Aided Design*, 138:103041, 2021.

[96] W. Zhang, Z. Yang, H. Jiang, S. Nigam, S. Yamakawa, T. Furuhata, K. Shimada, and L. B. Kara. 3d shape synthesis for conceptual design and optimization using variational autoencoders. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 59186, page V02AT03A017. American Society of Mechanical Engineers, 2019.

A Formulation

Density Ratio Estimation. Consider our formulation:

$$\arg\min_{\theta} \int p_\theta(\mathbf{x}) \left(\frac{1}{2} \left[\log \frac{p_\theta(\mathbf{x})}{n_x(\mathbf{x})} - \log \frac{p_x(\mathbf{x})}{n_x(\mathbf{x})} \right] + \frac{1}{2} \left[\log \frac{p_\theta(\mathbf{x})}{p_x(\mathbf{x})} \right] \right) d\,\mathbf{x}. \tag{6}$$

Our goal is to minimize the Kullback-Leibler (KL) divergence between our model $p_\theta(\mathbf{x})$ and the actual distribution $p_x(\mathbf{x})$, while simultaneously distancing our model $p_\theta(\mathbf{x})$ from any invalid designs represented by $n_x(\mathbf{x})$. Given that we lack access to the explicit functional form of this distribution, we employ density ratio estimation [80, 83] as a means of model learning.

In particular, we use f_ϕ to estimate $r(p_x, p_\theta) = \frac{p_x(\mathbf{x})}{p_\theta(\mathbf{x})}$, f_ψ to estimate $r(p_\theta, n_x) = \frac{p_\theta(\mathbf{x})}{n_x(\mathbf{x})}$, and f_ξ to estimate $r(p_x, n_x) = \frac{p_x(\mathbf{x})}{n_x(\mathbf{x})}$.

Leveraging the connection between density ratio and probabilistic classification [83], we can write (assuming balanced classes):

$$r(\mathbf{x}) = \frac{p_x(\mathbf{x})}{n_x(\mathbf{x})} = \frac{p_x(\mathbf{y}|\mathbf{x})}{n_x(\mathbf{y}|\mathbf{x})} = \frac{p_x(\mathbf{y}|\mathbf{x})}{1 - p_x(\mathbf{y}|\mathbf{x})}, \tag{7}$$

where given a sample $\mathbf{x}$, $p_x(\mathbf{y}|\mathbf{x})$ represents the probability of it being a valid design, whereas $n_x(\mathbf{y}|\mathbf{x})$ signifies the probability of it constituting an invalid design within the framework of a binary classifier. Notice that $n_x(\mathbf{y}|\mathbf{x}) = 1 - p_x(\mathbf{y}|\mathbf{x})$. We can apply the same reasoning to the other two ratios.

In situations where $p_x(\mathbf{x})$ and $n_x(\mathbf{x})$ cannot be quickly evaluated but we can easily collect samples from them, we can resort to directly estimating the ratios r_ϕ, r_ψ, r_ξ using discriminative models to estimate the class probability. This approach is facilitated by employing the following identity:

$$p_x(\mathbf{y}|\mathbf{x}) = \sigma(\log r(\mathbf{x})). \tag{8}$$

We see that there is a direct correspondence between the density ratio of the two distributions and the valid class probability. The following is a natural parameterization for the density ratio estimators:

$$\begin{aligned} f_\phi(\mathbf{x}; p_x, p_\theta) &= \sigma(\log r_\phi(\mathbf{x})) \\ f_\psi(\mathbf{x}; p_\theta, n_x) &= \sigma(\log r_\psi(\mathbf{x})) \\ f_\xi(\mathbf{x}; p_x, n_x) &= \sigma(\log r_\xi(\mathbf{x})), \end{aligned} \tag{9}$$

to estimate the class probability or equivalenty $f_\phi(\mathbf{x}) = \log r_\phi(\mathbf{x})$ to estimate the logits. Learning the density ratio estimators can be performed by binary cross-entropy:

$$\begin{aligned} \mathcal{F}_\phi(\mathbf{x}; \theta) &= \mathbb{E}_{p_x(\mathbf{x})} \log \left[f_\phi(\mathbf{x}) \right] + \mathbb{E}_{p_\theta(\mathbf{x})} \log \left[1 - f_\phi(\mathbf{x}) \right] \\ &= \mathbb{E}_{p_x(\mathbf{x})} \log \left[\sigma(\log r_\phi(\mathbf{x})) \right] + \mathbb{E}_{p_\theta(\mathbf{x})} \log \left[1 - \sigma(\log r_\phi(\mathbf{x})) \right]. \end{aligned} \tag{10}$$

The density ratio can be estimated by sampling from $p_x(\mathbf{x})$ and $p_\theta(\mathbf{x})$, and subsequently learning a discriminator f_ϕ using these samples. In practice, p_θ is learned leveraging adversarial training [29] and an auxiliary set of classifiers to push away samples from n_x. Additionally, we use parameter sharing between f_ψ and f_ξ to help the discriminator learn the difference between valid and invalid samples early during training.

$$\begin{aligned} &\max_{\phi} \mathcal{F}_\phi(\mathbf{x}; \theta) \\ &\max_{\psi, \xi} \mathcal{F}_\psi(\mathbf{x}; \theta) + \mathcal{F}_\xi(\mathbf{x}; \theta) \\ &\min_{\theta} \mathcal{F}_\phi(\mathbf{x}; \theta) - \mathcal{F}_{\psi,\xi}(\mathbf{x}; \theta). \end{aligned} \tag{11}$$

By employing this formulation during training, we strive to push r_ϕ towards 1, thereby maximizing entropy, while encouraging r_ψ and r_ξ to be large and equal, consequently minimizing entropy. It's crucial to note that in the absence of parameter sharing, it's not strictly necessary to jointly train $\mathcal{F}_\xi$. If we manage to learn a robust model using ϕ and θ, then $p\theta$ approximates p_x, and we can confidently rely on $\mathcal{F}_\psi$ for constraint satisfaction.

Nonetheless, based on empirical evidence, we observed improved performance and training stability, particularly during the early training stage, when we share weights and supply the auxiliary discriminator with valid and generated samples. This procedure assists the discriminator ϕ in differentiating invalid from generated samples while simultaneously situating the generated samples within the validity region.

By implementing this method, we learn a generative model that produces samples closely aligned with the training distribution of valid samples and distant from the invalid distribution. This yields a generative model that respects constraints.

λ Variant. We also consider a simpler variant of this approach:

$$\arg\min_{\theta} \int p_\theta(\mathbf{x}) \left(-\lambda \left[\log \frac{p_\theta(\mathbf{x})}{n_x(\mathbf{x})} \right] + \left[\log \frac{p_\theta(\mathbf{x})}{p_x(\mathbf{x})} \right] \right) d\mathbf{x}. \tag{12}$$

Given suitable values of λ, we can effectively learn a generative model and aptly differentiate the generated samples from invalid data. Upon testing this formulation with 2D densities, we observed promising results in terms of both coverage and constraint satisfaction, which resulted in a significant reduction in the number of invalid samples—approximately by an order of magnitude. See Algorithm 1 for training details.

Multiclass Discriminator. The problem can also be written as an elegant multiclass classification task [80]. For example, by defining a single multiclass classifier D_ϕ and assigning pseudo-labels 2 to invalid, 1 to valid, and 0 to generated samples, we can write:

$$\mathcal{F}_\phi^{\texttt{MC}}(\mathbf{x};\theta) = \mathbb{E}_{p_x(\mathbf{x})} \log \left[D_\phi(\mathbf{x}) \right] + \mathbb{E}_{n_x(\mathbf{x})} \log \left[D_\phi(\mathbf{x}) \right] + \mathbb{E}_{p_\theta(\mathbf{x})} \log \left[D_\phi(\mathbf{x}) \right]. \tag{13}$$

Under the assumption of one-hot encoding for the classes and the utilization of cross-entropy loss as a scoring mechanism, we can optimize this loss with respect to ϕ, thereby learning effective discriminators between valid, invalid, and generated samples—and minimize it with respect to θ. Refer to Algorithm 2 for detailed training procedures. Given our use of a singular classifier, the discriminator is implicitly estimating all relevant ratios [80]. Consequently, pushing generated samples closer to valid ones also distances them from invalid samples.

We conducted experiments with this formulation on 2D densities and achieved promising results. Nonetheless, in situations involving more intricate distributions and a diverse array of constraints, it is sensible to allocate varying levels of capacity for model learning and constraint satisfaction by employing different discriminators. Moreover, when we seek to extend our methodologies to generative models other than GANs, such as DDPM, it is preferable to instantiate a separate classifier specifically focused on meeting the constraints.

Negative Data Augmentation. Recently, a method to handle negative data leveraging adversarial sampling has been proposed. The idea is to introduce out-of-distribution samples through a negative distribution $n_x(\mathbf{x})$. Negative Data Augmentation (NDA [77]) proposes to learn the following objective:

$$\mathcal{F}_\phi^{\texttt{NDA}}(\mathbf{x};\theta,\lambda) = \mathbb{E}_{p_x(\mathbf{x})} \log \left[f_\phi(\mathbf{x}) \right] + \mathbb{E}_{\lambda p_\theta(\mathbf{x}) + (1-\lambda) n_x(\mathbf{x})} \log \left[1 - f_\phi(\mathbf{x}) \right], \tag{14}$$

and then train a min-max game like in a standard GAN. Following such formulation, we can build a ratio:

$$r^{\texttt{NDA}}(\mathbf{x}) = \frac{p_x(\mathbf{x})}{\lambda p_\theta(\mathbf{x}) + (1-\lambda) n_x(\mathbf{x})}.$$

Dividing by $n_x(\mathbf{x})$ leads to a mixture of two ratios: the ratio of valid to invalid samples, and the ratio of generated to invalid samples. While our formulation shares superficial similarities with the NDA approach, our goals, and underlying assumptions diverge in crucial ways.

NDA aims to incorporate out-of-distribution samples to improve the fit to the training distribution, primarily evaluated through the Fréchet Inception Distance (FID). On the other hand, our method introduces invalid samples that typically meet the in-distribution criterion (meaning they closely align in the sense of the Kullback-Leibler divergence) but fail to comply with a set of constraints we consider significant.

From a distributional perspective, the invalid samples integrated into CS-GAN are fully compatible with the training distribution. They would exhibit a low discrepancy under the training distribution, as any deviations are limited to the boundaries or small regions, like intersections between edges and minor volume fraction violations. Conversely, the negative samples incorporated in NDA are entirely out-of-distribution and, as such, would present a high discrepancy under the training distribution.

NDA is primarily crafted for image-based tasks and places a high priority on fidelity. In contrast, our method is conceived with constrained design in mind, focusing on the satisfaction of engineering requirements while ensuring sufficient mode coverage. In conclusion, NDA operates under the assumption of disjoint spaces between p_x and n_x. However, in engineering design, this assumption cannot be made, as most of the invalid samples will still fall within the in-distribution from a distributional viewpoint.

B Precision vs Diversity in Generative Models

Encouraging generative models to improve their precision can have unintended consequences for recall. One approach to improve recall is to explicitly encourage diversity of generated samples. Diversity is often a desired goal in generative modeling for engineering design applications [15, 16, 63, 66]. As [16] and [15] note, incorporating diversity can also help models generalize and avoid mode collapse. Diversity was first explicitly incorporated into deep generative models for design in [16] using a Determinantal Point Process (DPP). Determinantal Point Process (DPP) based diversity measures have been used in a variety of generative applications in design [16, 53] and elsewhere [26, 49].

The DPP loss is calculated using a positive semi-definite DPP kernel S. Entries of this matrix are calculated using some modality- and problem-dependent similarity kernel, such as the Euclidean distance kernel. The $(i, j)^{th}$ element of S can be expressed in terms of the similarity kernel k and samples x_i and x_j as:

$$S_{i,j} = k(x_i, x_j),$$

and the loss as:

$$L_{div} = -\frac{1}{B}\,\text{logdet}(S) = -\frac{1}{B}\sum_{i=1}^{B}\log\lambda_i,$$

where λ_i is the i-th eigenvalue of L and B is the number of samples in the batch. Adding this loss helps the generative model achieve better coverage, an observation supported by our experiments too.

C Experimental Details and Additional Experiments

C.1 2d densities

Setup and Training. 10k positive samples and 10k negative samples are randomly generated for each test problem. All tested networks (encoder, decoder, generator, auxiliary discriminator) are deep networks with two hidden layers of 400 neurons each, ReLU activations. A batch size of 256 is used throughout. Models are trained using the Adam optimizer [36] with a learning rate $3e^{-4}$ and $5e^{-4}$ for GANs and VAEs respectively. Models are trained for 100k steps. Diversity weight γ is set at 0.7 for GANs and 0.05 for VAEs. The noise dimension for the GAN is set at 2, while the latent dimension for the VAE is set at 16. The VAE's KL divergence loss term is weighted at 0.05. The VAE's auxiliary classifier is pretrained and validity weight parameter λ is set at 0.2. The GAN's validity weight parameter λ is set at 0.4.

In the main paper, we train a GAN on 2D densities using the variant in Equation 12, which we label the CS-GAN (or CSD-GAN with diversity). Psuedocode for this training procedure is shown below.

Algorithm 1 CSD-GAN Training Procedure (used in paper results)

while $step \leq n_{steps}$ **do**
  Sample $V_{batch} \sim V_{dataset}$ and $I_{batch} \sim I_{dataset}$
  Sample $\epsilon \sim N(0, 1)$
  G_{batch} = Generator(ϵ)
  $D^V_{preds} =$ Discriminator(V_{batch})
  $D^G_{preds} =$ Discriminator(G_{batch})
  $A^I_{preds} =$ Aux_Discriminator(I_{batch})
  $A^G_{preds} =$ Aux_Discriminator(G_{batch})
  loss_fn $=$ Binary Cross Entropy()
  $D\_loss =$ loss_fn(D^G_{preds}, 0) + loss_fn(D^V_{preds}, 1)
  $A\_loss =$ loss_fn(A^G_{preds}, 0) + loss_fn(A^I_{preds}, 1)
  $Diversity_loss =$DPP Diversity(G_{batch})
  $G\_loss =$ loss_fn(D^G_{preds}, 1) - $\lambda \cdot$ loss_fn(A^G_{preds}, 1) + $\gamma \cdot Diversity\_loss$
  Optimize(Discriminator, D_loss)
  Optimize(Aux_Discriminator, A_loss)
  Optimize(Generator, G_loss)
  $step = step + 1$
end while

Following Equation 6, we can also train the GAN using a multiclass discriminator, which we call MC-CS-GAN or MC-CSD-GAN. In practice, we find that the previous formulation yields slightly superier results on 2D densities. However, we include pseudocode for this formulation and results on 2D densities later.

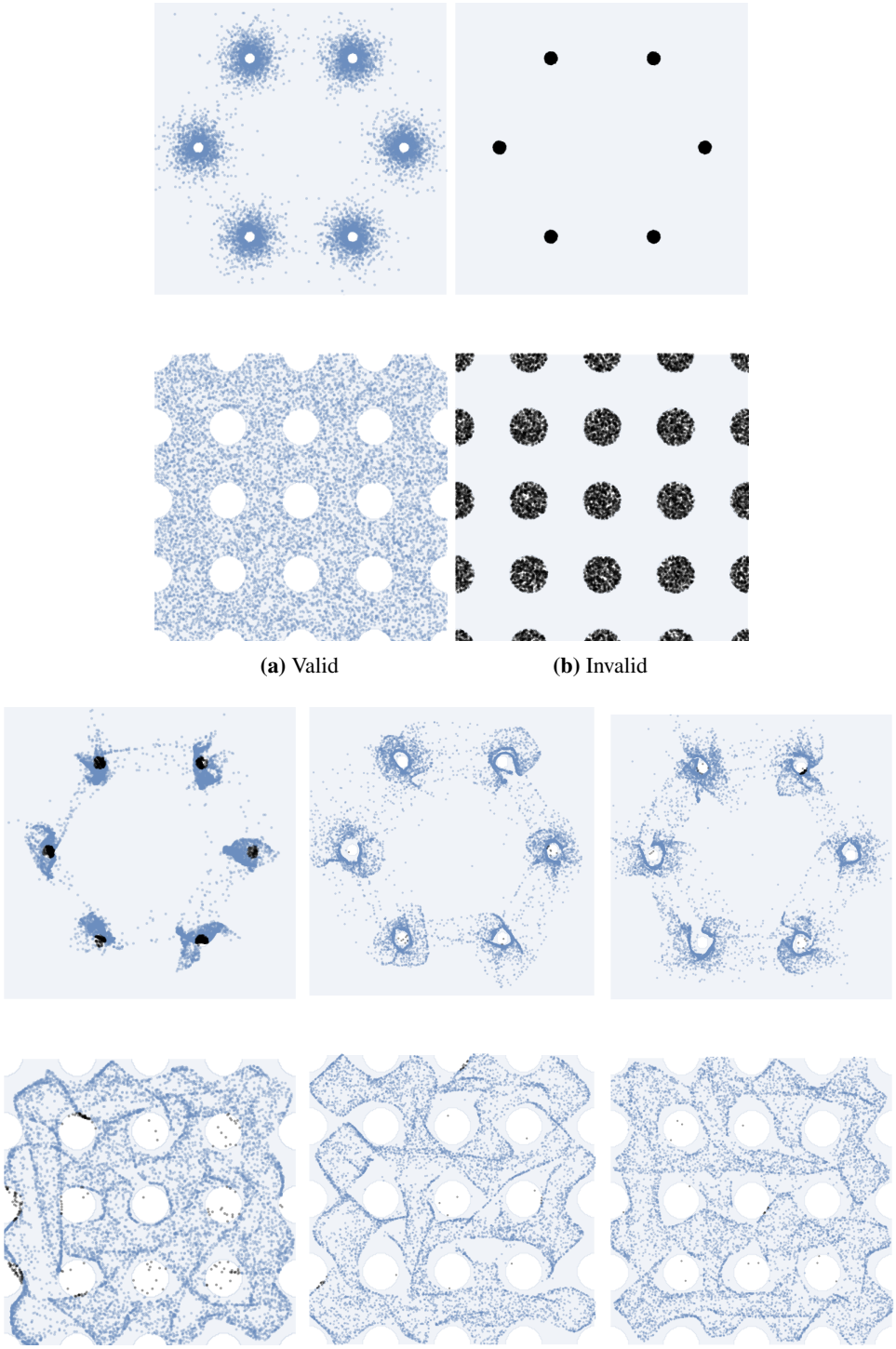
Algorithm 2 Multiclass CSD-GAN Training Procedure

while $step \leq n_{steps}$ **do**
  Sample $V_{batch} \sim V_{dataset}$ and $I_{batch} \sim I_{dataset}$
  Sample $\epsilon \sim N(0, 1)$
  $G_{batch} =$ Generator(ϵ)
  $D^V_{preds} =$ Discriminator(V_{batch})
  $D^I_{preds} =$ Discriminator(I_{batch})
  $D^G_{preds} =$ Discriminator(G_{batch})
  loss_fn $=$ Categorical Cross Entropy()
  $D\_loss =$ loss_fn(D^G_{preds}, 0) + loss_fn(D^V_{preds}, 1) loss_fn(D^I_{preds}, 2)
  $Diversity_loss =$DPP Diversity(G_{batch})
  $G\_loss =$ loss_fn(D^G_{preds}, 1) + $\gamma \cdot Diversity\_loss$
  Optimize(Discriminator, D_loss)
  Optimize(Generator, G_loss)
  $step = step + 1$
end while

We include an set of results on 2D density experiments expanding on our results from the main paper. We compare our models trained using Algorithm 1 (CS-GAN, CSD-GAN – shown in the original paper), Algorithm 2 (MC-CS-GAN, MC-CSD-GAN), as well as VAE variants (VAE, CS-VAE, and CSD-VAE). Mean scores and standard deviations are reported over 3-5 instantiations in Table 6.

Table 7: Extended results for 2D density experiments. Mean scores and standard deviations over three to five instantiations are shown. Best models are determined using a two-sample t-test with 98% confidence.

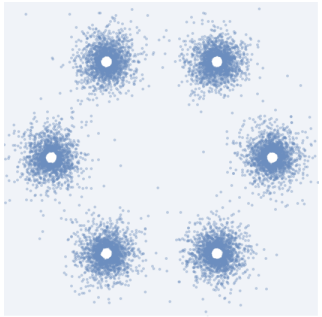
	GAN	CS-GAN	CSD-GAN	MC-CS-GAN	MC-CSD-GAN	VAE	CS-VAE	CSD-VAE
Problem 1								
↓ **NDP**	0.014±0.002	**0.012±0.001**	**0.013±0.000**	**0.012±0.001**	**0.013±0.001**	0.024±0.001	**0.020±0.001**	0.025±0.001
↓ **NGS**	0.034±0.004	0.018±0.001	**0.015±0.001**	0.020±0.002	0.018±0.001	**0.013±0.000**	0.051±0.004	0.043±0.005
↓ **MMD**	0.006±0.004	**0.002±0.002**	**0.001±0.000**	**0.002±0.002**	**0.001±0.001**	**0.003±0.000**	0.005±0.000	0.005±0.001
↑ **F1**	0.702±0.045	0.926±0.009	**0.944±0.009**	0.909±0.009	0.931±0.007	**0.840±0.011**	0.814±0.017	0.799±0.012
↑ **F10**	0.956±0.018	**0.994±0.002**	**0.995±0.001**	0.989±0.007	**0.994±0.001**	**0.956±0.002**	0.945±0.008	0.936±0.004
↑ **F0.1**	0.904±0.024	**0.994±0.004**	**0.995±0.001**	**0.995±0.000**	**0.995±0.001**	**0.989±0.001**	**0.971±0.026**	**0.983±0.008**
↑ **AUC-PR**	0.765±0.048	**0.982±0.005**	**0.990±0.003**	0.976±0.006	0.985±0.001	**0.915±0.008**	0.872±0.025	0.858±0.018
↑ **Validity**	0.889±0.035	0.993±0.004	**0.995±0.004**	**0.998±0.000**	**0.996±0.002**	0.881±0.007	**0.999±0.000**	**0.999±0.000**
↑ **NIS**	0.146±0.021	0.152±0.010	0.161±0.002	0.152±0.009	**0.169±0.004**	0.191±0.004	0.193±0.008	**0.237±0.009**
Problem 2								
↓ **NDP**	**0.017±0.001**	**0.016±0.000**	**0.016±0.000**	**0.016±0.000**	**0.016±0.000**	0.022±0.000	**0.016±0.000**	**0.016±0.000**
↓ **NGS**	0.032±0.008	0.025±0.002	**0.019±0.000**	0.031±0.003	0.024±0.001	**0.019±0.000**	0.072±0.015	0.059±0.007
↓ **MMD**	0.001±0.001	**0.001±0.000**	**0.001±0.000**	**0.001±0.000**	**0.001±0.000**	**0.001±0.000**	**0.004±0.003**	**0.003±0.002**
↑ **F1**	0.948±0.015	**0.963±0.004**	**0.966±0.005**	0.956±0.000	**0.963±0.007**	**0.960±0.004**	0.870±0.033	0.866±0.024
↑ **F10**	0.996±0.001	**0.998±0.000**	**0.998±0.000**	**0.997±0.001**	**0.998±0.001**	**0.998±0.000**	0.994±0.002	0.993±0.002
↑ **F0.1**	0.996±0.003	**0.998±0.000**	**0.998±0.001**	0.997±0.000	0.997±0.000	**0.998±0.000**	0.967±0.024	0.960±0.023
↑ **AUC-PR**	**0.991±0.005**	**0.996±0.001**	**0.996±0.001**	0.994±0.000	**0.995±0.001**	**0.995±0.001**	0.939±0.034	0.933±0.027
↑ **Validity**	0.969±0.009	**0.999±0.001**	**0.998±0.001**	**0.998±0.001**	**0.998±0.001**	0.897±0.003	**0.995±0.002**	**0.994±0.004**
↑ **NIS**	0.168±0.007	**0.176±0.005**	0.168±0.001	**0.176±0.002**	**0.174±0.006**	0.172±0.002	**0.212±0.010**	0.197±0.007

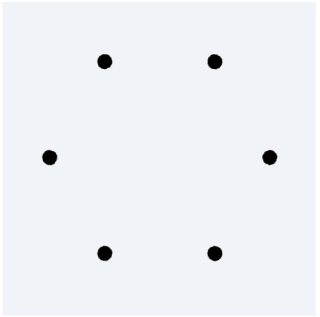


(a) Valid **(b)** Invalid

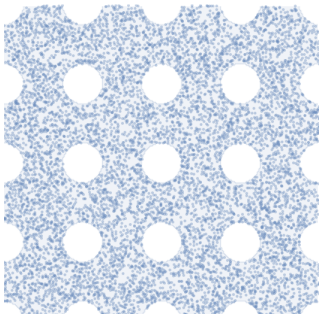
(c) GAN **(d)** MC-CS-GAN **(e)** MC-CSD-GAN

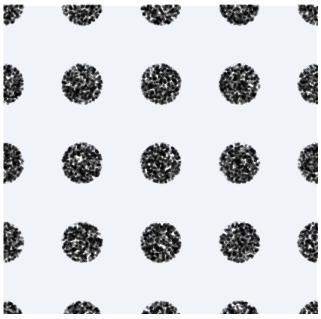
Figure 5: Generated Distributions from the multiclass variants of CS-GAN and CSD-GAN (MC-CS-GAN and MC-CSD-GAN). Though the multiclass variants have similar constraint satisfaction, they do not achieve quite as strong distributional similarity.

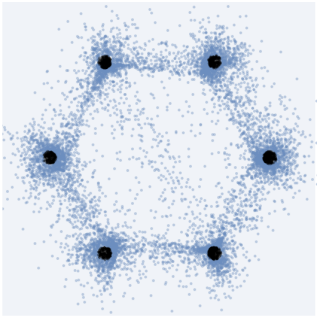


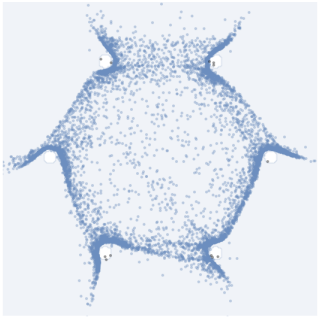


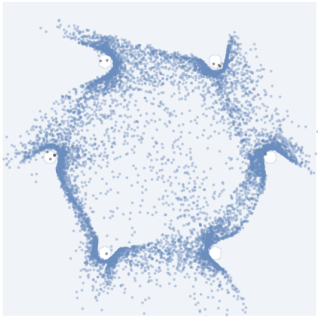
(a) Valid **(b)** Invalid

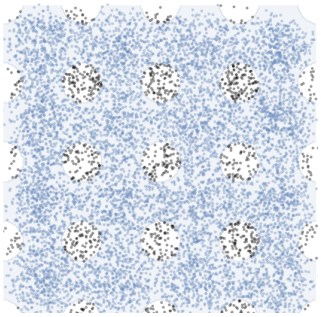


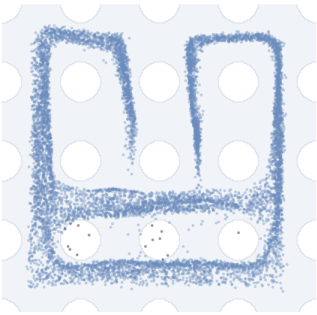


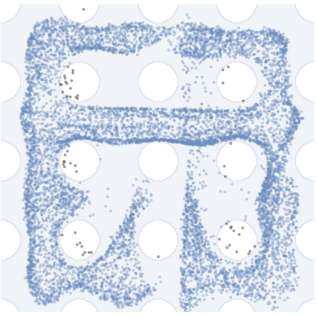












(c) VAE **(d)** CS-VAE **(e)** CSD-VAE

Figure 6: Distributions generated by VAE variants. Though the auxiliary classifier encourages significantly higher constraint satisfaction, the VAE struggles with coverage. Though diversity improves coverage, distrubutional similarity scores are still poorer than the vanilla VAE's.

C.2 Blocks Experiments

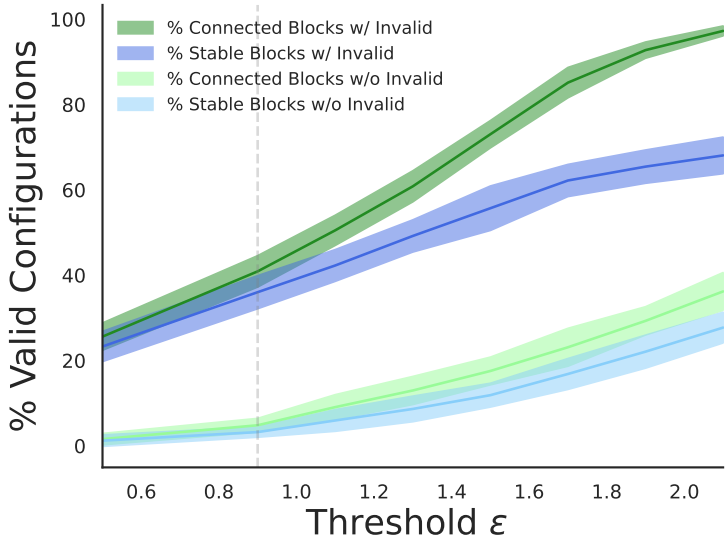


Figure 7: Constraint Satisfaction w/ and w/o leveraging Invalid Data during training for the block problem. The plot shows the % of generated samples that fulfill one (Connected) or both (Stable) global constraints increasing the constraint tolerance (reducing precision). We can see that leveraging Invalid Data improve constraint satisfaction by many folds, corroborating the hypothesis that divergence-based training using invalid data pushes the generated samples away from such regions.

Table 8: Handling Multiple Sets of Constraints. Valid data is Connected (constraint set I) and Stable (constraint set II). Invalid data is composed of two Invalid sets: Connected Unstable and Disconnected Stable. With pseudo we mean a structure that would be Stable if the other constraints were satisfied.

	GAN (w/o Invalid)	CS-GAN (w/ Invalid)
↑ `Connected`(y_b, y_m) (Ia)	21.13 ± 3.04	**100** ± 0.00
↑ `Connected`(y_m, y_t) (Ib)	22.68 ± 3.06	**41.10** ± 4.79
↑ `Stable` (II)	**83.20** ± 4.09	70.35 ± 3.72
↑ `Connected` (I)	4.83 ± 1.72 (-88.24 %)	**41.10** ± 4.80
↑ `Connected and Stable` (I and II)	3.28 ± 1.34 (- 90.89 %)	**36.02** ± 3.88

D Visualizations

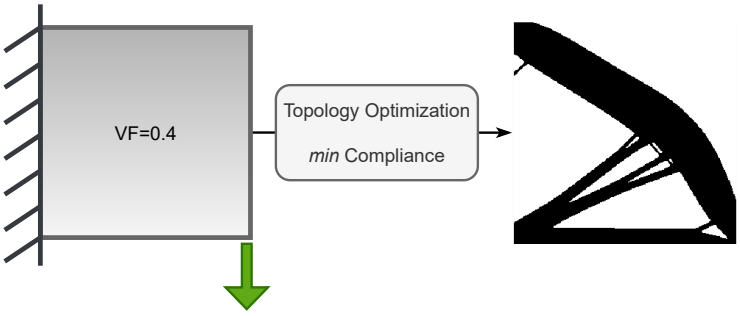


Figure 8: Constrained Optimization problem for engineering design. Given a set of constraints and a performance objective, the goal is to find the optimal design variables wrt the objective that satisfy the constraints over the given domain. In topology optimization, given a set of constraints in the form of loads, boundary conditions, and volume fractions, the goal is to find the design with minimum compliance that satisfies all the given constraints.

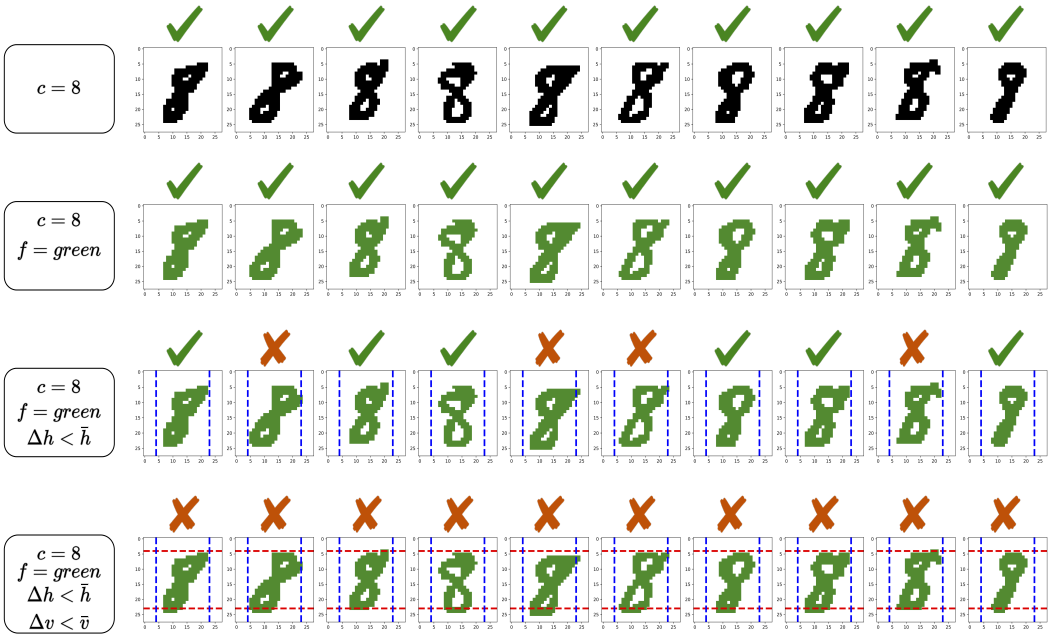


Figure 9: Generative Models easily fulfill simple constraints like class and color. However, as we move to more precise constraints, such as dimensionality, generative models struggle to fulfill the requirements.

D.1 Training Set - Valid

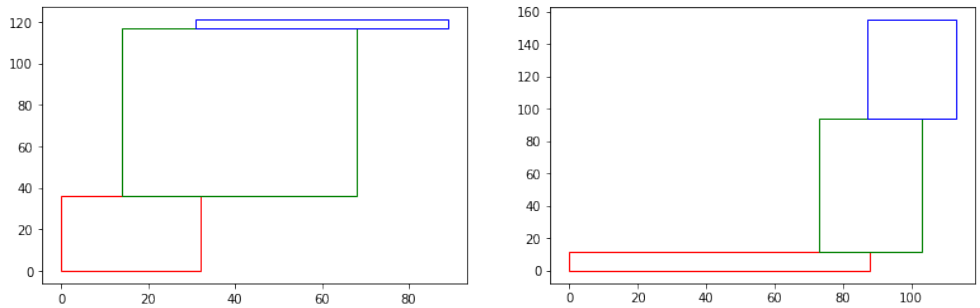


Figure 10: Valid Data for Constraint I. Example of Valid configurations for constraints I (connection). Blocks fulfilling only the connection constraints (I). Specifically, the blocks are stacked one on top of each other without any floating or intersection but in general in unstable configurations. This means that the blocks in the data are arranged in a way that satisfies certain rules or criteria, such as not being allowed to float (i.e., not being fully supported by the blocks below) or intersecting (i.e., overlapping with other blocks).

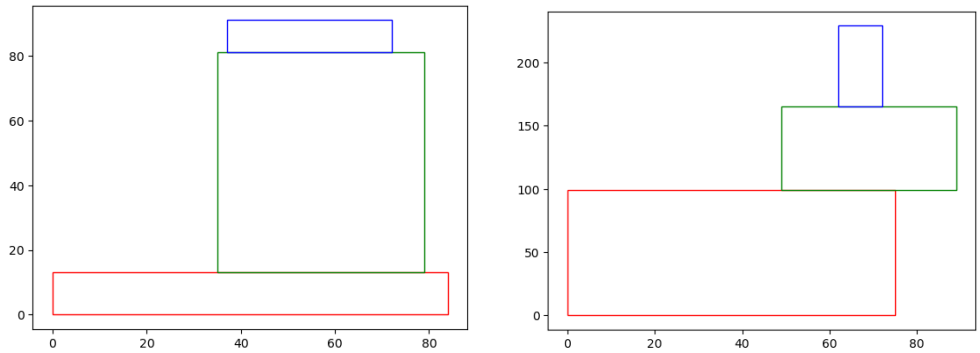


Figure 11: Valid Data for Constraints I+II. Example of Valid configurations for constraints I (connection) and II (stability). Blocks fulfilling both constraints connection constraints (I). Specifically, the blocks are stacked one on top of each other without any floating or intersection in a stable configuration. This means that the blocks in the data are arranged in a way that satisfies certain rules or criteria, such as not being allowed to float (i.e., not being fully supported by the blocks below) or intersecting (i.e., overlapping with other blocks) and having an internal center of mass.

D.2 Training Set - Invalid

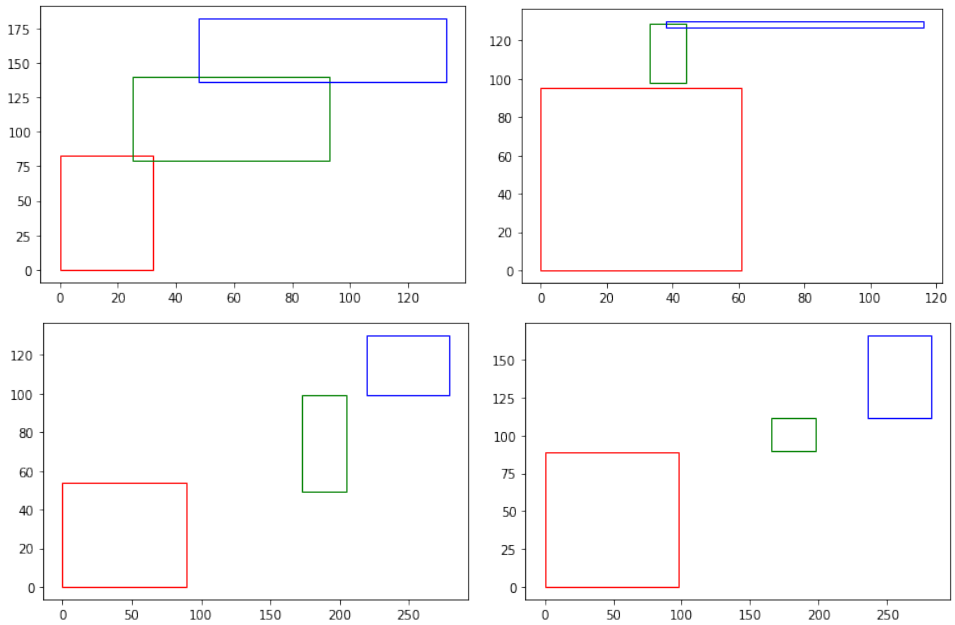


Figure 12: Invalid Data. Example of Invalid data for block stacking. The negative data consists of two categories: hard-invalid (top) and easy-invalid (bottom). The top section of the figure contains examples of hard invalid, which are generated by violating the constraints of the block stacking problem by a small degree (1 to 5 units). The constraints include the requirement that the blocks should not intersect or float above each other. These hard-invalids are designed to be challenging for the model to learn from, as they are close to satisfying the constraints but still violate them. The bottom section of the figure contains examples of easy-invalid, which are generated by violating the constraints by a large degree (1 to 20 units). These examples are easier for the model to learn from, as the violations are more pronounced and easier to detect.

D.3 Samples

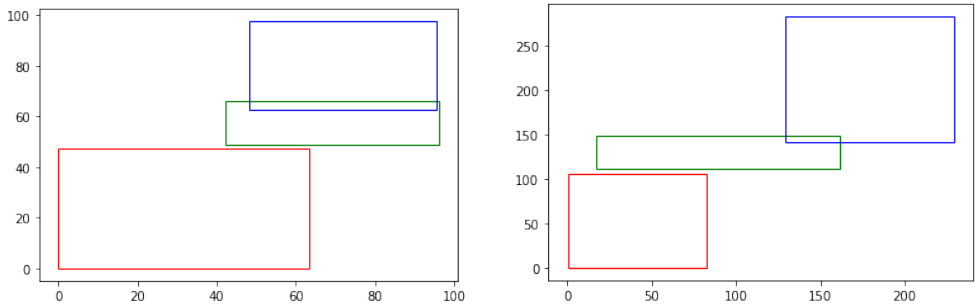


Figure 13: GAN Samples. Samples from a model trained only on valid data. The model generates reasonable samples but in most cases, the constraints are not satisfied. Precision at the boundary is challenging to enforce.

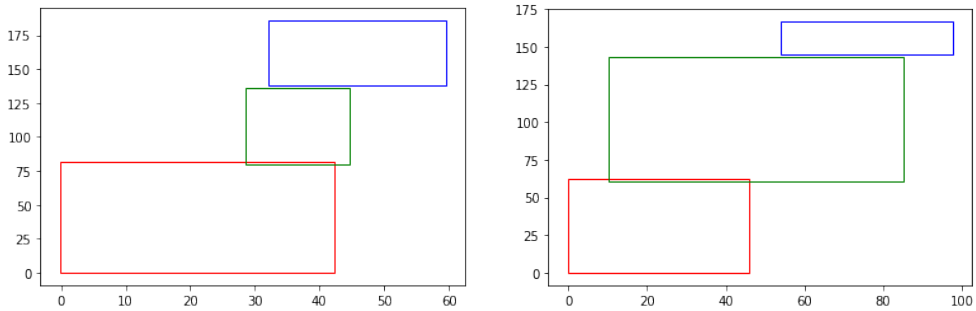


Figure 14: CS-GAN Samples with Invalid Constraints I. Samples from a model trained on valid and invalid data for constraint I (connection) using our divergence formulation. The model generates reasonable samples and in most cases the connectivity constraints are satisfied. The model can generate blocks that do not intersect and float (up to a small tolerance of 0.9 u). However, because we do not rely on invalid data for constraint II (stability), the generated samples do not fulfill this second set of constraints, and the generated blocks are connected but in general unstable. Precision at the boundary is enforced better than training only on valid data.

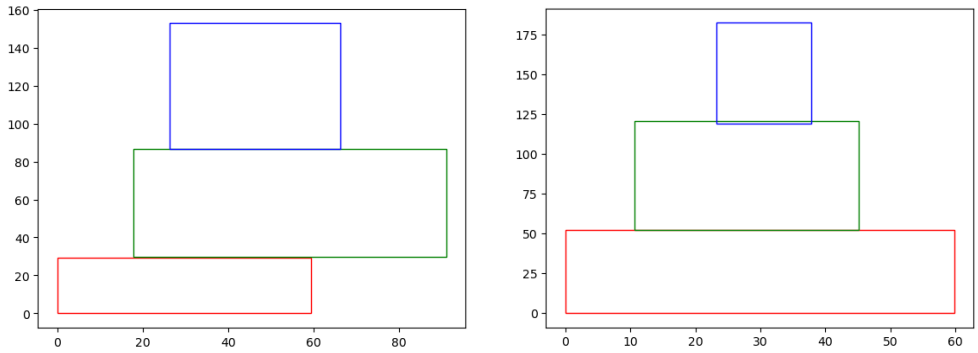


Figure 15: CS-GAN Samples with Invalid Constraints I+II. Samples from a model trained on valid and invalid data for constraint I (connection) and constraint II (stability) using our divergence formulation. The model generates reasonable samples and in most cases the connectivity constraints are satisfied and the blocks are stacked in a stable configuration. The model can generate blocks that do not intersect and float (up to a small tolerance of 0.9 u) and the center of mass is internal to the structure. Because we rely on invalid data for constraint I and II (connectivity and stability), the generated samples do fulfill both sets of constraints, and the generated blocks are connected and stable. Precision at the boundary is enforced better than training only on valid data. This visualization corroborates the need for invalid designs when dealing with constraint satisfaction in generative models.

E Experimental Details for Stacked Blocks

Table 9: Overview. Using Invalid data improves constraint satisfaction by an order of magnitude.

	Valid		Invalid		Metrics		
	Connected	Balanced	Disconnected	Unbalanced	Pseudo-Balanced ↑	Balanced ↑	Connected ↑
Valid Set	✓	✗	✗	✗	2.50 %	-	100.00 %
Valid Set	✓	✓	✗	✗	0.00 %	100.00 %	100.00 %
Invalid Set	✗	✗	✓	✗	0.00 %	100.00 %	0.00 %
Invalid Set	✗	✗	✗	✓	2.50 %	0.00 %	100.00 %

Table 10: Relevant Hyperparameters for baselines and CS-GM. C: connectivity. S: stability. FM: floating material. VFE: volume fraction error. CE: compliance error.

	Blocks (GAN)	Blocks (AR-GAN)	TO (GAN)	TO (DM)
Dimension	12	12	1x64x64	1x64x64
Valid Set	10K	10K	20K	30K
Invalid Set	10K	10K	20K	30K
Evaluation Set	1K	1K	9K	2.8K
Constraints	C+S	C+S	FM + VFE	CE
Generative Architecture	MLP (16-32-12)	LSTM (64-12)	Unet [52]	Unet [51]
Discriminative Architecture	MLP (128-64-32-1)	MLP (128-64-32-1)	Conv [61]	UnetEncoder [51]
Batch size	200	200	256	64
Iterations	30K	50K	100K	180K
Learning rate	$1e^{-3}$	$1e^{-3}$	$1e^{-4}$	$2e^{-4}$
Optimizer	Adam	Adam	Adam	Adam