Constraining Generative Models for Engineering Design with Negative Data

Anonymous authors
Paper under double-blind review

Abstract

Generative models have recently achieved remarkable success and widespread adoption in society, yet they still often struggle to generate realistic and accurate outputs. This challenge extends beyond language and vision into fields like engineering design, where safety-critical engineering standards and non-negotiable physical laws tightly constrain what outputs are considered acceptable. In this work, we introduce two approaches to guide models toward constraint-satisfying outputs using 'negative data' – examples of what to avoid. Our negative data generative models (NDGMs) outperform state-of-the-art NDGMs by 4x in constraint satisfaction and easily outperform classic generative models using 8x less data in certain problems. To demonstrate this, we rigorously benchmark our NDGMs against 14 baseline models across numerous synthetic and real engineering problems, such as ship hulls with hydrodynamic constraints and vehicle design with impact safety constraints. Our benchmarks showcase both the best-in-class performance of our new NDGM models and the widespread dominance of NDGMs over classic generative models in general. In doing so, we advocate for the more widespread use of NDGMs in engineering design tasks.

1 Introduction

Generative models have demonstrated impressive results in vision, language, and speech. However, even with massive datasets, they struggle with precision, generating physically impossible, factually incorrect, or otherwise 'invalid' samples. Most users can easily point to examples: Anatomical inaccuracies, imbalanced objects in natural scenes, erroneous text responses, etc. This invalidity can be thought of as a form of constraint violation – in the ideal scenario, generative models would be constrained to only generate valid samples. While this constraint violation 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. A generative model synthesizing designs for car or airplane components, for example, may be subject to geometric restrictions (such as disconnected or colliding components), functional requirements (such as load-bearing capacity or maximum weight), industry standards, and manufacturing limitations. As generative models are increasingly applied to engineering problems, their blatant violation of ubiquitous, objective, and non-negotiable constraints is becoming increasingly problematic.

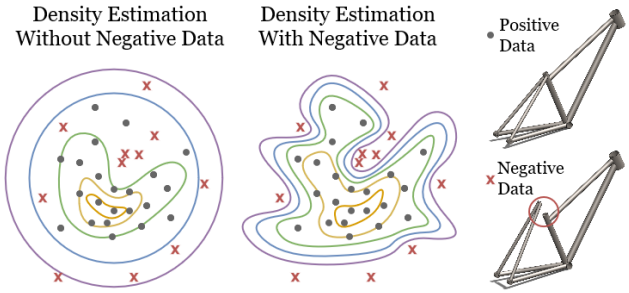


Figure 1: Many real-world data distributions have gaps in their support caused by constraints. Generative models classically estimate these distributions using constraint-satisfying (positive) samples. We analyze training methods for generative models that additionally leverage constraint-violating (negative) data to more accurately estimate the density of in-distribution (positive) data. For example, by examining bike frames with disconnected components, a model can better learn to generate geometrically valid frames.

We hypothesize that challenges with constraint violation in generative models are largely attributable to the fact that generative models are classically shown only 'positive' (constraint-satisfying) data points during training, and are never exposed to 'negative' (constraint-violating) data points to avoid. Completely satisfying constraints using this training approach is equivalent to learning a binary classification problem with only

one class present in the data, a challenging task. Instead, by studying negative data in addition to positive data, generative models can better avoid constraint-violating samples during generation (Fig. 1). This aligns with their distribution-matching objective since negative data points should have zero density in the original real-world distribution that the model is trying to mimic. We will refer to these models as ***Negative-Data Generative Models, or NDGMs***.

We conceptualize and test two new NDGM formulations. These formulations dominate both simple baselines and the current state-of-the-art (SOTA) NDGMs on highly non-convex test problems and more complex engineering problems. In certain tests, our NDGMs generate 4x as many valid (positive) samples as SOTA models. In benchmarking 16 training formulations over 15 test problems, we additionally identify several broader conclusions. (1) Unlike our new NDGMs, we find that existing formulations for NDGMs sometimes fail to surpass simple baselines. In particular, while SOTA models excel in simpler problems, they falls short of our method in more complex problems. (2) We find that NDGMs in general dominate conventional models. We therefore advocate for more the widespread use of NDGMs (including existing ones) over vanilla models. (3) We demonstrate that negative data can be significantly more informative than positive data. In some problems, we achieve a 40x improvement in constraint satisfaction by augmenting the dataset by 6% using negative data, indicating that NDGMs often improve sample efficiency over conventional models.

Contributions:

- **(i)** We introduce two "Negative-Data Generative Models" (NDGMs) which tie or outperform the existing SOTA in constraint satisfaction in all synthetic problems and the five most chellenging engineering problems tested.
- **(ii)** We curate an extensive set of benchmarks comprised of several synthetic problems and a dozen engineering tasks, featuring real constraints from engineering standards. We test 16 different NDGM formulations and baselines on these tests, the largest benchmark of NDGMs to our knowledge.
- **(iii)** We demonstrate that simple baselines dominate conventional generative models in constraint satisfaction. In fact, we show that the SOTA NDGMs cannot consistently outperform simple baselines when dealing with fine-grained constraints, even in low-dimensional settings.
- **(iv)** We show that NDGMs can significantly outperform vanilla models using ($\sim 90\%$) less data. We thus advocate for the more widespread use of NDGMs in engineering tasks over vanilla counterparts.

2 Background

In this section, we discuss constraint satisfaction in generative models and then discuss divergence minimization in generative models. For more related work, see Appendix A.

2.1 Constraints in Engineering and Design

Constraints are ubiquitous in design. A designer creating ship hulls, for example, must adhere to a medley of geometric constraints, performance requirements, and safety regulations from authoritative bodies. Generating constraint-satisfying designs can be exceedingly difficult. As many practitioners turn to data-driven generative models to tackle engineering problems (Regenwetter et al., 2022a), this difficulty remains (Woldseth et al., 2022; Regenwetter et al., 2023) (as we demonstrate, even a generative model that sees 30k examples of valid ship hulls can only generate valid hulls with a 2% success rate).

The overwhelming majority of deep generative models in design do not actively consider constraints (Woldseth et al., 2022; Regenwetter et al., 2022a), despite constraint satisfaction being an explicit goal in many of the design problems they address (Oh et al., 2019; Nie et al., 2021; Bilodeau et al., 2022; Chen et al., 2022; Chen & Fuge, 2019; Cheng et al., 2021). Several engineering design datasets (Regenwetter et al., 2022b; Bagazinski & Ahmed, 2023; Giannone & Ahmed, 2023; Mazé & Ahmed, 2023) feature constraint-violating designs, and many others have checks for validity (Whalen et al., 2021; Wollstadt et al., 2022), allowing datasets of invalid (negative) designs to be generated. In some cases, datasets of positive examples are generated through search by rejecting and discarding negative samples (Bagazinski & Ahmed, 2023; Regenwetter et al., 2022b), making negative data essentially free. In any problem where negative data is available or can be generated, NDGMs can be applied.

2.2 Divergence Minimization in Generative Models

Before discussing divergence minimization in NDGMs, we first discuss divergence minimization in conventional generative models. Let $p_p(x)$ be the (positive) data distribution and $p_\theta(x)$ the distribution sampled by the generative model. Given N samples from $p_p(x)$, the objective of generative modeling is to find a setting θ^* of θ, such that, for an appropriate choice of discrepancy measure, $p^*_\theta \approx p_p$. A common choice for this discrepancy measure is the Kullback–Leibler or KL divergence:

$$\mathbb{KL}[p_\theta \| p_p] = \int p_\theta(\boldsymbol{x}) \left[\log \frac{p_\theta(\boldsymbol{x})}{p_p(\boldsymbol{x})}\right] d\boldsymbol{x}. \tag{1}$$

To minimize the discrepancy, we find θ^* as the solution to the following optimization problem:

$$\theta^* = \arg\min_\theta \mathbb{KL}[p_\theta \| p_p]. \tag{2}$$

In practice, direct optimization of equation 2 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 equation 2 (Kingma & Welling, 2013; Burda et al., 2015; Ho et al., 2020; Sønderby et al., 2016) or by using plug-in or direct estimators of the divergence measure (Casella & Berger, 2002; Sugiyama et al., 2012a; Gutmann & Hyvärinen, 2010; Srivastava et al., 2017; Goodfellow et al., 2014; Srivastava et al., 2023; Poole et al., 2019). In both of these cases, under certain conditions, as $N \to \infty$, theoretically, it holds that, $\theta \to \theta^*$. However, since N is limited, 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_p may not have significant empirical support. In domains such as engineering design, where invalid (negative) designs tend to be very close to the valid (positive) 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 (Regenwetter et al., 2023).

Divergence minimization in GANs. Generative Adversarial Networks (GANs) (Goodfellow et al., 2014; Arjovsky et al., 2017; Mohamed & Lakshminarayanan, 2016; Srivastava et al., 2017; Nowozin et al., 2016) are a powerful framework for generating realistic and diverse data samples. GANs have two main components: a generator f_θ, which generates samples according to the density p_θ, and a discriminator f_ϕ, which is a binary classifier. 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:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{p_p(\boldsymbol{x})}[\log f_\phi(\boldsymbol{x})] + \mathbb{E}_{p_\theta(\boldsymbol{x})}[1 - \log(f_\phi(\boldsymbol{x}_\theta))], \tag{3}$$

Training a GAN involves iterating over $\min_\theta \max_\phi \mathcal{L}(\theta, \phi)$. GANs can also be interpreted in terms of estimating the density ratio (Gutmann & Hyvärinen, 2010; Srivastava et al., 2017) between the data and the generated distribution $r(\boldsymbol{x}) = p_p(\boldsymbol{x})/p_\theta(\boldsymbol{x})$. This ratio can be estimated by a discriminative model as $r_\phi = f_\phi(\boldsymbol{x})/(1 - f_\phi(\boldsymbol{x}))$ and $r_\phi = 1$ gives us $p_\theta = p_p$. The optimal discriminator prediction and generator distribution are:

$$f_\phi(\boldsymbol{x}) = \frac{p_p(\boldsymbol{x})}{(p_\theta(\boldsymbol{x}) + p_p(\boldsymbol{x}))}, \; p^*_\theta(\boldsymbol{x}) = p_p(\boldsymbol{x}). \tag{4}$$

Divergence minimization in other generative models. Many other types of generative models similarly minimize divergence between p_θ and p_p. These models include popular likelihood-based models like Variational Autoencoders (VAEs) (Kingma & Welling, 2013) and Denoising Diffusion Probabilistic Models (DDPMs) (Ho et al., 2020). We will not discuss the mathematics behind divergence minimization for these likelihood-based models, but we do benchmark several variants in our results. In general, we refer to unaugmented GANs, VAEs, and DDPMs as 'vanilla models' throughout the paper.

3 Negative-Data Generative Models

In this section, we discuss the NDGM framework. We explain how generative models can be adjusted to exploit negative data to improve constraint satisfaction. Let p_n denote the ***negative distribution*** i.e., the distribution of constraint-violating datapoints. Instead of training using only the positive distribution p_p,

we now seek to train a generative model using both p_p and p_n. In this section, we discuss several existing methods to do so. These methods range from simple baselines like rejection sampling to state-of-the-art formulations like discriminator overloading.

3.1 Class Conditioning (CC)

One approach to incorporating implicit constraints is through conditioning, which is popular in many design generation problems (Nie et al., 2021; Behzadi & Ilieş, 2021; Mazé & Ahmed, 2023; Malviya, 2020; Heyrani Nobari et al., 2021). In conditional modeling, a generative model typically conditions on the constraints denoted as $\boldsymbol{c}$ and learns a conditional distribution, $p(\boldsymbol{x}|\boldsymbol{c})$, where $\boldsymbol{x}$ represents the generated output. 'Off-the-shelf' class-conditional models can be simple NDGMs, where the positive and negative data each constitute one class. During inference, the model attempts to satisfy constraints by generating conditionally positive samples. Other conditional variants such as auxiliary-classifier GANs (AC-GANs) (Odena et al., 2017) can also be used. Broadly speaking, the negative data formulation for generative models can be seen as a specific case of class-conditional generation. However, as we demonstrate in this paper, generic class-conditional training formulations for generative models are not as effective as specialized NDGMs.

3.2 Pre-Trained Classifier (PC)

One common approach for active constraint satisfaction involves pre-training a supervised model to predict constraint satisfaction and querying this model during training, inference, or postprocessing. Often, this model predicts constraint violation likelihood, though it can also predict intermediates that are combined in a more complex constraint check (Wang et al., 2022). Typically, this classifier f_ψ learns:

$$f_\psi(\boldsymbol{x}) = \frac{p_n(\boldsymbol{x})}{p_n(\boldsymbol{x}) + p_p(\boldsymbol{x})}. \tag{5}$$

This frozen classifier can be incorporated into the training of a generative model by adding an auxiliary loss, $\mathcal{L}_{PC}$ to the generative model's loss, $\mathcal{L}_{GM}$ to calculate a total loss, $\mathcal{L}_{Tot} = \mathcal{L}_{GM} + \lambda \mathcal{L}_{PC}$, as in (Regenwetter & Ahmed, 2022). Here, λ is some weighting parameter and $\mathcal{L}_{PC}$ is expressed as:

$$\mathcal{L}_{PC} = \mathbb{E}_{p_\theta(\boldsymbol{x})}[\log f_\psi(\boldsymbol{x})]. \tag{6}$$

Pre-trained classifiers can also be applied during inference in certain models, such as in diffusion model guidance (Mazé & Ahmed, 2023; Giannone & Ahmed, 2023). This pre-trained classifier can alternatively be appended to a vanilla model as a rejection sampling layer – a simple but surprisingly effective baseline. We abbreviate pre-trained classifier loss, guidance, and rejection sampling as CL, G, and Rej, respectively, in our testing.

3.3 Discriminator Overloading (DO)

Discriminator overloading is a technique to directly incorporate negative data into GAN model training. This formulation was proposed in two of the first papers to train a generative model using both positive and negative data (though we have made slight modifications for generality): Rumi-GAN (Asokan & Seelamantula, 2020) and Negative Data Augmentation GAN (NDA-GAN) (Sinha et al., 2021). We refer to these formulations as 'discriminator overloading' since the discriminator is 'overloaded' by learning to discriminate between (1) positives and (2) fakes or negatives. As such, the discriminator estimates:

$$f_\phi(\boldsymbol{x}) = \frac{p_p(\boldsymbol{x})}{\lambda p_\theta(\boldsymbol{x}) + (1-\lambda) p_n(\boldsymbol{x}) + p_p(\boldsymbol{x})}, \tag{7}$$

with λ being a weighting parameter. As usual, the generator attempts to generate samples that are classified as real, in this case indicating that they look similar to positive data and dissimilar to negative data. The loss is expressed as:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{p_p(\boldsymbol{x})}[\log f_\phi(\boldsymbol{x})] + \mathbb{E}_{p_\theta(\boldsymbol{x})}[1 - \log(f_\phi(\boldsymbol{x}_\theta))] + \mathbb{E}_{p_n(\boldsymbol{x})}[1 - \log(f_\phi(\boldsymbol{x}))]. \tag{8}$$

As we will show, discriminator overloading is effective. However, instead of conflating the negatives and fakes, we propose two formulations to learn the ratios between the positive, negative, and fake distributions individually. As we demonstrate, this adjustment yields superior performance.

4 Proposed Negative Data Formulation

In this section, we propose several novel formulations for NDGMs. The training pseudocode for each is included in Appendix C, while more details on the mathematical formulation and derivations are included in Appendix B. Instead of learning a density ratio between p_p and an amalgamation of p_n and p_θ, as in discriminator overloading, we instead propose methods to learn pairwise density ratios between the three distributions. The most straightforward method to do so is with a multi-class discriminator.

4.1 Multi-Class Discriminator (MC)

Noting that multi-class classifiers are strong density ratio estimators (Srivastava et al., 2023), we propose an NDGM variant using a multi-class discriminator model that learns three classes: positive, negative, and fake. This discriminator is implicitly estimating all relevant ratios:

$$f_{\phi,c}(\boldsymbol{x}) = \frac{p_c(\boldsymbol{x})}{p_p(\boldsymbol{x}) + p_\theta(\boldsymbol{x}) + p_n(\boldsymbol{x})} \;\forall\; c \in p, n, \theta. \tag{9}$$

However, these ratios are not learned independently. Note that $f_{\phi,p}$ is a reweighted version of Eq. 7. Though this multi-class formulation is similar to discriminator overloading, instead of showing the discriminator a weighted amalgamation of fakes and negatives (as in DO), the multi-class discriminator instead treats fakes and negatives as separate classes, and can potentially refine its knowledge by distinguishing them.

4.2 Double Discriminator (DD)

Though using a single multi-class discriminator to learn pairwise density ratios between p_p, p_n, and p_θ is simpler, we can also accomplish the task using multiple discriminative models. For example, f_ϕ can estimate the ratio p_p/p_θ (this is a standard GAN discriminator), while f_ψ estimates p_n/p_θ. The loss is then expressed as:

$$\begin{aligned}\mathcal{L}(\theta,\phi,\psi) = \mathbb{E}_{p_p(\boldsymbol{x})}[\log f_\phi(\boldsymbol{x})] + \mathbb{E}_{p_\theta(\boldsymbol{x})}[1 - \log(f_\phi(\boldsymbol{x}_\theta))] \\ - \lambda\mathbb{E}_{p_n(\boldsymbol{x})}[\log f_\psi(\boldsymbol{x})] - \lambda\mathbb{E}_{p_\theta(\boldsymbol{x})}[1 - \log(f_\psi(\boldsymbol{x}_\theta))].\end{aligned} \tag{10}$$

Here, $\lambda \in [0, 1]$ is a tuning parameter adjusting the weight of the negative data's contribution to the loss and avoiding instability. Optimal discriminators learn:

$$f_\phi(\boldsymbol{x}) = \frac{p_p(\boldsymbol{x})}{(p_\theta(\boldsymbol{x}) + p_p(\boldsymbol{x}))}, \quad f_\psi(\boldsymbol{x}) = \frac{p_n(\boldsymbol{x})}{(p_\theta(\boldsymbol{x}) + p_n(\boldsymbol{x}))}. \tag{11}$$

The rationale behind the double discriminator algorithm is intuitive when viewed as an expansion of a vanilla GAN. The generator now wants its samples to be classified as positive by the original discriminator and not be classified as negative by the extra discriminator.

This simple double discriminator variant is benchmarked in the appendix (abbreviated DD-a). In practice, however, we find that an alternate formulation that combines this simple two-discriminator concept with discriminator overloading (DO) works better in many cases (we call this DD in the main paper and DD-b in the appendix). As we show in Appendix B, this alternative formulation has a mathematical basis that directly relates to the original GAN training objective.

The alternative formulation consists of the classic discriminator, f_ϕ estimating p_p/p_θ and an overloaded discriminator, f_ψ estimating $(p_p + p_\theta)/p_n$. The total loss function is then expressed as:

$$\begin{aligned}\mathcal{L}(\theta,\phi,\psi) = \; & \mathbb{E}_{p_p(\boldsymbol{x})}[\log f_\phi(\boldsymbol{x})] + \mathbb{E}_{p_\theta(\boldsymbol{x})}[1 - \log(f_\phi(\boldsymbol{x}_\theta))] \\ & + \lambda\mathbb{E}_{p_p(\boldsymbol{x})}[\log f_\psi(\boldsymbol{x})] + \lambda\mathbb{E}_{p_\theta(\boldsymbol{x})}[\log f_\psi(\boldsymbol{x}_\theta)] + \lambda\mathbb{E}_{p_n(\boldsymbol{x})}[1 - \log(f_\psi(\boldsymbol{x}))].\end{aligned} \tag{12}$$

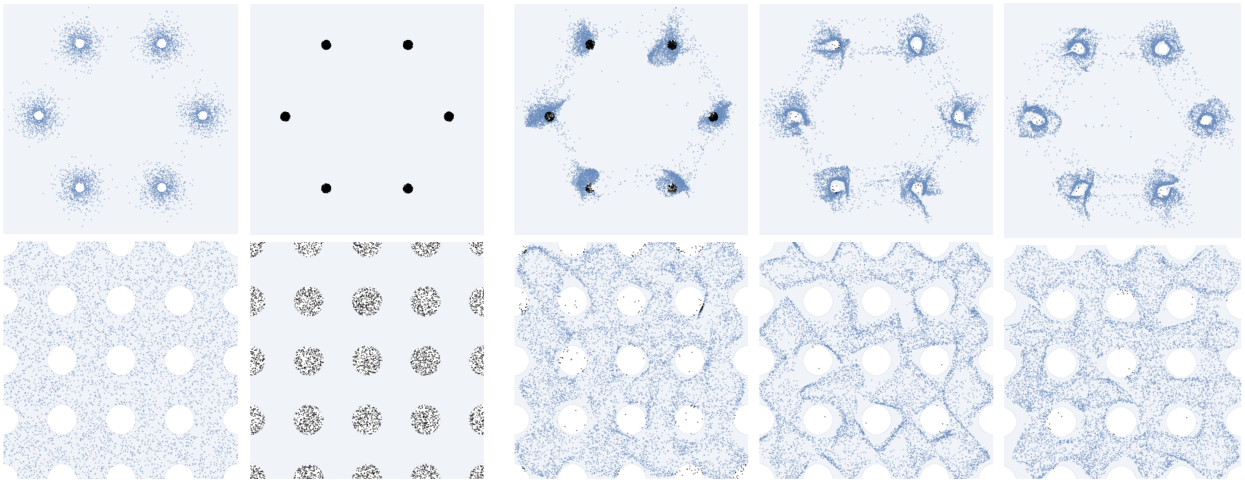
Once again, λ is a weighting parameter modulating the contribution of the negative data. Appendix B contains more detailed derivations and comparisons to similar formulatiuons. For more details on the training algorithms, see Appendix C.

5 Experiments

We now present experiments on (i) 2D densities, where we benchmark 16 different model including ours, the SOTA, baseline NDGMs, and vanilla models; (ii) 9 diverse engineering tasks with different levels of complexity, and (iii) a block stack problem where we investigate multiple detailed constraints. For more experiments on 2D densities, engineering tasks, and block stacks, see Appendices D, G, and F. For additional experiments on diversity-augmented generation, see Appendix E.

5.1 Negative Data for Densities with Constraints

We first showcase our approach using two easy-to-visualize but highly non-convex 2D test problems. `Problem 1` is an adaptation of a classic multi-modal test problem made significantly more challenging with the addition of small negative 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 positive and negative data points are randomly sampled, as shown in Figures 5a and 5b. Architecture and training details are included in Appendix D.



(a) Positive Data **(b)** Negative Data **(c)** GAN **(d)** GAN-MC (Ours) **(e)** GAN-DD (Ours)

Figure 2: Generated Distributions from several generative models on two highly non-convex test problems (`Problem 1` on top, `Problem 2` on the bottom). Positive data points and samples are shown in blue and negative ones in black. Our proposed NDGM models (GAN-MC, GAN-DD) generate significantly fewer negative samples.

Models. We test 16 variants of GAN, VAE, and DDPM models. Among these are: Vanilla models (GAN, VAE, DDPM), trained only on positive data; Class conditional models (GAN-CC, VAE-CC) trained on both datasets in a binary class conditional setting as in Sec 3.1; Models augmented with a frozen pre-trained classifier to steer models during training though a classification loss (GAN-CL, VAE-CL, DDPM-CL); DDPM with pre-trained classifier guidance only during inference (DDPM-G); Vanilla models trained on only positive data, but augmented with a rejection sampling layer using a frozen classifier pre-trained on both positive and negative data (GAN-Rej, VAE-Rej, DDPM-Rej); Auxiliary Classifier GAN (GAN-AC); GAN with discriminator overloading as in NDA-GANs (Sinha et al., 2021) and Rumi-GANs (Asokan & Seelamantula, 2020) (GAN-DO). Our multi-class-discriminator GAN (Sec. 4.1) (GAN-MC); Our GAN variant with two discriminators (Sec. 4.2) (GAN-DD); GAN-DD-a is included in Appendix D.3.

Table 1: Mean scores across two highly non-convex test problems. The **best** result is in bold. The <u>next two best</u> are underlined. Both of the formulations we propose (GAN-MC, GAN-DD) outperform the previous state-of-the-art (GAN-DO) in all metrics. Each model is tested three times for each of the two toy densities.

Model	Invalidity (%) ↓	MMD (10^{-3}) ↓	F_1 ↑
VAE	12.2	3.19	0.89
VAE-CC	14.6	3.13	0.88
VAE-CL	<u>0.29</u>	5.06	0.86
VAE-Rej	1.30	4.25	0.90
DDPM	7.74	3.83	0.91
DDPM-CL	9.41	3.60	0.90
DDPM-G	5.20	4.19	0.89
DDPM-Rej	1.11	3.65	0.90
GAN	6.22	4.96	0.84
GAN-CC	6.20	4.75	0.90
GAN-AC	2.87	3.16	<u>0.94</u>
GAN-CL	0.31	<u>2.30</u>	**0.95**
GAN-Rej	0.43	4.88	0.86
GAN-DO (SOTA)	0.46	4.33	0.93
GAN-MC (Ours)	<u>0.30</u>	<u>2.26</u>	<u>0.94</u>
GAN-DD (Ours)	**0.28**	**2.03**	**0.95**

Metrics. We score each model on three metrics. 1) Invalidity – the fraction of generated samples that violate the constraints (negative samples). 2) Maximum Mean Discrepancy (MMD), a common distributional similarity metric. 3) F_1 score for generative models, as proposed in (Sajjadi et al., 2018). We present an expanded study with more metrics in Appendix D.3.

Results. Figure 5 plots the datasets and the generated distributions of several select models (vanilla GAN and the two NDGMs we propose). Our GAN-MC, and GAN-DD variants both achieve near-perfect constraint satisfaction compared to the vanilla model. Plots for all 16 models are included in Appendix D.3.

Table 1 presents scores across all models (expanded tables included in Appendix D.3). Both our GAN-MC and GAN-DD variants score within the top three models for each of the metrics. Furthermore, they both outperform GAN-DO, the previous state-of-the-art formulation in every metric. Note that GAN-DO underperforms certain baselines, such as training with a frozen classifier loss (GAN-CL). In general, most NDGM baselines like classifier loss (-CL) and rejection sampling (-Rej) significantly outperform vanilla models. In all, our proposed GAN-MC and GAN-DD models achieve the highest performance across all metrics.

5.2 How Much Negative Data is Enough?

Table 2: Comparison of **invalidity metric** for NDGM models trained with different numbers of positive datapoints (N_p) and negative datapoints (N_n). NDGMs can generate many times fewer constraint-violating samples, even when trained on orders of magnitude less data. A GAN-DD is benchmarked when $N_n > 0$, otherwise a vanilla GAN is benchmarked. Scores are averaged over four instantiations. **Lower is better.**

(a) Models **(b)** Problem 1 **(c)** Problem 2

	Negative Samples	Positive Samples $1K$	Positive Samples $4K$	Positive Samples $16K$	Positive Samples $1K$	Positive Samples $4K$	Positive Samples $16K$
GAN	0	10.3%	10.0%	12.3%	2.4%	2.3%	5.9%
GAN-DD	$1K$	0.6%	0.3%	**0.3**%	0.8%	0.6%	0.6%
GAN-DD	$4K$	**0.2** %	0.3%	0.4%	**0.2**%	0.3%	0.5%
GAN-DD	$16K$	**0.2** %	**0.1**%	**0.3**%	**0.2**%	**0.2**%	**0.1**%

In the realm of generative models, it is theoretically possible to exactly recover the underlying data distribution, p_x, when provided with an infinite amount of 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 data is not a viable strategy to improve constraint satisfaction. Fortunately, we find that NDGMs can be significantly more data-efficient than vanilla generative models. In Table 2, we present empirical evidence to support our arguments. By solely increasing the amount of positive data without incorporating negative data (first row - vanilla GAN), we observed no reduction in invalidity metric, despite a 16x increase in positive data. Conversely, when we introduce a modest proportion of negative data (6% – $N_n = 1K$, $N_p = 16K$), we can achieve a 10-40x reduction in the rate of invalid sample generation. Furthermore, we can even remove much of the positive data with only minor performance drops. Notably, even with 1k positive and 1k negative data points, NDGMs generate 7-20x fewer invalid samples compared to models trained on 16K positive data points. These experiments demonstrate that NDGMs can significantly (7-20x) outperform vanilla models using a fraction (13%) of the data. Importantly, practitioners seeking to improve their generative models may achieve much more value by collecting negative data, rather than additional positive data.

5.3 Handling Connectivity and Stability Constraints

Block-stacking problems have long been studied as a case study in 'intuitive physics' (Battaglia et al., 2013; Riochet et al., 2020), on which many predictive and generative computational approaches have been tested (Hamrick et al., 2018; Smith et al., 2019b). In this study, we address an intuitive block-stacking problem featuring connectivity and stability constraints. These constraints are reflective of common engineering constraints related to interfacing of mechanical components and assembly-level functional requirements. Therefore, the block stacking problem is a representative, yet intuitive case study for

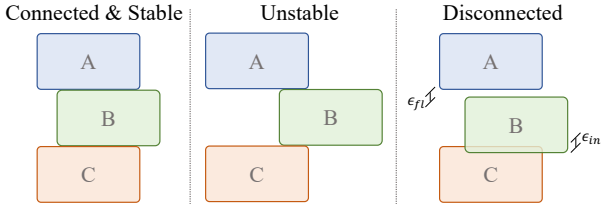


Figure 3: Overview of constraints in stacked blocks problem. Our goal is to generate valid stacks of blocks (left) that are (I) connected and (II) stable. Stacks that violate either the connectivity or stability constraint are considered invalid.

engineering applications. The constraints are defined as follows: (i) `connectivity`: Blocks must stack without floating or intersecting up to a prescribed tolerance, and (ii) `stability`: Any block (or sub-stack) must overlap with its support in such a way that its 'center of mass' falls in the supporting blocks' area. Therefore, the positive data consists of stable & connected stacks, while the negative data consists of unstable & connected, stable & disconnected, and unstable & disconnected stacks.

Fulfilling Constraints. We train a vanilla model using only the positive data, and two NDGMs: a GAN-DO (as in (Sinha et al., 2021) and (Asokan & Seelamantula, 2020)) and our proposed GAN-DD, which leverages Eq. 12. We test on 20 splits (1000 samples each) and report scores in Table 3, where we break down constraint satisfaction into individual scores. GAN-DD outperforms the base model and the GAN-DO by a large margin in most constraint-satisfaction scores, with fewer intersecting and floating blocks, and in particular on global connectivity between boxes, indicating that the GAN-DD approach is effective in improving constraint satisfaction in situations where precision is important. Factoring in stability, we see an even larger gap between the baselines and our model, emphasizing the challenges in fulfilling multiple sets of fine-grained constraints. These experiments are reported in Appendix F, alongside more visualizations.

Table 3: Constraint satisfaction on block stacking problem. Blocks are floating or intersecting if the distance between their touching edges is larger than 0.9 mm (the minimum distance between constraints in the negative data is 1 mm). b: base block; m: middle block; t: top block. For floating $\boldsymbol{y}_b < \boldsymbol{y}_m$ and for intersecting $\boldsymbol{y}_b > \boldsymbol{y}_m$.

Metrics	GAN	GAN-DO	**GAN-DD**
↓ Median($\lvert \boldsymbol{y}_b^1 - \boldsymbol{y}_m^0 \rvert$)	2.78 *mm*	5.12 *mm*	**0.54** *mm*
↓ Median($\lvert \boldsymbol{y}_m^1 - \boldsymbol{y}_t^0 \rvert$)	2.12 *mm*	1.91 *mm*	**0.83** *mm*
↓ Floating($\boldsymbol{y}_b, \boldsymbol{y}_m$)	20.44 %	14.47 %	**13.78** %
↓ Floating($\boldsymbol{y}_m, \boldsymbol{y}_t$)	38.04 %	20.73 %	**13.94** %
↓ Intersect($\boldsymbol{y}_b, \boldsymbol{y}_m$)	64.59 %	77.20 %	**0.00** %
↓ Intersect($\boldsymbol{y}_m, \boldsymbol{y}_t$)	43.80 %	54.82 %	**30.79** %
↑ Connected($\boldsymbol{y}_b, \boldsymbol{y}_m$)	14.96 %	8.32 %	**86.21** %
↑ Connected($\boldsymbol{y}_m, \boldsymbol{y}_t$)	18.15 %	24.44 %	**55.26** %
↑ Connected & Stable	3.28 %	8.85 %	**36.02** %

We also plot the distance distribution between blocks with and without negative data (Fig. 4). In the absence of negative data, the relative placement of blocks is much less precise, resulting in significant overlap (negative distance values) or gaps (positive distance values). When leveraging negative data, even when the constraints are not fulfilled, the errors have a much smaller magnitude, providing additional qualitative evidence of the effectiveness and importance of using negative data for fine-grained constraint satisfaction.

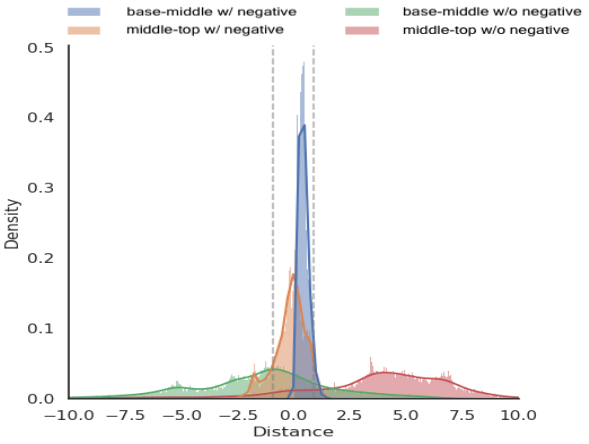


Figure 4: Block placement by NDGM (w/ negative) vs vanilla model (w/o negative). The two vertical grey lines indicate the acceptable tolerance such that the constraints are considered satisfied. Our GAN-DD greatly reduces the overlap or air gap between blocks compared to a GAN, demonstrating its aptitude for constraint satisfaction.

5.4 Negative Data in Engineering Tasks

Generative models are commonly use to tackle engineering problems with constraints (Oh et al., 2019; Nie et al., 2021), but are often criticized for their inability to satisfy them (Woldseth et al., 2022; Regenwetter et al., 2023). To assess how our NDGMs fare in real engineering problems, we next benchmark the same 16 methods as in Sec. 5.1 on a dozen diverse engineering tasks, which are discussed in detail in Appendix G. These problems span numerous engineering disciplines including assorted industrial design tasks (compression spring, gearbox, heat exchanger, pressure vessel), structural and material design tasks (Ashby chart, cantilever beam, reinforced concrete, truss, welded beam), and several complex high-level design problems: Ship hulls with hydrodynamic constraints; bike frames with loading requirements; automobile chassis with performance requirements in impact testing. A variety of constraints are applied, including engineering standards from authoritative bodies like the American Concrete Institute (ACI), the American Society of Mechanical Engineers (ASME), and the European Enhanced Vehicle-Safety Committee (EEVC).

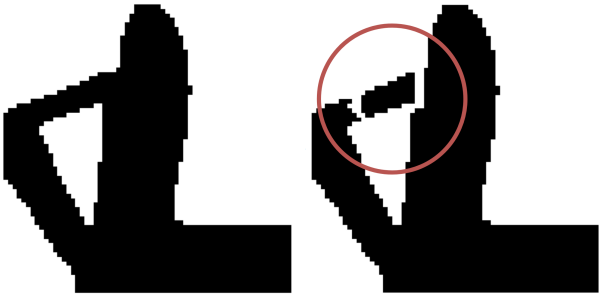
We include the scores across the same 16 models tested in Sec. 5.1, all 12 engineering problems, and a larger set of metrics in Appendix G. Shown in Fig. 4 are invalidity scores over a subset of models and problems (problems where a vanilla model is already $> 99\%$ successful in generating positive samples are considered to be 'solved' and are only shown in Appendix G). The median score over three training runs is shown. In every problem, either the discriminator overloading GAN (GAN-DO) or our multiclass discriminator model (GAN-MC) are the top performers, indicating that negative data GANs significantly outperform the vanilla GAN. However, on the problems where the vanilla model struggles, our GAN-MC model outperforms GAN-DO. Specifically, GAN-DO is only the top performer on problems where the vanilla model is already at least 97% successful in generating positive samples. Poor validity scores are expected on the ship hull dataset because most constraints are not represented in the negative data (thus, for the majority of constraints, NDGMs have no natural advantage over vanilla models). However GAN-MC still manages to generate valid designs at 2.8x the rate of the GAN and over 1.6x the rate of GAN-DO. Notably, simple baselines (rejection sampling, classifier loss) also perform much better than vanilla models, as demonstrated in Appendix G. Additionally, we find that likelihood-based models outperform GANs in constraint satisfaction but lag behind in distributional similarity in many problems. All in all, NDGMs display a widespread dominance over their vanilla counterparts across a variety of engineering tasks, with our proposed GAN-MC excelling in more challenging ones.

Table 4: Constraint Satisfaction on 9 Engineering Benchmarks. Percentage (%) of generated samples violating constraints is shown. **Best** is bolded. Smaller (↓) is better. Scores are averaged over three instantiations. Problems are sorted by the invalidity of the baseline GAN model. Our GAN-MC outperforms GAN-DO on problems that the vanilla GAN struggles with (i.e. hard engineering problems where NDGMs are especially needed).

Dataset	GAN	GAN-DO	**GAN-MC**
Compression Spring	2.01	**0.31**	0.55
Ashby Chart	2.35	**2.24**	3.22
Pressure Vessel	2.64	**0.05**	0.38
Welded Beam	2.86	**0.64**	1.25
Bike Frame	6.02	7.32	**5.89**
Heat Exchanger	7.75	6.41	**4.64**
Cantilever Beam	8.22	5.27	**4.67**
Car Impact	10.43	6.00	**5.33**
Ship Hull	98.0	96.4	**94.3**

5.5 Negative Data in High-Dimensional Problems

Having tested a variety of tabular engineering problems, we next consider whether our proposed methods can translate to higher-dimensional image domains such as images. We examine a common engineering design problem known as topology optimization (TO), which seeks to optimally distribute material in a spatial domain to achieve a certain objective (often minimizing mechanical compliance) (Sigmund & Maute, 2013). Simply put, TO is often used to create structures with high rigidity and low weight. The use of generative models for TO is very popular (Shin et al., 2023), but existing methods have been criticized for significant shortcomings (Woldseth et al., 2022) related to constraint satisfaction, such as generated topologies not being fully connected. Disconnected topologies tend to be highly sub-optimal and are impractical to fabricate.



(a) Positive **(b)** Negative

Figure 5: Examples of Positive and Negative Topologies.

Table 5: Constraint Satisfaction on Topology Optimization Problem. **Best** is bolded. Our GAN-MC outperforms GAN, both generating fewer invalid designs and generating designs with less severe constraint violations. However, the quality of the negative data has a significant impact on GAN-MC's performance. When trained on "harder" negative data (rejected samples from a vanilla model), it performs better than when trained on "easier" procedurally-generated negative data.

	Invalidity (%) ↓	Invalidity (Pixels) ↓
GAN	36.3	1.28
GAN-MC (Synthetic negative data)	24.4	0.38
GAN-MC (Rejected negative data)	**16.0**	**0.29**

To address this, we train NDGMs using disconnected topologies as negative data, using the classification guidance dataset from (Mazé & Ahmed, 2023). This dataset features procedurally-generated synthetic negatives with artificially-added floating components. We also replace the negatives in the dataset with disconnected topologies generated by a vanilla GAN trained on the positive data. These rejected negatives are "harder" negatives (closer to the positive distribution) and are hence more informative than the synthetic negatives. In evaluating models, we measure the proportion of generated topologies with disconnected components, as well as the average number of disconnected pixels in each generated topology. A vanilla GAN generates many more invalid topologies than either GAN-MC variant. As expected, the stronger negatives generated through rejection sampling result in superior performance. Compared to the stronger GAN-MC, the vanilla GAN generates 2.3x as many invalid topologies and violates constraints by 4.4x the severity. Additional details are included in Appendix H.

6 Discussion & Conclusion

Deconflating fakes and negatives. We presented two new NDGM formulations which separately estimate several density ratios to better learn a true data distribution, rather than conflating fakes and negatives as done in the current SOTA. Our models dominate in highly non-convex 2D problems and outperform baselines in the 5 most challenging engineering problems tested. Our methods also excel in a block stacking problem, generating 11x and 4x more valid stacks than vanilla models and the previous SOTA, respectively. Finally, our methods outperform baselines by 4x in an image-based topology optimization problem.

GANs versus diffusion models using negative data. Despite the growing popularity of diffusion models, GANs remain state of the art in many engineering design problems. Having benchmarked DDPMs in several of the problems tested, we find this statement to hold true in the context of NDGMs. Although negative-data-augmented DDPMs surpassed our GAN models in some metrics, this typically came at the expense of others. Conversely, GANs outperformed across all metrics in several problems (Sec. 5.1, for example). We look forward to future research which advances the capabilities of negative data diffusion models and makes them more viable in engineering design.

NDGMs are underutilized. We believe NDGMs are underutilized in engineering design. This assertion is substantiated by several observations: 1) The widespread use of vanilla models in engineering design (Regenwetter et al., 2022a). 2) The relatively low cost of collecting negative data versus positive data in many engineering contexts. 3) The overwhelming dominance of NDGMs over vanilla models in our engineering benchmarks. 4) The data-efficiency improvements we demonstrated using negative data. Though our methods achieved SOTA performance in many problems, even the simple baseline NDGMs that we tested significantly outperform their vanilla counterparts. Therefore, we generally advocate for the increased use of NDGMs in engineering design.

Generating high-quality negative data. Selecting strategies to generate negative data is an important research question. In the final case study on topology optimization, rejection sampling resulted in "stronger" negative data than the procedural generation method. It also required access to an oracle (constraint evaluator), which may be unavailable or prohibitively expensive in some applications. However, there are not always cheap, viable procedural generation approaches for negative data either. Effective negative data generation remains largely problem-dependent and the relative quality of negative data generation approaches is not necessarily straightforward. We anticipate that domain-agnostic methods to generate high-quality negative data could pair well with NDGMs and expand their impact.

Limitations. As we demonstrate, NDGMs are sensitive to the quality of their negative training data. Although negative data is often cheaper than positive data in engineering design problems, generating high-quality negative data may be challenging in some domains. In other domains, sourcing any kind of negative data may be impossible. In domains where high-quality negative data is unavailable, NDGMs will naturally be impractical.

Conclusion. In this paper, we presented two new Negative-Data Generative Models (NDGMs). We demonstrated that these models outperform more than a dozen other formulations in extensive benchmarks across several test problems and a dozen real engineering problems. We displayed that simple baseline NDGMs also achieve strong performance compared to vanilla models, demonstrating the general potency of NDGMs. Notably, we showed that NDGMs can often be much more data-efficient than classic models.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, ICML'17, pp. 214–223. PMLR, JMLR.org, 2017.

Siddarth Asokan and Chandra Seelamantula. Teaching a gan what not to learn. *Advances in Neural Information Processing Systems*, 33:3964–3975, 2020.

Noah J Bagazinski and Faez Ahmed. Ship-d: Ship hull dataset for design optimization using machine learning. *arXiv preprint arXiv:2305.08279*, 2023.

Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.

Mohammad Mahdi Behzadi and Horea 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. ISSN 1050-0472. doi: 10.1115/1.4052757. URL https://doi.org/10.1115/1.4052757. 021711.

Martin Philip Bendsøe and Noboru Kikuchi. Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, 71(2):197–224, 11 1988. ISSN 00457825. doi: 10.1016/0045-7825(88)90086-2. URL https://linkinghub.elsevier.com/retrieve/pii/0045782588900862.

Camille Bilodeau, Wengong Jin, Tommi Jaakkola, Regina Barzilay, and Klavs F Jensen. Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 12(5):e1608, 2022.

Ramin Bostanabad, Yichi Zhang, Xiaolin Li, Tucker Kearney, L Catherine Brinson, Daniel W Apley, Wing Kam Liu, and Wei Chen. Computational microstructure characterization and reconstruction: Review of the state-of-the-art techniques. *Progress in Materials Science*, 95:1–41, 2018.

Andrew Brock, Theodore Lim, James Millar Ritchie, and Nick Weston. Context-aware content generation for virtual environments. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 50084, pp. V01BT02A045. American Society of Mechanical Engineers, 2016.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Ruijin Cang, Hechao Li, Hope Yao, Yang Jiao, and Yi 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.

George Casella and Roger L. Berger. *Statistical Inference*. Duxbury, Pacific Grove, CA, 2002.

Michael Chang, Alyssa L Dayan, Franziska Meier, Thomas L Griffiths, Sergey Levine, and Amy Zhang. Neural constraint satisfaction: Hierarchical abstraction for combinatorial generalization in object rearrangement. *arXiv preprint arXiv:2303.11373*, 2023.

Hongrui Chen and Xingchen 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.

Qiuyi Chen, Jun Wang, Phillip Pope, Wei Chen, and Mark Fuge. Inverse design of two-dimensional airfoils using conditional generative models and surrogate log-likelihoods. *Journal of Mechanical Design*, 144(2): 021712, 2022.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Wei Chen and Faez Ahmed. Mo-padgan: Reparameterizing engineering designs for augmented multi-objective optimization. *Applied Soft Computing*, 113:107909, 2021a.

Wei Chen and Faez Ahmed. Padgan: Learning to generate high-quality novel designs. *Journal of Mechanical Design*, 143(3):031703, 2021b.

Wei Chen and Mark Fuge. Béziergan: Automatic generation of smooth curves from interpretable low-dimensional parameters. *arXiv preprint arXiv:1808.08871*, 2018.

Wei Chen and Mark Fuge. Synthesizing designs with interpart dependencies using hierarchical generative adversarial networks. *Journal of Mechanical Design*, 141(11):111403, 2019.

Wei Chen, Kevin Chiu, and Mark Fuge. Aerodynamic design optimization and shape exploration using generative adversarial networks. In *AIAA Scitech 2019 Forum*, pp. 2351, 2019.

Yu Cheng, Yongshun Gong, Yuansheng Liu, Bosheng Song, and Quan Zou. Molecular design in drug discovery: a comprehensive review of deep generative models. *Briefings in bioinformatics*, 22(6):bbab344, 2021.

Kristy Choi, Chenlin Meng, Yang Song, and Stefano Ermon. Density ratio estimation via infinitesimal classification. In *International Conference on Artificial Intelligence and Statistics*, pp. 2552–2573. PMLR, 2022.

Matthew Dering, James Cunningham, Raj Desai, Michael A Yukish, Timothy W Simpson, and Conrad 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, pp. V02AT03A015. American Society of Mechanical Engineers, 2018.

Shrinath Deshpande and Anurag Purwar. Computational creativity via assisted variational synthesis of mechanisms using deep generative models. *Journal of Mechanical Design*, 141(12), 2019.

Shrinath Deshpande and Anurag Purwar. An Image-Based Approach to Variational Path Synthesis of Linkages. *Journal of Computing and Information Science in Engineering*, 21(2), 10 2020. ISSN 1530-9827. doi: 10.1115/1.4048422. URL https://doi.org/10.1115/1.4048422. 021005.

Nikolaos Dionelis, Sotirios A Tsaftaris, and Mehrdad Yaghoobi. Omasgan: Out-of-distribution minimum anomaly score gan for anomaly detection. In *2022 Sensor Signal Processing for Defence Conference (SSPD)*, pp. 1–5. IEEE, 2022.

Mohamed Elfeki, Camille Couprie, Morgane Riviere, and Mohamed Elhoseiny. Gdpp: Learning diverse generations using determinantal point processes. In *International conference on machine learning*, pp. 1774–1783. PMLR, 2019.

Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

Amir Hossein Gandomi and Xin-She Yang. Benchmark problems in structural optimization. In *Computational optimization, methods and algorithms*, pp. 259–281. Springer, 2011.

Amir Hossein Gandomi, Xin-She Yang, and Amir Hossein Alavi. Mixed variable structural optimization using firefly algorithm. *Computers & Structures*, 89(23-24):2325–2336, 2011.

Giorgio Giannone and Faez Ahmed. Diffusing the optimal topology: A generative optimization approach. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 87301, pp. V03AT03A012. American Society of Mechanical Engineers, 2023.

Giorgio Giannone, Akash Srivastava, Ole Winther, and Faez Ahmed. Aligning optimization trajectories with diffusion models for constrained design generation. *Advances in Neural Information Processing Systems*, 36, 2024.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, NIPS'14, pp. 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.

Jessica B Hamrick, Kelsey R Allen, Victor Bapst, Tina Zhu, Kevin R McKee, Joshua B Tenenbaum, and Peter W Battaglia. Relational inductive bias for physical construction in humans and machines. *arXiv preprint arXiv:1806.01203*, 2018.

Amin Heyrani Nobari, Wei (Wayne) Chen, and Faez Ahmed. RANGE-GAN: Design Synthesis Under Constraints Using Conditional Generative Adversarial Networks. *Journal of Mechanical Design*, pp. 1–16, 09 2021. ISSN 1050-0472. doi: 10.1115/1.4052442. URL https://doi.org/10.1115/1.4052442.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Fergus Imrie, Anthony R Bradley, Mihaela van der Schaar, and Charlotte M Deane. Deep generative models for 3d linker design. *Journal of chemical information and modeling*, 60(4):1983–1995, 2020.

Cole Jetton, Matthew Campbell, and Christopher Hoyle. Constraining the Feasible Design Space in Bayesian Optimization With User Feedback. *Journal of Mechanical Design*, 146(4):041703, 11 2023. ISSN 1050-0472. doi: 10.1115/1.4063906. URL https://doi.org/10.1115/1.4063906.

Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Jin-Woong Lee, Nam Hoon Goo, Woon Bae Park, Myungho Pyo, and Kee-Sun Sohn. Virtual microstructure design for steels using generative adversarial networks. *Engineering Reports*, 3(1):e12274, 2021.

Baotong Li, Congjia Huang, Xin Li, Shuai Zheng, and Jun Hong. Non-iterative structural topology optimization using deep learning. *Computer-Aided Design*, 115:172–180, 2019. ISSN 0010-4485. doi: https://doi.org/10.1016/j.cad.2019.05.038. URL https://www.sciencedirect.com/science/article/pii/S001044851930185X.

Runze Li, Yufei Zhang, and Haixin Chen. Learning the aerodynamic design of supercritical airfoils through deep reinforcement learning. *AIAA Journal*, pp. 1–14, 2021.

Xiang Li, Shaowu Ning, Zhanli Liu, Ziming Yan, Chengcheng Luo, and Zhuo 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.

Siyan Liu, Zhi Zhong, Ali Takbiri-Borujeni, Mohammad Kazemi, Qinwen Fu, and Yuhao Yang. A case study on homogeneous and heterogeneous reservoir porous media reconstruction by using generative adversarial networks. *Energy Procedia*, 158:6164–6169, 2019.

Zhaocheng Liu, Lakshmi Raju, Dayu Zhu, and Wenshan 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.

Manoj Malviya. A systematic study of deep generative models for rapid topology optimization. 2020.

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. URL https://arxiv.org/abs/2208.09591.

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

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

Lukas Mosser, Olivier Dubrule, and Martin J Blunt. Reconstruction of three-dimensional porous media using generative adversarial neural networks. *Physical Review E*, 96(4):043309, 2017.

Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pp. 607–617, 2020.

Zhenguo Nie, Tong Lin, Haoliang Jiang, and Levent Burak Kara. Topologygan: Topology optimization using generative adversarial networks based on physical fields over the initial domain. *Journal of Mechanical Design*, 143(3):031715, 2021.

Amin Heyrani Nobari, Wei Chen, and Faez Ahmed. Pcdgan: A continuous conditional diverse generative adversarial network for inverse design. *arXiv preprint arXiv:2106.03620*, 2021.

Amin Heyrani Nobari, Wei Chen, and Faez Ahmed. Range-constrained generative adversarial network: Design synthesis under constraints using conditional generative adversarial networks. *Journal of Mechanical Design*, 144(2), 2022.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pp. 2642–2651. PMLR, 2017.

Sangeun Oh, Yongsu Jung, Ikjin Lee, and Namwoo 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, pp. V02AT03A008. American Society of Mechanical Engineers, 2018.

Sangeun Oh, Yongsu Jung, Seongsin Kim, Ikjin Lee, and Namwoo Kang. Deep generative design: Integration of topology optimization and generative models. *Journal of Mechanical Design*, 141(11), 2019.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. Sketchgen: Generating constrained cad sketches. *Advances in Neural Information Processing Systems*, 34: 5077–5088, 2021.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pp. 5171–5180. PMLR, 2019.

Sharad Rawat and MH Herman Shen. Application of adversarial networks for 3d structural topology optimization. Technical report, SAE Technical Paper, 2019.

Lyle Regenwetter and Faez 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, pp. V03BT03A046. American Society of Mechanical Engineers, 2022.

Lyle Regenwetter, Brent Curry, and Faez 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.

Lyle Regenwetter, Amin Heyrani Nobari, and Faez Ahmed. Deep generative models in engineering design: A review. *Journal of Mechanical Design*, 144(7):071704, 2022a.

Lyle Regenwetter, Colin Weaver, and Faez Ahmed. Framed: Data-driven structural performance analysis of community-designed bicycle frames, 2022b.

Lyle Regenwetter, Akash Srivastava, Dan Gutfreund, and Faez Ahmed. Beyond statistical similarity: Rethinking metrics for deep generative models in engineering design. *arXiv preprint arXiv:2302.02913*, 2023.

Benjamin Rhodes, Kai Xu, and Michael U Gutmann. Telescoping density-ratio estimation. *Advances in neural information processing systems*, 33:4905–4916, 2020.

Ronan Riochet, Mario Ynocente Castro, Mathieu Bernard, Adam Lerer, Rob Fergus, Véronique Izard, and Emmanuel Dupoux. Intphys: A framework and benchmark for visual intuitive physics reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

Mikael Sabuhi, Ming Zhou, Cor-Paul Bezemer, and Petr Musilek. Applications of generative adversarial networks in anomaly detection: a systematic literature review. *Ieee Access*, 9:161003–161029, 2021.

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

Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P Adams. Vitruvion: A generative model of parametric cad sketches. *arXiv preprint arXiv:2109.14124*, 2021.

Shashank Sharma and Anurag 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, pp. V010T10A034. American Society of Mechanical Engineers, 2020.

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

Seungyeon Shin, Dongju Shin, and Namwoo Kang. Topology optimization via machine learning and deep learning: A review. *Journal of Computational Design and Engineering*, 10(4):1736–1766, 2023.

Dule Shu, James Cunningham, Gary Stump, Simon W Miller, Michael A Yukish, Timothy W Simpson, and Conrad S Tucker. 3d design using generative adversarial networks and physics-based validation. *Journal of Mechanical Design*, 142(7):071701, 2020.

Ole Sigmund and Kurt Maute. Topology optimization approaches: A comparative review. *Structural and Multidisciplinary Optimization*, 48(6):1031–1055, 2013. ISSN 1615-147X. doi: 10.1007/s00158-013-0978-6.

Abhishek Sinha, Kumar Ayush, Jiaming Song, Burak Uzkent, Hongxia Jin, and Stefano Ermon. Negative data augmentation. *arXiv preprint arXiv:2102.05113*, 2021.

Kevin Smith, Lingjie Mei, Shunyu Yao, Jiajun Wu, Elizabeth Spelke, Josh Tenenbaum, and Tomer Ullman. Modeling expectation violation in intuitive physics with coarse probabilistic object representations. *Advances in neural information processing systems*, 32, 2019a.

Kevin A. Smith, Lingjie Mei, Shunyu Yao, Jiajun Wu, Elizabeth S. Spelke, Joshua B. Tenenbaum, and Tomer David Ullman. Modeling expectation violation in intuitive physics with coarse probabilistic object representations. In *Neural Information Processing Systems*, 2019b.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pp. 3738–3746, 2016.

Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles 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 (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 3308–3318. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/44a2e0804995faf8d2e3b084a1e2db1d-Paper.pdf.

Akash Srivastava, Seungwook Han, Kai Xu, Benjamin Rhodes, and Michael U. Gutmann. Estimating the density ratio between distributions with high discrepancy using multinomial logistic regression. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=jM8nzUzBWr.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012a.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012b.

Ren Kai Tan, Nevin L Zhang, and Wenjing Ye. A deep learning–based method for the design of microstructural materials. *Structural and Multidisciplinary Optimization*, 61(4):1417–1438, 2020.

Yingheng Tang, Keisuke Kojima, Toshiaki Koike-Akino, Ye Wang, Pengxiang Wu, Mohammad Tahersima, Devesh Jha, Kieran Parsons, and Minghao Qi. Generative deep learning model for a multi-level nano-optic broadband power splitter. In *2020 Optical Fiber Communications Conference and Exhibition (OFC)*, pp. 1–3. IEEE, 2020.

Sofia Valdez, Carolyn Seepersad, and Sandilya 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.

Jun Wang, Wei Wayne Chen, Daicong Da, Mark Fuge, and Rahul 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.

Liwei Wang, Yu-Chin Chan, Faez Ahmed, Zhao Liu, Ping Zhu, and Wei Chen. Deep generative modeling for mechanistic-based learning and design of metamaterial systems. *Computer Methods in Applied Mechanics and Engineering*, 372:113377, 2020.

Eamon Whalen, Azariah Beyene, and Caitlin Mueller. Simjeb: simulated jet engine bracket dataset. In *Computer Graphics Forum*, volume 40, pp. 9–17. Wiley Online Library, 2021.

Rebekka V Woldseth, Niels Aage, J Andreas Bærentzen, and Ole Sigmund. On the use of artificial neural networks in topology optimisation. *Structural and Multidisciplinary Optimization*, 65(10):294, 2022.

Patricia Wollstadt, Mariusz Bujny, Satchit Ramnath, Jami J Shah, Duane Detwiler, and Stefan 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.

Tianju Xue, Thomas J Wallin, Yigit Menguc, Sigrid Adriaenssens, and Maurizio Chiaramonte. Machine learning generative models for automatic design of multi-material 3d printed composite solids. *Extreme Mechanics Letters*, 41:100992, 2020.

Xin-She Yang and Amir Hossein Gandomi. Bat algorithm: a novel approach for global engineering optimization. *Engineering computations*, 29(5):464–483, 2012.

Zijiang Yang, Xiaolin Li, L Catherine Brinson, Alok N Choudhary, Wei Chen, and Ankit Agrawal. Microstructural materials design via deep adversarial learning methodology. *Journal of Mechanical Design*, 140(11), 2018.

Emre Yilmaz and Brian German. Conditional generative adversarial network framework for airfoil inverse design. In *AIAA aviation 2020 forum*, pp. 3185, 2020.

Yonggyun Yu, Taeil Hur, Jaeho Jung, and In Gwun Jang. Deep learning for determining a near-optimal topological design without any iteration. *Structural and Multidisciplinary Optimization*, 59(3):787–799, 2019.

Muhammad Zaigham Zaheer, Jin-ha Lee, Marcella Astrid, and Seung-Ik Lee. Old is gold: Redefining the adversarially learned one-class classifier training paradigm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14183–14193, 2020.

Hui Zhang, Lei Yang, Changjian Li, Bojian Wu, and Wenping Wang. Scaffoldgan: Synthesis of scaffold materials based on generative adversarial networks. *Computer-Aided Design*, 138:103041, 2021. ISSN 0010-4485. doi: https://doi.org/10.1016/j.cad.2021.103041. URL https://www.sciencedirect.com/science/article/pii/S001044852100052X.

Wentai Zhang, Zhangsihao Yang, Haoliang Jiang, Suyash Nigam, Soji Yamakawa, Tomotake Furuhata, Kenji Shimada, and Levent Burak 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, pp. V02AT03A017. American Society of Mechanical Engineers, 2019.

A Related Work

Constraints in Engineering Problems. Generally, we can categorize the constraint information of engineering problems into four types (Regenwetter et al., 2023).

(i) *No Constraint Information*: No information about constraints is given or can be collected, and learning constraints is typically infeasible or extremely challenging in a finite data regime.

(ii) *'Negative' Dataset of Invalid Designs*: A collection of constraint-violating negative designs is available. Our method leverages such negative data to learn a constraint-satisfying generative model. The value of negative data is largely governed by their relative difficulty. Hard negatives fall near the positive data manifold, while easy negatives lie deep within constrain-violating regions.

(iii) *Constraint Check*: A black-box 'oracle' that determines whether a design satisfies constraints is available. This check may be computationally expensive, limiting its use.

(iv) *Closed-form Constraints*: An inexpensive closed-form constraint is available. 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 (Cheng et al., 2021; Imrie et al., 2020).

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 limited dataset of negative samples is available (ii) or can be generated using an oracle (iii), but closed-form constraints are not available. This scenario is common in applications such as structural design, mobility design (e.g., cars, bikes, ships, airplanes), and material synthesis.

Density Ratio Estimation. Density Ratio Estimation (DRE) (Sugiyama et al., 2012b) is a critical technique in machine learning, particularly when evaluating distributions is not feasible or is computationally expensive (Mohamed & Lakshminarayanan, 2016). DRE techniques are heavily employed for generative modeling and score matching estimation (Goodfellow et al., 2014; Gutmann & Hyvärinen, 2010; Srivastava et al., 2023; Choi et al., 2022). In the context of GANs (Goodfellow et al., 2014; Arjovsky et al., 2017; **?**), the DRE methodology forms the underlying basis for their operation. A well-known technique for DRE is probabilistic classification Sugiyama et al. (2012b), where a binary classifier is used to learn the ratio. However, accurate DRE from finite samples can be challenging, especially in high dimensions. 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 (Gutmann & Hyvärinen, 2010; Rhodes et al., 2020), which divides the problem into a sequence of easier DRE sub-problems. Despite its success, there are limitations to this approach, especially when the number of intermediate bridge distributions is increased. Noise contrastive estimator (NCE (Gutmann & Hyvärinen, 2010)) and hybrid generative models (Srivastava et al., 2023; 2017; Rhodes et al., 2020) 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 (Regenwetter et al., 2022a). Generative Adversarial Nets, for example, have seen extensive use in many applications. In Topology Optimization, GANs (Li et al., 2019; Rawat & Shen, 2019; Oh et al., 2018; 2019; Sharpe & Seepersad, 2019; Nie et al., 2021; Yu et al., 2019; Valdez et al., 2021) are often used to create optimal topologies, bypassing the need for iterative solvers like SIMP. In computational materials design GANs (Tan et al., 2020; Yang et al., 2018; Zhang et al., 2021; Mosser et al., 2017; Lee et al., 2021; Liu et al., 2019), VAEs (Cang et al., 2018; Li et al., 2020; Liu et al., 2020; Wang et al., 2020; Xue et al., 2020; Tang et al., 2020; Chen & Liu, 2021), and other models are used to generate synthetic data to better learn process-structure-property relations (Bostanabad et al., 2018). A variety of generative models have been applied to 2D shape synthesis problems (Yilmaz & German, 2020; Chen & Fuge, 2018; Chen et al., 2019; Chen & Fuge, 2019; Nobari et al., 2022; Li et al., 2021; Dering et al., 2018), such as airfoil design, and 3D shape synthesis problems (Shu et al., 2020; Nobari et al., 2022; Brock et al., 2016; Zhang et al., 2019) 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 (Deshpande & Purwar, 2019; Sharma & Purwar, 2020; Regenwetter et al., 2021; Deshpande & Purwar, 2020).

Constraint Satisfaction in Machine Learning. From a general point of view, Constraint Satisfaction Problems (CSPs) have been long studied in computer science and optimization about optimal allocation, graph search, games, and path planning (Russell, 2010). However, such constraints are mostly related to algorithmic complexity and memory allocation. In generative design (Regenwetter et al., 2022a; Sigmund & Maute, 2013), 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. Recently, Neural Constraint Satisfaction (Chang et al., 2023) has been proposed to deal with objects in a scene to solve intuitive physics problems (Smith et al., 2019a; Hamrick et al., 2018). In the CAD domain, structured models to handle constraints have been proposed (Seff et al., 2021; Para et al., 2021). Conditional generative models have been proposed for structural topology optimization (Nie et al., 2021), leveraging physical fields (Nie et al., 2021; Mazé & Ahmed, 2023), dense approximations (Giannone & Ahmed, 2023), and trajectory alignment (Giannone et al., 2024) 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.

Anomaly Detection. Anomaly detection attempts to solve a one-class classification problem (anomalous vs not) (Sabuhi et al., 2021), much like constraint handling in generative models (positive vs negative). Several approaches have generated synthetic negative data as a stand-in for anomalies to train models (Zaheer et al., 2020; Dionelis et al., 2022). However, unlike anomaly detection, active constraint handling focuses on training a generative model to avoid negative samples, rather than simply identifying them. Furthermore, the point of training on negative data is to avoid the challenging one-class classification problem. Negative data has also been studied in the context of retrieval, using triplet losses Kaya & Bilge (2019) and contrastive estimators (Gutmann & Hyvärinen, 2010) for representation learning Oord et al. (2018); Chen et al. (2020).

B Negative Data Derivations& Density Ratio

Let p_n denote the *negative distribution* i.e., the distribution of constraint-violating datapoints. Instead of training using only the positive distribution p_p, we now seek to train a generative model p_θ using both p_p and p_n. Assuming mutual absolute continuity of p_p, p_θ and p_n, and starting from first principles, we can now re-write Eq. 2 as:

$$\begin{aligned} & \arg\min_\theta \int p_\theta(\boldsymbol{x}) \left[\log p_\theta(\boldsymbol{x}) - \log p_p(\boldsymbol{x})\right] d\boldsymbol{x} \\ = \; & \arg\min_\theta \int p_\theta(\boldsymbol{x}) \left[\log p_\theta(\boldsymbol{x}) - \log p_p(\boldsymbol{x}) + \left(\log \frac{p_n(\boldsymbol{x})}{p_n(\boldsymbol{x})}\right)\right] d\boldsymbol{x} \\ = \; & \arg\min_\theta \int p_\theta(\boldsymbol{x}) \left[\log \frac{p_\theta(\boldsymbol{x})}{p_n(\boldsymbol{x})} - \log \frac{p_p(\boldsymbol{x})}{p_n(\boldsymbol{x})}\right] d\boldsymbol{x}. \end{aligned} \tag{13}$$

While the solution for Eq. 13 is the same as the solution for equation 2 i.e. $p_{\theta^*} = p_p$, the model is now directly incentivized to allocate the same amount of probability mass to the samples from p_n as does the data distribution p_p. This ensures that when trained using finite N, the model does not allocate high probability mass to invalid samples. In other words, training under Eq. 13 encourages the model to minimize its discrepancy with respect to p_p such that its discrepancy with respect to p_n matches exactly that of p_p and p_n. Another important benefit of the reformulation in Eq. 13 is that in cases where sampling from p_n is inexpensive (such as in the engineering design domain), the sample efficiency of the model with respect to samples from p_p improves as shown in the next section.

B.1 Double Discriminator (DD) Formulation

We now re-write Eq. 13 using the equivalent formulation:

$$\arg\min_\theta \int p_\theta(\boldsymbol{x}) \left(\frac{1}{2}\left[\log \frac{p_\theta(\boldsymbol{x})}{p_n(\boldsymbol{x})} - \log \frac{p_p(\boldsymbol{x})}{p_n(\boldsymbol{x})}\right] + \frac{1}{2}\left[\log \frac{p_\theta(\boldsymbol{x})}{p_p(\boldsymbol{x})}\right]\right) d\,\boldsymbol{x}. \tag{14}$$

Our goal is to minimize the Kullback-Leibler (KL) divergence between our model $p_\theta(\boldsymbol{x})$ and the actual distribution $p_p(\boldsymbol{x})$, while simultaneously distancing our model $p_\theta(\boldsymbol{x})$ from any invalid designs represented by $p_n(\boldsymbol{x})$. Given that we lack access to the explicit functional form of this distribution, we employ density ratio estimation Sugiyama et al. (2012b); Srivastava et al. (2023) as a means of model learning.

In particular, we use f_ϕ to estimate $r(p_p, p_\theta) = \dfrac{p_p(\boldsymbol{x})}{p_\theta(\boldsymbol{x})}$, f_ψ to estimate $r(p_\theta, p_n) = \dfrac{p_\theta(\boldsymbol{x})}{p_n(\boldsymbol{x})}$, and f_ξ to estimate $r(p_p, p_n) = \dfrac{p_p(\boldsymbol{x})}{p_n(\boldsymbol{x})}$.

Leveraging the connection between density ratio and probabilistic classification (Sugiyama et al., 2012b), we can write (assuming balanced classes):

$$r(\boldsymbol{x}) = \frac{p_p(\boldsymbol{x})}{p_n(\boldsymbol{x})} = \frac{p_p(\boldsymbol{y}|\boldsymbol{x})}{p_n(\boldsymbol{y}|\boldsymbol{x})} = \frac{p_p(\boldsymbol{y}|\boldsymbol{x})}{1 - p_p(\boldsymbol{y}|\boldsymbol{x})}, \tag{15}$$

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

In situations where $p_p(\boldsymbol{x})$ and $p_n(\boldsymbol{x})$ 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:

$$p_p(\boldsymbol{y}|\boldsymbol{x}) = \sigma(\log r(\boldsymbol{x})). \tag{16}$$

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:

$$\begin{aligned} f_\phi(\boldsymbol{x}; p_p, p_\theta) &= \sigma(\log r_\phi(\boldsymbol{x})) \\ f_\psi(\boldsymbol{x}; p_\theta, p_n) &= \sigma(\log r_\psi(\boldsymbol{x})) \\ f_\xi(\boldsymbol{x}; p_p, p_n) &= \sigma(\log r_\xi(\boldsymbol{x})), \end{aligned} \tag{17}$$

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

$$\begin{aligned} \mathcal{F}_\phi(\boldsymbol{x}; \theta) &= \mathbb{E}_{p_p(\boldsymbol{x})} \log\left[f_\phi(\boldsymbol{x})\right] + \mathbb{E}_{p_\theta(\boldsymbol{x})} \log\left[1 - f_\phi(\boldsymbol{x})\right] \\ &= \mathbb{E}_{p_p(\boldsymbol{x})} \log\left[\sigma(\log r_\phi(\boldsymbol{x}))\right] + \mathbb{E}_{p_\theta(\boldsymbol{x})} \log\left[1 - \sigma(\log r_\phi(\boldsymbol{x}))\right]. \end{aligned} \tag{18}$$

The density ratio can be estimated by sampling from $p_p(\boldsymbol{x})$ and $p_\theta(\boldsymbol{x})$, and subsequently learning a discriminator f_ϕ using these samples. In practice, p_θ is learned leveraging adversarial training Goodfellow et al. (2014) and an auxiliary set of classifiers to push away samples from p_n. 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.

$$\begin{aligned} &\max_\phi \mathcal{F}_\phi(\boldsymbol{x}; \theta) \\ &\max_{\psi,\xi} \mathcal{F}_\psi(\boldsymbol{x}; \theta) + \mathcal{F}_\xi(\boldsymbol{x}; \theta) \\ &\min_\theta \mathcal{F}_\phi(\boldsymbol{x}; \theta) - \mathcal{F}_{\psi,\xi}(\boldsymbol{x}; \theta). \end{aligned} \tag{19}$$

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 is crucial to note that in the absence of parameter sharing, it is not strictly necessary to jointly train $\mathcal{F}_\xi$. If we manage to learn a robust model using ϕ and θ, then p_θ approximates p_p, 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 shared weights and supplied 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 – the same GAN-DD-b model presented in the main paper. See Algorithm 3 for training details.

The alternate double discriminator formulation (GAN-DD-a) is a variant of this approach:

$$\arg\min_\theta \int p_\theta(\boldsymbol{x}) \left(-\lambda \left[\log \frac{p_\theta(\boldsymbol{x})}{p_n(\boldsymbol{x})} \right] + \left[\log \frac{p_\theta(\boldsymbol{x})}{p_p(\boldsymbol{x})} \right] \right) d\boldsymbol{x}. \tag{20}$$

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 also 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 2 for training details.

B.2 Multi-Class Discriminator (MC) Formulation

Noting that multi-class classifiers are strong density ratio estimators (Srivastava et al., 2023), we also propose a variant using a multiclass discriminator model. By defining a single multiclass classifier f_ϕ and assigning pseudo-labels 2 to invalid, 1 to valid, and 0 to generated samples, we can write:

$$\mathcal{F}_\phi^{\texttt{MC}}(\boldsymbol{x}; \theta) = \mathbb{E}_{p_p(\boldsymbol{x})} \log\left[f_\phi(\boldsymbol{x})\right] + \mathbb{E}_{p_n(\boldsymbol{x})} \log\left[f_\phi(\boldsymbol{x})\right] + \mathbb{E}_{p_\theta(\boldsymbol{x})} \log\left[f_\phi(\boldsymbol{x})\right], \tag{21}$$

where we assume one-hot-encoding for the classes and cross-entropy loss as a scoring mechanism. We can then maximize this loss with respect to ϕ, learning good discriminators between valid, invalid, and generated. and minimize it with respect to θ. Given that we are using a single classifier, the discriminator is implicitly estimating all the relevant ratios (Srivastava et al., 2023). Pushing the generated samples close to valid will also push them far from invalid samples. We experiment with this formulation on the 2d densities with good results. However, when in the presence of more complex distributions and constraints variety, it is reasonable to allocate different levels of capacity for model learning and constraint satisfaction, using different discriminators. Additionally, when we want to generalize our methods to generative models other than GANs, like DDPM, it makes sense to instantiate a separate classifier for guidance with the only focus on fulfilling the constraints. See Algorithm 1 for training details.

B.3 Comparison to AC-GAN

Since our double-descriminator variant has two discriminative models, including one that learns to distinguish positive and negative samples, it bears a superficial similarity to AC-GANs (Odena et al., 2017) applied to binary class-conditional problems. In this section, we demonstrate that GAN-DD is distinct from AC-GANs. Consider an AC-GAN model applied as an NDGM. AC-GAN trains two discriminative models which share some weights, f_ϕ and f_ψ, which learn two ratios:

$$r_\phi(\boldsymbol{x}) = \frac{p_{real}(\boldsymbol{x})}{p_\theta(\boldsymbol{x})} \tag{22}$$

$$r_\psi(\boldsymbol{x}) = \frac{p_p(\boldsymbol{x})}{p_n(\boldsymbol{x})} \tag{23}$$

An important observation is that AC-GAN's "real" class consists of the full distribution across all classes, hence is comprised of both positive and negative data: $p_{real}(\boldsymbol{x}) = p_p(\boldsymbol{x}) + p_n(\boldsymbol{x})$.

The optimal discriminators now learn:

$$f_\phi(\boldsymbol{x}) = \frac{p_p(\boldsymbol{x}) + p_n(\boldsymbol{x})}{p_p(\boldsymbol{x}) + p_n(\boldsymbol{x}) + p_\theta(\boldsymbol{x})} \tag{24}$$

$$f_\psi(\boldsymbol{x}) = \frac{p_p(\boldsymbol{x})}{p_p(\boldsymbol{x}) + p_n(\boldsymbol{x})} \tag{25}$$

Here α is a parameter determined by the ratio of the classes. Assuming perfect discriminators, the generator loss is then expressed as:

$$\begin{aligned}
\mathcal{L}(\theta, \phi, \psi) &= \mathbb{E}_{p_p(\boldsymbol{x})+p_n(\boldsymbol{x})}[\log f_\psi(\boldsymbol{x})] + \mathbb{E}_{p_\theta(\boldsymbol{x})}[1 - \log f_\psi(\boldsymbol{x}_\theta)] \\
&\quad -\mathbb{E}_{p_p(\boldsymbol{x})+p_n(\boldsymbol{x})}[\log(f_\phi(\boldsymbol{x}))] - \mathbb{E}_{p_\theta(\boldsymbol{x})}[\log(f_\phi(\boldsymbol{x}_\theta))] \\
&= \mathbb{E}_{p_\theta(\boldsymbol{x})}\left[\log \frac{p_p(\boldsymbol{x}_\theta)}{p_p(\boldsymbol{x}_\theta) + p_n(\boldsymbol{x}_\theta)} - \left[1 - \log \frac{p_p(\boldsymbol{x}_\theta) + p_n(\boldsymbol{x}_\theta)}{p_p(\boldsymbol{x}_\theta) + p_n(\boldsymbol{x}_\theta) + p_\theta(\boldsymbol{x}_\theta)}\right]\right]
\end{aligned} \tag{26}$$

In contrast, the GAN-MC formulation's discriminator learns three density ratios. One of these is the reciprocal of r_ϕ (eq. 22).

$$r_{\zeta,\theta}(\boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x})}{p_p(\boldsymbol{x}) + p_n(\boldsymbol{x})} \tag{27}$$

However, the ratio that is actually used for generator training is a different ratio:

$$r_{\zeta,p}(\boldsymbol{x}) = \frac{p_p(\boldsymbol{x})}{p_n(\boldsymbol{x}) + p_\theta(\boldsymbol{x})} \tag{28}$$

This leads an optimal discriminator to learn:

$$f_{\zeta,p}(\boldsymbol{x}) = \frac{p_p(\boldsymbol{x})}{p_p(\boldsymbol{x}) + p_n(\boldsymbol{x}) + p_\theta(\boldsymbol{x})} \tag{29}$$

Assuming the discriminator is perfectly optimal, the generator optimizes:

$$\mathcal{L}(\theta, \zeta) = \mathbb{E}_{p_p(\boldsymbol{x}_\theta)}[\log f_{\zeta,p}(\boldsymbol{x}_\theta)] = \mathbb{E}_{p_p(\boldsymbol{x}_\theta)}\left[\log \frac{p_p(\boldsymbol{x}_\theta)}{p_p(\boldsymbol{x}_\theta) + p_n(\boldsymbol{x}_\theta) + p_\theta(\boldsymbol{x}_\theta)}\right]. \tag{30}$$

This is indeed equivalent to eq. 26, indicating that AC-GAN can also function as an NDGM. However, it is critical to observe that this equivalence is contingent on **perfect discriminative models**. It does not indicate that the methods are equivalent. In fact, it can be easily seen that many of the other NDGMs tested in the paper (simple baselines, existing NDGMs, etc.) have this exact final loss, contingent on perfect models. The difference in performance between different adversarial NDGM formulations arises from the difference in discriminator training and discriminative performance.

C Pseudocode

Psuedocode for the multiclass discriminator (GAN-MC) and double discriminator variants (GAN-DD-a, GAN-DD-b) are shown below. Diversity loss is included, and is discussed in section E. For non-diversity-augmented variants, γ is set to 0.

Algorithm 1 GAN-MC Training Procedure

while $step \leq n_{steps}$ **do**
  Sample $P_{batch} \sim P_{dataset}$ and $N_{batch} \sim N_{dataset}$
  Sample $\epsilon \sim N(0, 1)$
  G_{batch} = Generator(ϵ)
  D^P_{preds} = Discriminator(P_{batch})
  D^N_{preds} = Discriminator(N_{batch})
  D^G_{preds} = Discriminator(G_{batch})
  loss_fn = CategoricalCrossEntropy()
  D_loss = loss_fn(D^G_{preds}, 0) + loss_fn(D^P_{preds}, 1) + loss_fn(D^N_{preds}, 2)
  $Diversity\_loss$ = DPP(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

Algorithm 2 GAN-DD-a Training Procedure

while $step \leq n_{steps}$ **do**
  Sample $P_{batch} \sim P_{dataset}$ and $N_{batch} \sim N_{dataset}$
  Sample $\epsilon \sim N(0, 1)$
  G_{batch} = Generator(ϵ)
  D^P_{preds} = Discriminator(P_{batch})
  D^G_{preds} = Discriminator(G_{batch})
  A^I_{preds} = Aux_Discriminator(N_{batch})
  A^G_{preds} = Aux_Discriminator(G_{batch})
  loss_fn = BinaryCrossEntropy()
  D_loss = loss_fn(D^G_{preds}, 0) + loss_fn(D^P_{preds}, 1)
  A_loss = loss_fn(A^G_{preds}, 0) + loss_fn(A^N_{preds}, 1)
  $Diversity\_loss$ = DPP(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

Algorithm 3 GAN-DD-b Training Procedure

while $step \leq n_{steps}$ **do**
  Sample $P_{batch} \sim P_{dataset}$ and $N_{batch} \sim N_{dataset}$
  Sample $\epsilon \sim N(0, 1)$
  G_{batch} = Generator(ϵ)
  D^P_{preds} = Discriminator(P_{batch})
  D^G_{preds} = Discriminator(G_{batch})
  A^P_{preds} = Aux_Discriminator(P_{batch})
  A^N_{preds} = Aux_Discriminator(N_{batch})
  A^G_{preds} = Aux_Discriminator(G_{batch})
  loss_fn = BinaryCrossEntropy()
  D_loss = loss_fn(D^G_{preds}, 0) + loss_fn(D^P_{preds}, 1)
  A_loss = 0.5 · loss_fn(A^G_{preds}, 0) + 0.5 · loss_fn(A^P_{preds}, 0) + loss_fn(A^N_{preds}, 1)
  $Diversity\_loss$ = DPP(G_{batch})
  G_loss = loss_fn(D^G_{preds}, 1) + β · loss_fn(A^G_{preds}, 1) + γ · $Diversity\_loss$
  Optimize(Discriminator, D_loss)
  Optimize(Aux_Discriminator, A_loss)
  Optimize(Generator, G_loss)
  $step = step + 1$
end while

D 2D Densities

D.1 2D Datasets and Test Problems

We discuss more details on the 2D datasets presented in Sec. 5.1.

D.1.1 2D Problem 1

This problem is labeled `Problem 1` in the main paper. Data points are randomly sampled from one of six modes, each of which is a regular 2D gaussian. Distribution centers are spaced at an equal radius. Points in a close proximity to the center of any of the distribution are labeled as negatives and others are labeled as positives. Sampling is performed until 10k positive samples and 10k negative samples are acquired and excess of the oversampled class are discarded.

D.1.2 2D Problem 2

This problem is labeled `Problem 2` in the main paper. Datapoints are uniformly sampled. A square grid of 'centerpoints' is overlaid over the distribution. Any datapoint in a close enough proximity to a 'centerpoint' is considered negative, while any others are considered positive. Sampling is performed until 10k positive samples and 10k negative samples are acquired and excess of the oversampled class are discarded.

D.2 Setup and Training

All tested networks (encoder, decoder, generator, DDPM noise model, auxiliary discriminator) are deep networks with two hidden layers of 400 neurons each and ReLU activations. A batch size of 256 is used throughout. Models are trained using the Adam optimizer (Kingma & Ba, 2014) with a learning rate $3e^{-4}$, $5e^{-4}$, and $1e^{-3}$ for GANs, VAEs, and DDPMs respectively. Models are trained for 3500 epochs. 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 the validity weight parameter λ is set at 0.2. The GAN's validity weight parameter λ is set at 0.4.

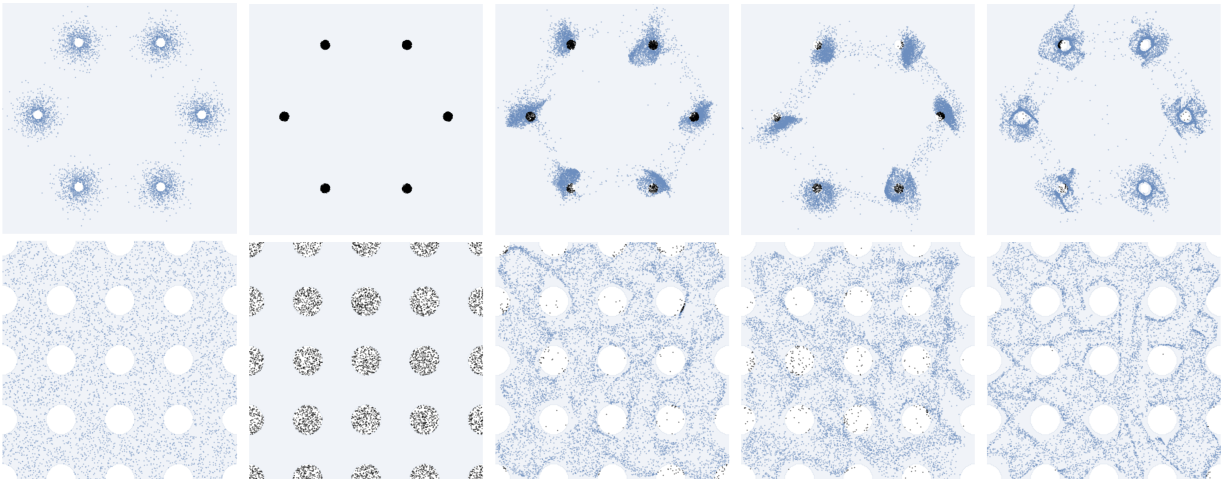
D.3 Additional Results

We include a set of results on 2D density experiments expanding on our results from the main paper. Discriptions of the models can be found in the main paper. Mean scores and standard deviations are reported over 3 instantiations in Table 6.

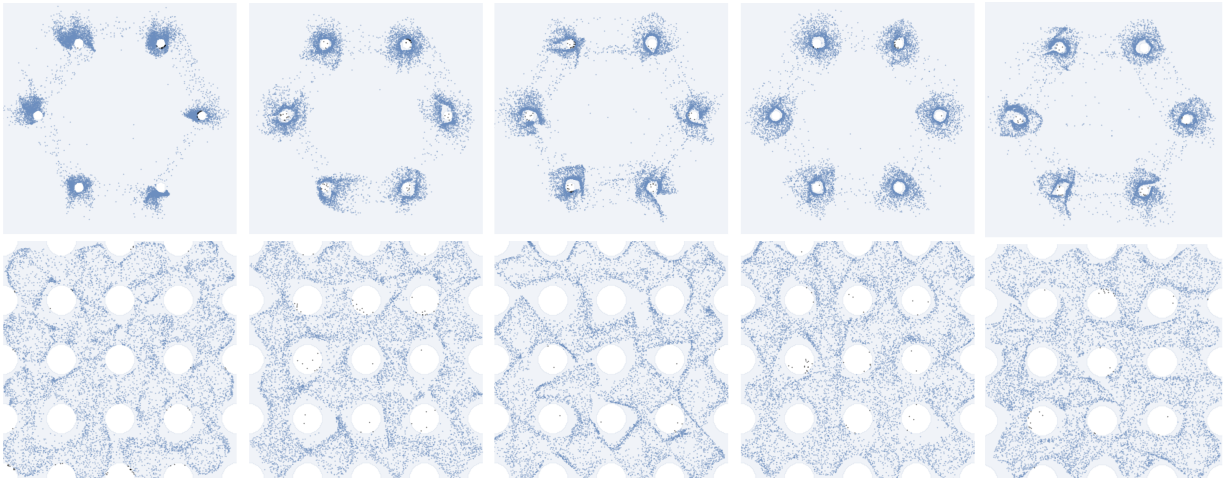
Evaluation Metrics. We utilize several of the metrics proposed in (Regenwetter et al., 2023) for constraint-satisfaction and distributional similarity in generative models. To measure performance, we calculate several scores obtained from precision-recall curves (Sajjadi et al., 2018), including F1, F0.1, F10, and the area under the curve (abv. AUC-PR). F0.1 roughly captures coverage (the degree to which the generated distribution covers the span of the dataset), while F10 roughly captures precision (not straying beyond the boundaries of the data). F1 and AUC-PR roughly serve to assess overall distributional similarity. We calculate the mean distance to the nearest dataset point for each generated sample as another 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) as a simple estimate for coverage. Finally, we calculate Maximum mean Discrepancy and proportion of invalid generated samples (abv. Validity).

Table 6: Extended results for 2D density experiments. Mean scores and standard deviations over three instantiations are shown. Best models are determined using a two-sample t-test with 95% confidence and boldfaced. Our models (GAN-DD, GAN-MC, GAN-RM) surpass the previous state of the art (GAN-DO) in most metrics.

Metric:	**Validity ↓**	**MMD ↓**	**NDS ↓**	**NGS ↓**	**F1 ↑**	**F10 ↑**	**F0.1 ↑**	**AUC-PR ↑**
				Problem 1				
GAN	0.093±0.028	0.008±0.002	0.018±0.002	0.027±0.003	0.725±0.022	0.968±0.010	0.931±0.051	0.800±0.035
GAN-CC	0.074±0.018	0.005±0.002	0.016±0.000	0.020±0.007	0.821±0.071	0.982±0.008	0.947±0.062	0.876±0.097
GAN-CL	**0.005±0.005**	**0.003±0.001**	0.014±0.001	0.014±0.000	**0.943±0.013**	**0.997±0.001**	0.996±0.000	**0.990±0.003**
GAN-Rej	0.008±0.005	0.010±0.003	0.015±0.001	0.021±0.003	0.751±0.035	0.971±0.005	0.925±0.025	0.822±0.045
VAE	0.134±0.013	0.004±0.000	0.033±0.001	0.013±0.000	0.828±0.000	0.932±0.005	0.988±0.001	0.888±0.004
VAE-CC	0.191±0.025	0.004±0.000	0.030±0.002	0.026±0.002	0.800±0.013	0.943±0.004	0.986±0.002	0.874±0.012
VAE-CL	**0.001±0.001**	0.005±0.000	0.030±0.001	0.039±0.004	0.828±0.009	0.925±0.007	0.988±0.002	0.881±0.010
VAE-Rej	0.024±0.010	0.006±0.001	0.031±0.002	0.017±0.004	0.826±0.013	0.924±0.006	0.989±0.001	0.872±0.011
DDPM	0.083±0.003	**0.002±0.000**	0.018±0.000	**0.010±0.000**	0.912±0.003	0.994±0.000	**0.997±0.000**	0.978±0.001
DDPM-CL	0.120±0.004	**0.002±0.000**	0.019±0.000	**0.010±0.000**	0.893±0.005	0.995±0.000	0.995±0.000	0.973±0.002
DDPM-G	0.038±0.000	0.003±0.000	0.038±0.000	**0.010±0.000**	0.867±0.003	0.981±0.001	0.995±0.000	0.947±0.002
DDPM-Rej	0.012±0.006	**0.002±0.000**	0.017±0.000	**0.010±0.000**	0.895±0.007	0.994±0.001	0.996±0.000	0.972±0.003
GAN-DO	0.004±0.001	0.005±0.002	**0.014±0.000**	0.015±0.001	0.912±0.013	0.992±0.001	0.993±0.004	0.974±0.008
GAN-MC	0.005±0.001	**0.003±0.001**	0.016±0.000	0.018±0.002	**0.915±0.023**	0.991±0.002	0.993±0.001	0.976±0.010
GAN-DD-a	**0.004±0.005**	**0.002±0.000**	**0.015±0.001**	0.015±0.003	**0.924±0.036**	**0.994±0.003**	**0.986±0.016**	**0.973±0.027**
GAN-DD-b	**0.002±0.000**	**0.002±0.000**	**0.016±0.002**	0.015±0.002	**0.928±0.015**	**0.992±0.005**	0.995±0.001	**0.983±0.008**
				Problem 2				
GAN	0.018±0.007	**0.002±0.000**	**0.022±0.001**	0.020±0.000	0.964±0.004	**0.998±0.000**	**0.998±0.000**	**0.996±0.001**
GAN-CC	0.041±0.011	**0.002±0.001**	0.024±0.001	0.022±0.003	0.959±0.012	**0.997±0.001**	0.997±0.001	**0.994±0.003**
GAN-CL	**0.006±0.004**	**0.002±0.000**	0.022±0.000	0.021±0.001	0.958±0.007	**0.997±0.001**	0.997±0.000	0.994±0.002
GAN-Rej	**0.003±0.002**	**0.002±0.000**	**0.021±0.000**	0.020±0.000	0.969±0.002	**0.998±0.000**	**0.998±0.001**	**0.997±0.000**
VAE	0.104±0.002	**0.002±0.000**	0.029±0.000	0.018±0.000	0.960±0.002	**0.998±0.000**	**0.998±0.000**	0.995±0.000
VAE-CC	0.101±0.003	**0.002±0.000**	0.029±0.000	0.018±0.000	0.958±0.003	0.997±0.000	**0.998±0.000**	0.995±0.000
VAE-CL	0.005±0.001	0.005±0.001	0.023±0.000	0.050±0.004	0.901±0.021	0.993±0.001	0.991±0.003	0.969±0.009
VAE-Rej	**0.004±0.001**	**0.002±0.000**	0.022±0.000	**0.018±0.001**	**0.972±0.003**	**0.998±0.000**	**0.998±0.001**	**0.997±0.000**
DDPM	0.068±0.006	0.006±0.001	0.026±0.000	**0.017±0.000**	0.907±0.005	0.995±0.000	0.995±0.001	0.974±0.003
DDPM-CL	0.069±0.005	0.005±0.000	0.026±0.000	**0.017±0.000**	0.911±0.002	0.995±0.000	0.995±0.000	0.975±0.002
DDPM-G	0.065±0.002	0.005±0.000	0.026±0.000	**0.017±0.000**	0.907±0.001	0.994±0.001	0.995±0.000	0.973±0.000
DDPM-Rej	**0.006±0.004**	0.005±0.000	0.022±0.000	**0.017±0.000**	0.914±0.006	0.995±0.000	0.995±0.001	0.977±0.003
GAN-DO	**0.004±0.002**	**0.003±0.001**	0.022±0.000	0.025±0.001	0.948±0.012	**0.997±0.001**	0.996±0.001	0.991±0.003
GAN-MC	**0.002±0.001**	**0.002±0.000**	0.022±0.000	0.027±0.002	0.953±0.013	**0.997±0.001**	**0.997±0.001**	**0.993±0.004**
GAN-DD-a	**0.003±0.001**	**0.003±0.001**	0.022±0.000	0.022±0.002	0.949±0.004	0.997±0.000	**0.997±0.001**	0.992±0.001
GAN-DD-b	**0.004±0.001**	**0.002±0.001**	0.022±0.000	0.022±0.000	**0.962±0.008**	**0.997±0.001**	**0.997±0.001**	**0.995±0.002**

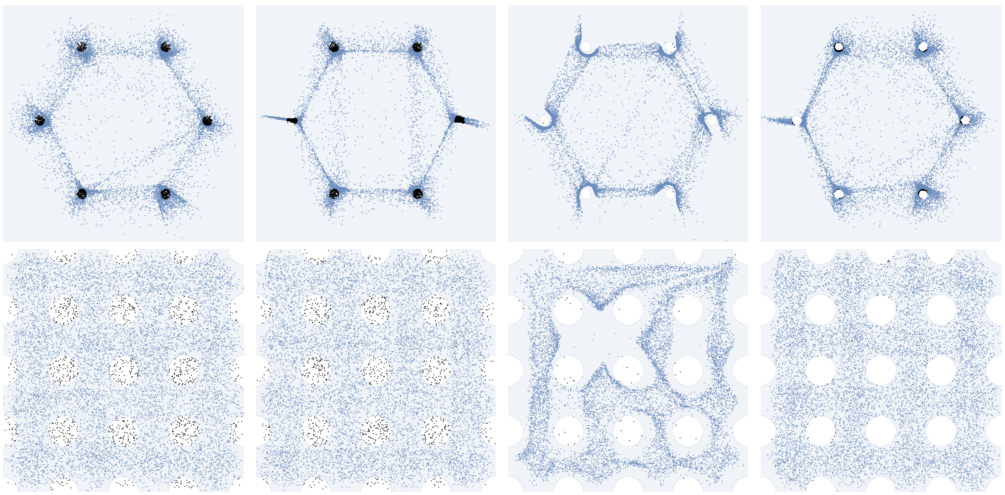


(a) Positive Data **(b)** Negative Data **(c)** GAN **(d)** GAN-CC **(e)** GAN-CL

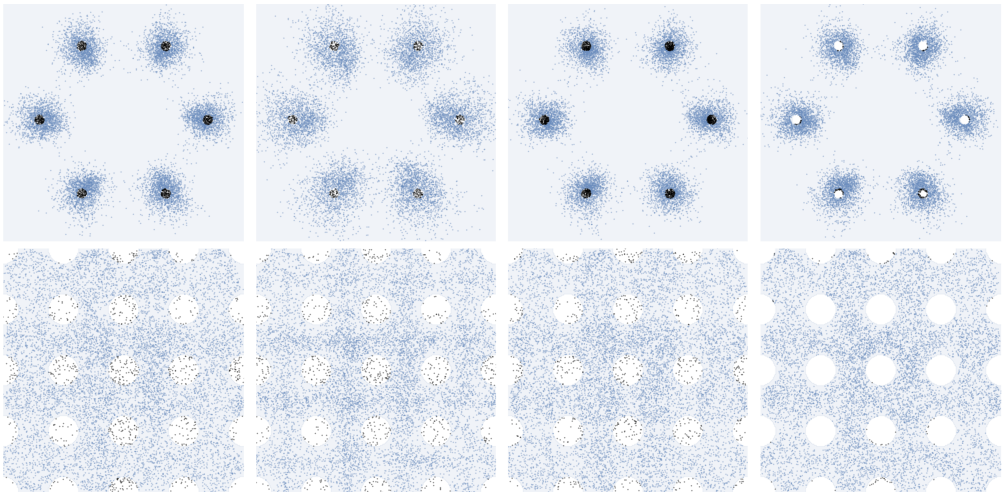


(f) GAN-Rej **(g)** GAN-DO **(h)** GAN-MC (ours) **(i)** GAN-DD-a (ours) **(j)** GAN-DD-b (ours)

Figure 6: Extended Results on 2 toy 2D densities. Positive and Negative Datasets are shown in the first two panes, respectively. Valid (positive) generated samples are colored blue, while invalid (negative) generated samples are colored black. Constraint-satisfying regions are indicated in the background of plots in blue, while constraint-violating regions are colored white.



(a) VAE **(b)** VAE-CC **(c)** VAE-CL **(d)** VAE-Rej



(e) DDPM **(f)** DDPM-G **(g)** DDPM-CL **(h)** DDPM-Rej

Figure 7: Extension of Figure 6.

E Encouraging Diverse Generation in NDGMs

NDGMs tend to have high precision, but may struggle with recall. This tendancy arises because a conservative NDGM will avoid fringe regions of the distribution, resulting in incomplete coverage. 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 (Chen & Ahmed, 2021b;a; Regenwetter & Ahmed, 2022; Regenwetter et al., 2023). As (Chen & Ahmed, 2021b) and (Chen & Ahmed, 2021a) note, incorporating diversity can also help models generalize and avoid mode collapse. Diversity was first explicitly incorporated into deep generative models for design engineering in (Chen & Ahmed, 2021b) using a Determinantal Point Process (DPP). Determinantal Point Process (DPP)-based diversity measures have been used in a variety of generative applications in design (Chen & Ahmed, 2021b; Nobari et al., 2021) and elsewhere (Elfeki et al., 2019; Mothilal et al., 2020).

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:

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

where λ_i is the i-th eigenvalue of L and B is the number of samples in the batch. The loss is incorporated by appending it to the overall loss term of the generative model $\mathcal{L}_{GM}$

$$\mathcal{L}_{tot} = \mathcal{L}_{GM} + \gamma\,\mathcal{L}_{div}$$

Adding this loss can help the generative model achieve better coverage, an observation supported by our experiments below.

E.1 Methodology

We train a GAN-DD-a, a GAN-DD-b, and a GAN-MC model, each augmented with a diversity weight, as indicated in Sec C. These models are labeled GAN-DD-a-DA, GAN-DD-b-DA, and GAN-MC-DA, respectively. To demonstrate that other models can be similarly augmented with diversity, we also train a VAE-CL with diversity (VAE-CL-DA). Diversity weight γ is set at 0.7 for GANs and 0.05 for VAEs. Architecture, dataset, and training parameters are unchanged from Appendix D.2.

E.2 Results

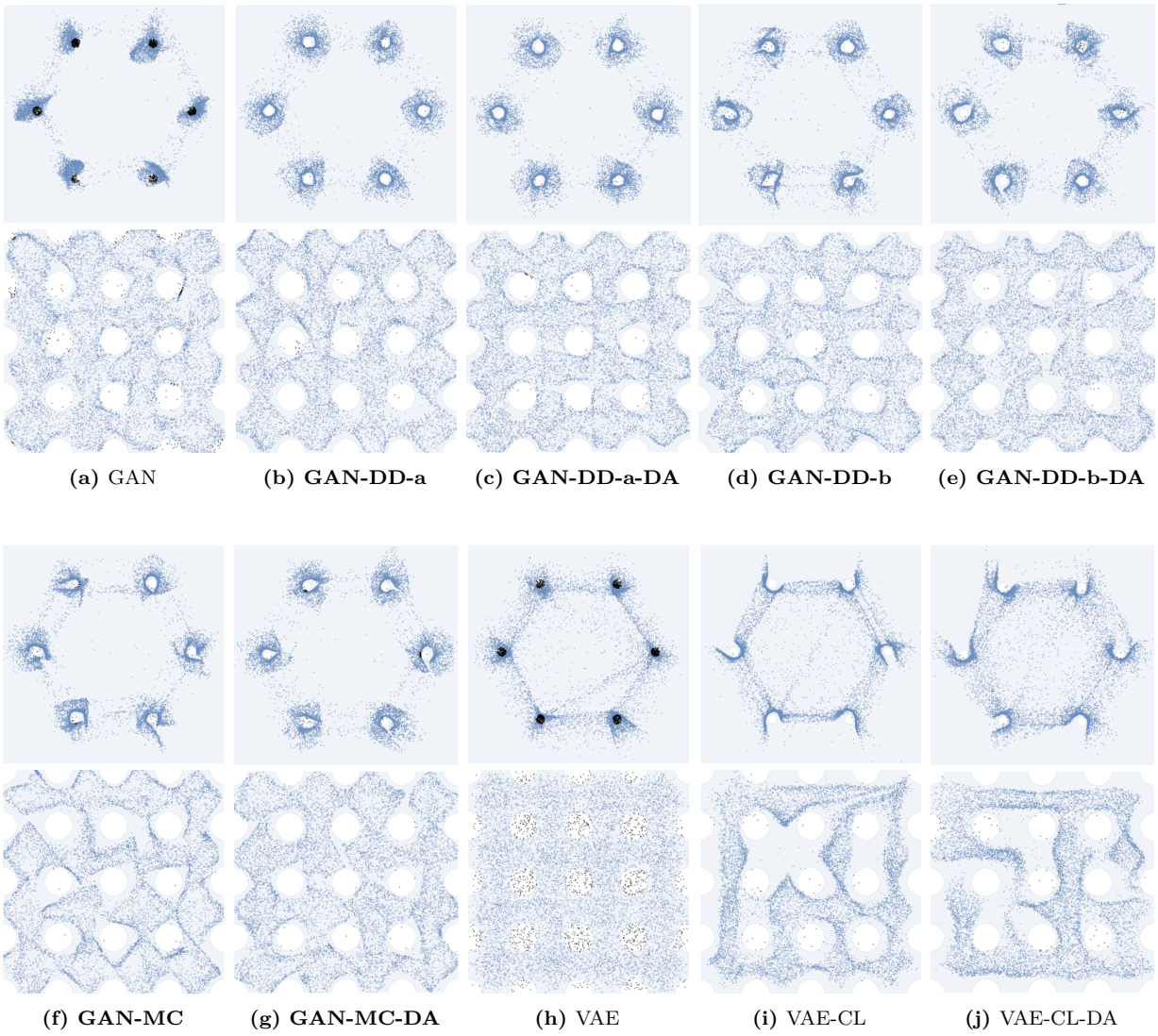
Table 7 shows scores over a variety of distributional similarity metrics and validity. The diversity-augmented double discriminator GAN variant is the strongest performer across most objectives. Plots of generated distributions are included in Figure 8. Visually, diversity-augmented models better distribute samples over the distribution, but sometimes generate more negative samples. We also include a percentage differences between diversity-augmented (DA) NDGMs and their non-DA counterparts in Table 8. Diversity improves the double discriminator (GAN-DD) variant on most metrics. However, the GAN-MC and VAE-CL models mainly improve only in recall-related scores (NGS, F0.1), while suffering in others, most notably validity.

Table 7: Table showing scores over various vanilla models, NDGMs, and Diversity-Augmented (DA) NDGMs across a variety of distributional similarity metrics and validity metric.

	Validity ↓	MMD ↓	NDS ↓	NGS ↓	F1 ↑	F10 ↑	F0.1 ↑	AUC-PR ↑
GAN-Vanilla	0.0622	0.0050	0.0199	0.0241	0.8417	0.9845	0.9807	0.9006
GAN-DD-a (ours)	0.0028	0.0022	0.0184	0.0177	0.9435	0.9960	0.9970	0.9918
GAN-DD-a-DA (ours)	**0.0023**	0.0025	**0.0182**	**0.0156**	**0.9556**	0.9970	**0.9974**	**0.9937**
GAN-DD-b (ours)	0.0028	**0.0020**	0.0188	0.0183	0.9491	0.9959	0.9964	0.9813
GAN-DD-b-DA (ours)	0.0026	0.0022	0.0191	0.0176	0.9514	**0.9972**	0.9963	0.9828
GAN-MC (ours)	0.0030	0.0023	0.0190	0.0220	0.9357	0.9941	0.9950	0.9854
GAN-MC-DA (ours)	0.0054	0.0022	0.0200	0.0170	0.9351	0.9949	0.9958	0.9848
VAE-Vanilla	0.1223	0.0032	0.0309	0.0158	0.8942	0.9662	0.9931	0.9431
VAE-CL	0.0029	0.0051	0.0261	0.0456	0.8619	0.9579	0.9895	0.9226
VAE-CL-DA	0.0035	0.0055	0.0295	0.0340	0.8608	0.9571	0.9925	0.9190

Table 8: Table showing percantage (%) differences between diversity-augmented (DA) NDGMs and their non-DA counterparts. Improvements are bolded. For objectives that are maximized at 1 (F1, F0.1, F10, AUC-PR), we calculate percentage difference as $(s_{new} - s_{old})/(1 - s_{old})$.

Percent (%) Difference	Validity ↓	MMD ↓	NDS ↓	NGS ↓	F1 ↑	F10 ↑	F0.1 ↑	AUC-PR ↑
GAN-DD-a-DA (vs. GAN-DD-a)	**-14.5**	13.3	**-0.8**	**-11.4**	**21.4**	**24.9**	**13.5**	**23.6**
GAN-DD-b-DA (vs. GAN-DD-b)	**-7.27**	8.9	1.6	**-3.5**	**4.6**	**32.0**	-1.7	**16.7**
GAN-MC-DA (vs. GAN-MC)	81.7	**-1.2**	5.2	**-22.6**	-1.0	**14.3**	**14.9**	-4.5
VAE-CL-DA (vs. VAE-CL)	22.8	8.3	13.1	**-25.5**	-0.8	-1.9	**28.6**	-4.7



(a) GAN **(b) GAN-DD-a** **(c) GAN-DD-a-DA** **(d) GAN-DD-b** **(e) GAN-DD-b-DA**

(f) GAN-MC **(g) GAN-MC-DA** **(h)** VAE **(i)** VAE-CL **(j)** VAE-CL-DA

Figure 8: Results demonstrating diversity-augmented (DA) NDGMs on two toy 2D densities. Positive and negative datasets are shown in the first two panes, respectively. Valid (positive) generated samples are colored blue, while invalid (negative) generated samples are colored black. Constraint-satisfying regions are indicated in the background of plots in blue, while constraint-violating regions are colored white. Diversity-augmented (DA) NDGMs achieve much better recall than their non-DA counterparts, though occasionally generate more negative samples. Our models are bolded.

F Block Stacking: Details and Additional Experiments

F.1 Training and Dataset Details

For GAN-DO, we select $\lambda = 0.75$ as the best-performing hyperparameter. For GAN-DD, we select $\lambda = 0.5$ as the best-performing hyperparameter. We also benchmark an autoregressive GAN model (GAN-AR) (Mogren, 2016; Esteban et al., 2017). The architecture, dataset, and training details are shown below.

Table 9: Relevant Hyperparameters for block stacking models. C: connectivity. S: stability. FM: floating material. VFE: volume fraction error. CE: compliance error.

	GAN-DO/GAN-DD	GAN-AR
Dimension	12	12
Valid Set	10K	10K
Invalid Set	10K	10K
Evaluation Set	1K	1K
Constraints	C+S	C+S
Generator	MLP (16-32-12)	LSTM (64-12)
Discriminator	MLP (128-64-32-1)	MLP (128-64-32-1)
Batch size	200	200
Iterations	30K	50K
Learning rate	$1e^{-3}$	$1e^{-3}$
Optimizer	Adam	Adam

Example positive and negative configurations from the dataset are visualized in Sec. F.3 and F.4

F.2 Fulfilling Multiple Sets of Constraints

We consider the more challenging problem where stacks must simultaneously satisfy connectivity and stablility constraints. S-C is used as positive data while the S-D, U-C, and U-D are pooled to constitute the negative data. Results are summarized in the lower half of Tables 11 and 12. We see that models trained using only positive data perform poorly in constraint satisfaction scores across the board. Autoregressive models (GAN-AR, (Mogren, 2016; Esteban et al., 2017)) tend to work better than standard GAN (second row) but still fall behind compared to GAN-DD. GAN-DD connects and stabilizes an order of magnitude more configurations than the GAN (fifth row) training on only positive (connected and balanced) configurations. Interestingly, GAN-AR (sixth row) and GAN-DO (seventh row) can achieve high scores on connectivity, with GAN-DO performing better than GAN-DD. However, when the model is challenged to fulfill both constraints, GAN-AR generates almost exclusively invalid configurations, and GAN-DO satisfies all constraints less than 1/4 as frequently as the GAN-DD, even if presented with the same amount of positive and negative data. We note that the benefits of negative data only extend to constraints that were included in the negative data. As shown in the upper half of Table 11, models trained on the relaxed case, including GAN-DD (fourth row) struggle to satisfy constraints not represented in the negative dataset since they do not see representative examples of unstable configurations in negative data.

Table 10: Base model vs Negative Data model trained with negative 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 negative data is 1 unit). b: base block; m: middle block; t: top block. For floating $\boldsymbol{y}_b < \boldsymbol{y}_m$ and for intersecting $\boldsymbol{y}_b > \boldsymbol{y}_m$.

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

Table 11: 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 negative data to GAN-DO and GAN-DD). When stability is not considered during training, no generative models can reliably fulfill the stability constraint. The lower half shows results where both disconnected stacks and unstable stacks are considered negative (and are provided to GAN-DO and GAN-DD). GAN-DD improves constraint satisfaction by an order of magnitude over most baselines.

	Positive Data		Negative Data		Metrics		
	Connected	Stable	Disconnected	Unstable	Stability $\uparrow$	Connectivity $\uparrow$	Both $\uparrow$
GAN	✓	✗	✗	✗	2.44 %	2.80 %	0.19 %
GAN-AR	✓	✗	✗	✗	0.085 %	13.23 %	0.00 %
GAN-DO	✓	✗	✓	✗	0.39 %	6.39 %	0.00 %
GAN-DD (ours)	✓	✗	✓	✗	**3.75** %	**45.30** %	**1.03** %
GAN	✓	✓	✗	✗	83.20 %	4.83 %	3.28 %
GAN-AR	✓	✓	✗	✗	32.16 %	8.12 %	0.07 %
GAN-DO	✓	✓	✓	✓	**84.65** %	12.63 %	8.85 %
GAN-DD (ours)	✓	✓	✓	✓	70.35 %	**41.10** %	**36.02** %

Table 12: Handling Multiple Sets of Constraints. Positive data is `connected` (constraint set I) and `stable` (constraint set II). Negative data is composed of two negative sets: Connected-Unstable and Disconnected-Stable.

	GAN (w/o Negative)	GAN-DD (w/ Negative)
$\uparrow$ Connected($\boldsymbol{y}_b$, $\boldsymbol{y}_m$) (Ia)	21.13 $\pm$ 3.04	**100** $\pm$ 0.00
$\uparrow$ Connected($\boldsymbol{y}_m$, $\boldsymbol{y}_t$) (Ib)	22.68 $\pm$ 3.06	**41.10** $\pm$ 4.79
$\uparrow$ Stable (II)	**83.20** $\pm$ 4.09	70.35 $\pm$ 3.72
$\uparrow$ Connected (I)	4.83 $\pm$ 1.72 (-88.24 %)	**41.10** $\pm$ 4.80
$\uparrow$ Connected and Stable (I and II)	3.28 $\pm$ 1.34 (- 90.89 %)	**36.02** $\pm$ 3.88

Table 13: Overview. Using negative data improves constraint satisfaction by an order of magnitude. "Pseudo-balanced" stacks would be `stable` if the other constraints were satisfied.

	Positive		Negative		Metrics		
	Connected	Balanced	Disconnected	Unbalanced	Pseudo-Balanced ↑	Balanced ↑	Connected ↑
Positive Set	✓	✗	✗	✗	2.50 %	-	100.00 %
Positive Set	✓	✓	✗	✗	0.00 %	100.00 %	100.00 %
Negative Set	✗	✗	✓	✗	0.00 %	100.00 %	0.00 %
Negative Set	✗	✗	✗	✓	2.50 %	0.00 %	100.00 %

F.3 Training Set - Positive

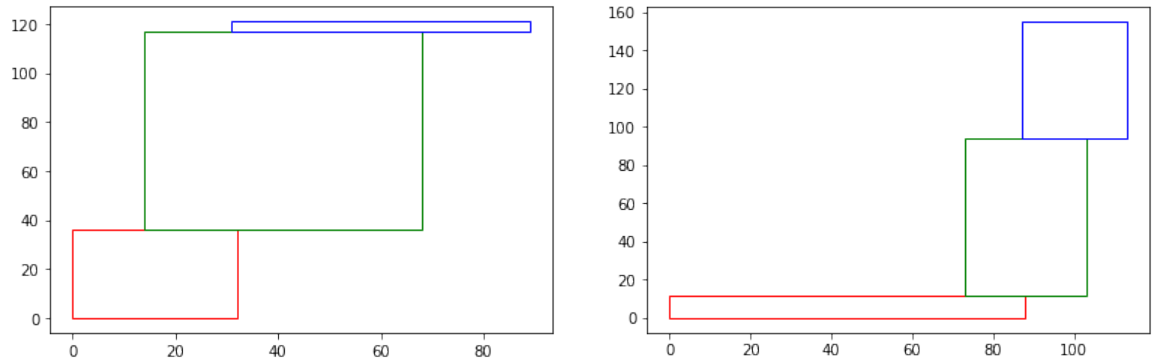


Figure 9: Positive Data for Constraint I. Example of positive configurations for constraints I (connection). Blocks fulfilling only the connection constraints (I). Specifically, the blocks are stacked one on top of each other without any floating or intersection but in general in unstable configurations. 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).

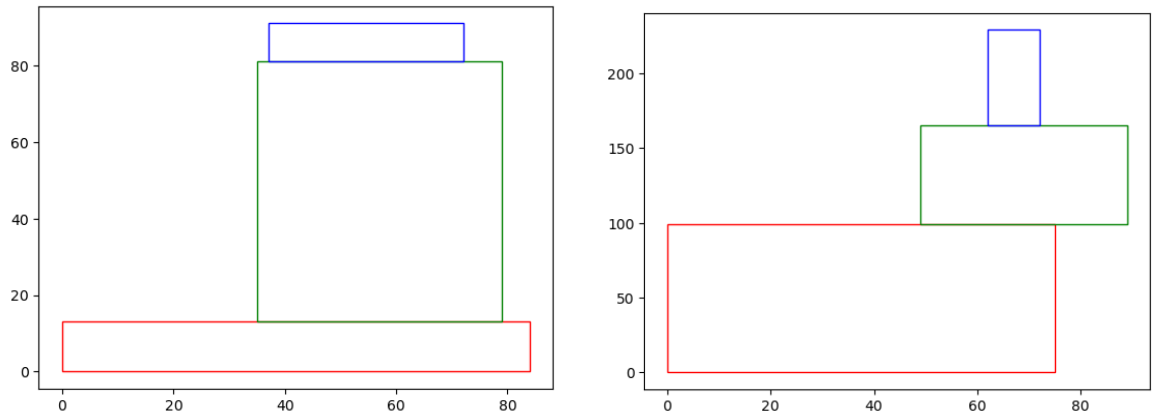


Figure 10: Positive Data for Constraints I+II. Example of positive configurations for constraints I (connection) and II (stability). Blocks fulfilling both constraints connection constraints (I). Specifically, the blocks are stacked one on top of each other without any floating or intersection in a stable 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.

F.4 Training Set - Negative

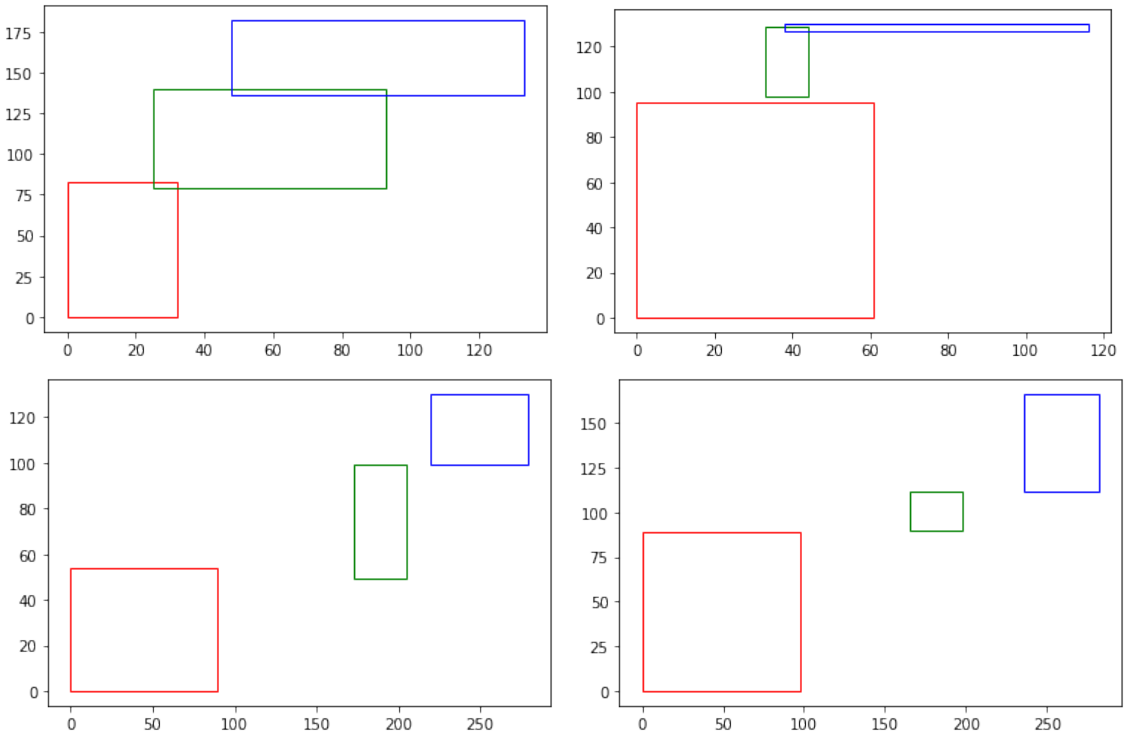


Figure 11: Negative Data. Example of negative data for block stacking. The negative data consists of two categories: hard-negative (top) and easy-negative (bottom). The top section of the figure contains examples of hard negative, which are generated by violating the constraints of the block stacking 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-negatives 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-negative, 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.

F.5 Samples

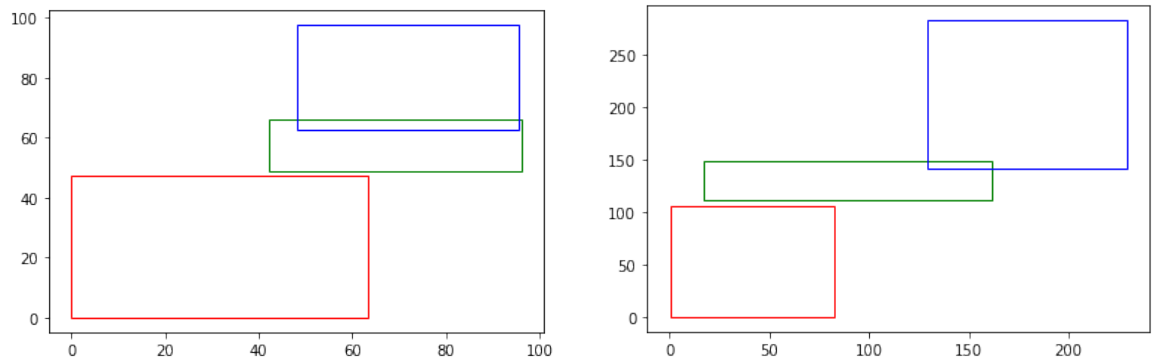


Figure 12: GAN Samples. Samples from a model trained only on positive data. The model generates reasonable samples but in most cases, the constraints are not satisfied. Precision at the boundary is challenging to enforce.

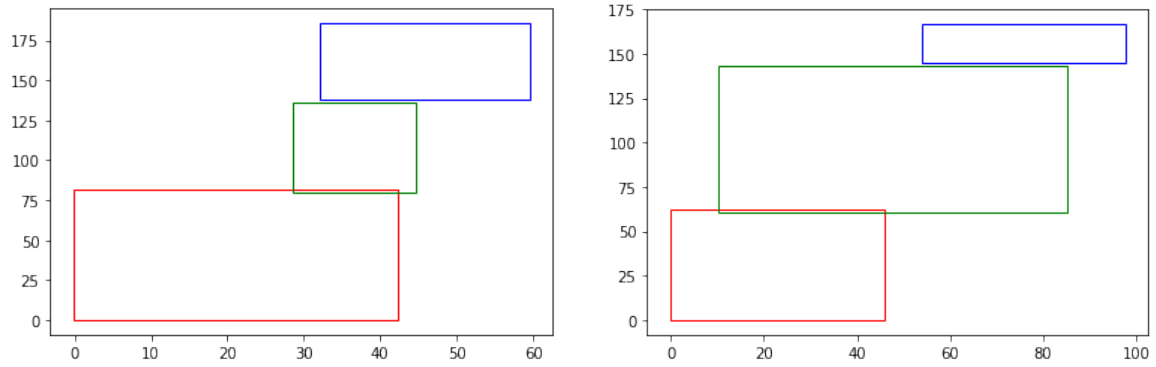


Figure 13: GAN-DD Samples with Negative Constraints I. Samples from a model trained on positive and negative data for constraint I (connection) using our divergence formulation. The model generates reasonable samples and in most cases the connectivity constraints are satisfied. The model can generate blocks that do not intersect and float (up to a small tolerance of 0.9 u). However, because we do not rely on negative 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 positive data.

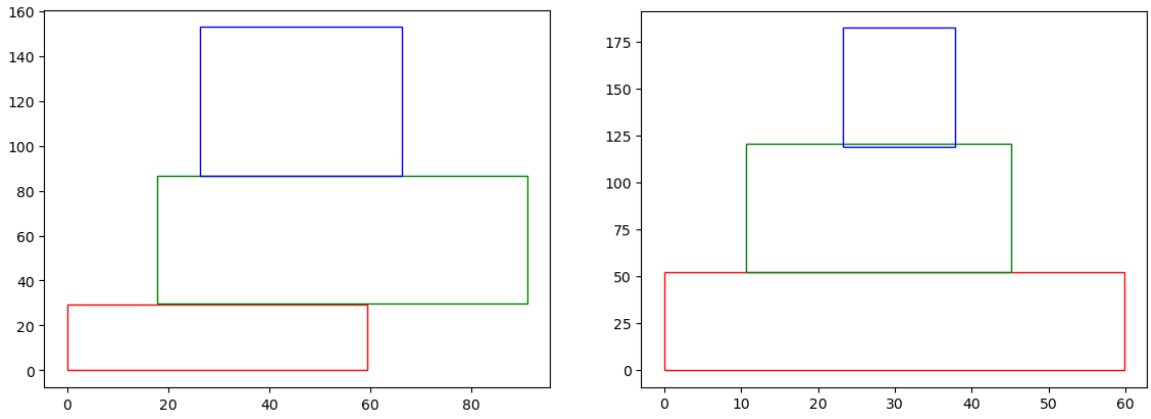


Figure 14: GAN-DD Samples with Negative Constraints I+II. Samples from a model trained on positive and negative data for constraint I (connection) and constraint II (stability) using our divergence formulation. The model generates reasonable samples and in most cases the connectivity constraints are 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 negative 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 positive data. This visualization corroborates the need for negative designs when dealing with constraint satisfaction in generative models.

G Assorted Engineering Problems

G.1 Datasets and Problems

In this section, we present details on the 12 engineering problems and datasets used for benchmarking.

G.1.1 Ashby Chart

Taken from (Jetton et al., 2023), this problem explores physically feasible combinations of material properties, according to known physical materials from an Ashby chart. The constraint function is a combination of an analytical constraint and a lookup from an Ashby chart. Material properties considered are density, yield strength, and Young's modulus. Material classes included are foams, natural materials, polymers, composites, ceramics, and metals. 1k positive samples and 1k negative samples are selected using uniform random sampling.

G.1.2 Bike Frame

The FRAMED dataset (Regenwetter et al., 2022b) is comprised of 4292 in-distribution (positive) human-designed bicycle frame models. FRAMED also contains 3242 constraint-violating (negative) designs, some of which were human-designed and some of which were synthesized by generative models. FRAMED also contains 10095 generative model-synthesized valid designs that are not assumed to be in-distribution and are thus unused in this benchmark. Constraints are comprised of a set of empirical geometric checks and a black-box 3D reconstruction check. Constraints are unified using an all-or-nothing approach. Validity scores on this dataset are only evaluated using empirical checks.

G.1.3 Cantilever Beam

This problem considers the design of a five-component stepped cantilever beam. The thickness and height of each of the five components are the design variables, while the lengths of each component are given (fixed). Taken from (Gandomi & Yang, 2011), this problem has numerous geometric constraints and an overall constraint limiting the total deflection allowed by the design under a simple concentrated load at the tip of the beam. The optimization objective is not utilized. 1k positive samples and 1k negative samples are selected using uniform random sampling.

G.1.4 Car Impact

This problem quantifies the performance of a car design under a side impact scenario based on European Enhanced Vehicle-Safety Committee (EEVC) procedures (Gandomi et al., 2011). The car chassis is represented by 11 design parameters. Several critical deflection, load, and velocity thresholds are specified over several components of a crash dummy, constituting 10 constraints. The optimization objective is not utilized. 1k positive samples and 1k negative samples are selected using uniform random sampling.

G.1.5 Compression Spring

This problem, taken from (Gandomi & Yang, 2011), centers around the design of a helical compression spring parameterized over coil diameter, wire diameter, and number of spring coils. A constraint on free length and a constraint on displacement under a compressive load are specified. The optimization objective is not utilized. 1k positive samples and 1k negative samples are selected using uniform random sampling.

G.1.6 Gearbox

This gearbox (speed-reducer) design problem, taken from (Gandomi & Yang, 2011) features 7 parameters describing key geometric components like shaft diameters, number of teeth on gears, and face width of gears. Nine constraints are given, spanning considerations like bending stress on gear teeth, transverse stress and deflection on shafts, and surface stresses. The optimization objective is not utilized. 1k positive samples and 1k negative samples are selected using uniform random sampling.

G.1.7 Heat Exchanger

This problem, sourced from (Yang & Gandomi, 2012) considers the design of a heat exchanges, involving eight design parameters and six constraints focused on geometric validity. The optimization objective is not utilized. 1k positive samples and 1k negative samples are selected using uniform random sampling.

G.1.8 Pressure Vessel

This cylindrical pressure vessel design problem is taken from (Gandomi & Yang, 2011). The pressure vessel is parametrized according to four parameters, namely the cylinder thickness, spherical head thickness, inner radius, and cylinder length. Four geometric and structural constraints are specified in accordance with

American Society of Mechanical Engineers (ASME) design codes. The optimization objective is not utilized. 1k positive samples and 1k negative samples are selected using uniform random sampling.

G.1.9 Reinforced Concrete

Taken from (Gandomi & Yang, 2011), this problem centers around the design of a simply supported concrete beam under a simple distributed load case. The beam is parameterized using a cross sectional area, base length, and height and is subject to a safety requirement indicated in the American Concrete Institute (ACI) 319-77 code. The optimization objective is not utilized. 1k positive samples and 1k negative samples are selected using uniform random sampling.

G.1.10 Ship Hull

The SHIPD Dataset (Bagazinski & Ahmed, 2023) is comprised of 30k valid (positive) ship hull designs and 20k invalid (negative) ship hull designs. The SHIPD dataset includes numerous constraints spanning geometric rules and functional performance targets, focusing on various types of hydrodynamic performance.

G.1.11 Truss

Taken from (Yang & Gandomi, 2012), this truss design problem considers the design of a three-beam truss parameterized by the length length of two of the beams (symmetry specifies the length of the third). The system is subject to one geometric constraint and two maximum stress constraints. 1k positive samples and 1k negative samples are selected using uniform random sampling.

G.1.12 Welded Beam

Taken from (Gandomi & Yang, 2011), this problem concerns a cantilever beam welded to a flat surface under a simple concentrated load at the tip of the beam. The beam is parametrized using a weld thickness, welded joint length, beam thickness, and beam width. Five structural constraints are given, specifying a maximum shear stress, bending stress, buckling load, and deflection, as well as a geometric constraint on the beam. The optimization objective is not utilized. 1k positive samples and 1k negative samples are selected using uniform random sampling.

G.2 Training and Architecture Details

We train the same 16 models tested on the 2D test problems. Model details are the same as described in D.2. Models trained on "bike frame" and "ship hull" are trained for 2000 epochs (since the datasets are larger) and models trained on the other 10 problems are trained for 5000 epochs.

G.3 Metrics

We measure validity, maximum mean discrepancy (MMD), and F1 score, as discussed in Sec. 5.1 of the main paper.

G.4 Extended Results and Discussion

Included in Tables 14 to 19 are Validity, MMD, and F1 scores for both adversarial and likelihood-based models for the 12 engineering problems. Several key takeaways can be extracted:

- NDGMs achieve significantly better validity scores than vanilla models.
- Likelihood models generally achieve better validity scores, lower MMD scores, and similar F1 scores.
- Vanilla GANs generally achieve better MMD scores than negative data GANs. This trend is not mirrored in likelihood-based models.
- As discussed in the main paper, discriminator overloading (GAN-DO) and multiclass discriminator (GAN-MC) GANs are dominant in validity metric.
- VAE with classifier loss (VAE-CL) is dominant in validity metric.

We stress that these takeaways are highly dataset dependent and should not be taken as general analysis of models from a standpoint of broad applicability. Underscoring this point: The models that perform best in these engineering-related tests are not the same models that perform best on the non-convex 2D test problems nor the block-stacking problem. Many of these engineering problems are fairly convex, lending themselves to different optimal models.

Table 14: Validity scores for adversarial models on engineering problems. **Best** is bolded and next two best are underlined. Lower (↓) is better. Median scores over three runs are reported.

	GAN	GAN-CC	GAN-CL	GAN-Rej	GAN-DO	GAN-D2	GAN-MC	GAN-RM
Ashby Chart	2.35	3.05	**0.87**	4.12	1.28	3.76	3.17	2.37
Bike Frame	4.98	14.28	3.08	6.05	5.11	3.18	**1.09**	2.91
Cantilever Beam	8.22	7.13	6.54	5.42	5.97	6.39	**4.53**	6.23
Compression Spring	2.23	3.73	2.26	1.22	0.31	2.63	**0.18**	2.49
Car Impact	10.43	10.72	7.55	8.67	5.89	8.23	**4.67**	8.26
Gearbox	0.57	1.48	0.19	0.12	**0.01**	0.09	0.09	0.14
Heat Exchanger	8.65	5.98	7.47	10.09	6.23	9.09	**4.86**	8.56
Pressure Vessel	2.84	2.69	0.58	0.73	**0.01**	1.05	0.42	1.45
Reinforced Concrete	0.66	2.25	0.58	0.33	**0.03**	0.49	0.28	0.55
Ship Hull	97.44	96.85	97.37	**86.69**	96.67	98.82	96.51	99.74
Truss	0.04	0.34	0.09	0.06	**0**	**0**	0.73	**0**
Welded Beam	2.5	2.73	2.18	1.85	**0.63**	1.95	0.66	1.63

Table 15: Validity scores for likelihood-based models on engineering problems. **Best** is bolded and next two best are underlined. Lower (↓) is better. Median scores over three runs are reported.

	VAE	VAE-CC	VAE-CL	VAE-Rej	DDPM	DDPM-CL	DDPM-G	DDPM-Rej
Ashby Chart	0.45	NA	**0.33**	0.51	1.77	1.66	13.93	1.89
Bike Frame	0.88	NA	0.73	**0.63**	0.86	1.52	2.46	0.83
Cantilever Beam	1.02	1.1	**0.54**	1.1	1.3	1.14	2.07	1.56
Compression Spring	0.34	0.39	**0.14**	0.22	1.28	11.32	3.07	1.32
Car Impact	1.42	1.35	**0.39**	1.09	1.86	2.58	2.72	1.94
Gearbox	0.15	0.02	0.01	**0**	0.08	0.02	0.12	0.01
Heat Exchanger	1.24	1.71	**1.15**	1.41	1.95	2.7	3.53	2.2
Pressure Vessel	0.26	0.16	**0.05**	0.06	0.7	0.67	0.95	0.52
Reinforced Concrete	0.12	0.15	**0.01**	0.08	0.37	0.36	0.5	0.42
Ship Hull	**0**	31.76	**0**	**0**	85.37	84.76	91.2	84.76
Truss	0.08	**0.01**	**0.01**	**0.01**	0.3	0.19	0.58	0.13
Welded Beam	0.41	0.28	**0.08**	0.16	1.01	1.25	1.51	0.63

Table 16: MMD scores for adversarial models on engineering problems. **Best** is bolded and next two best are underlined. Lower (↓) is better. Median scores over three runs are reported.

	GAN	GAN-CC	GAN-CL	GAN-Rej	GAN-DO	GAN-D2	GAN-MC	GAN-RM
Ashby Chart	**1.255**	1.996	2.212	1.304	1.516	1.607	1.441	1.309
Bike Frame	**14.88**	98.26	20.70	16.40	29.07	15.52	55.95	18.73
Cantilever Beam	2.642	**2.511**	2.898	2.948	3.170	2.639	3.402	2.596
Compression Spring	1.446	1.673	1.198	1.480	**1.133**	1.497	1.529	1.567
Car Impact	3.165	**2.556**	2.716	3.395	2.991	2.612	3.440	2.674
Gearbox	2.005	3.037	2.301	**1.895**	4.741	2.318	3.663	2.570
Heat Exchanger	2.411	3.479	2.174	**2.102**	3.295	2.269	4.860	2.359
Pressure Vessel	**1.714**	1.891	2.363	1.724	6.658	1.947	3.019	1.719
Reinforced Concrete	**1.182**	1.615	1.525	1.456	3.584	1.624	1.795	1.946
Ship Hull	5.320	19.76	6.542	**4.506**	8.588	8.998	6.029	12.47
Truss	1.026	1.506	**1.006**	1.310	1.575	3.375	8.552	4.739
Welded Beam	1.675	2.306	1.632	**1.384**	4.575	1.799	3.687	2.651

Table 17: MMD scores for likelihood-based models on engineering problems. **Best** is bolded and next two best are underlined. Lower (↓) is better. Median scores over three runs are reported.

	VAE	VAE-CC	VAE-CL	VAE-Rej	DDPM	DDPM-CL	DDPM-G	DDPM-Rej
Ashby Chart	3.240	NA	3.343	**3.018**	10.86	10.45	8.643	10.25
Bike Frame	42.76	NA	53.74	49.18	**2.479**	**2.494**	**2.499**	**2.487**
Cantilever Beam	5.870	5.631	5.858	5.583	4.401	4.497	**3.809**	4.288
Compression Spring	3.203	**2.725**	3.079	2.810	20.46	21.41	20.28	20.78
Car Impact	4.756	4.951	5.145	4.565	3.943	3.956	**3.334**	3.845
Gearbox	5.450	5.468	6.055	5.771	**3.276**	3.551	4.333	3.499
Heat Exchanger	6.962	6.389	7.075	6.729	6.677	6.787	**4.500**	6.835
Pressure Vessel	3.931	4.056	4.223	**3.897**	6.593	7.107	8.043	7.165
Reinforced Concrete	**3.095**	3.673	3.786	3.473	10.93	10.46	12.66	10.88
Ship Hull	1001	32.66	1001	1001	**2.135**	**2.133**	**2.132**	**2.134**
Truss	1.604	1.610	**1.416**	1.502	9.985	9.535	30.25	10.11
Welded Beam	4.539	**4.011**	4.577	4.261	7.722	8.442	9.185	8.456

Table 18: F1 scores for adversarial models on engineering problems. **Best** is bolded and next two best are underlined. Higher (↑) is better. Median scores over three runs are reported.

	GAN	GAN-CC	GAN-CL	GAN-Rej	GAN-DO	GAN-D2	GAN-MC	GAN-RM
Ashby Chart	**0.967**	0.947	0.951	0.963	0.963	0.960	0.960	0.964
Bike Frame	0.684	0.214	0.663	0.675	0.253	0.666	0.506	**0.692**
Cantilever Beam	**0.940**	0.938	0.930	0.934	0.914	0.924	0.914	0.937
Compression Spring	0.953	0.956	0.957	0.954	0.958	**0.959**	0.955	0.949
Car Impact	0.927	0.930	0.931	0.922	0.934	**0.936**	0.897	0.928
Gearbox	0.945	0.930	0.941	**0.954**	0.903	0.943	0.939	0.947
Heat Exchanger	0.942	0.929	0.948	**0.954**	0.929	0.948	0.894	0.942
Pressure Vessel	0.957	0.942	0.943	**0.962**	0.904	0.955	0.944	0.946
Reinforced Concrete	**0.964**	0.952	0.956	0.960	0.932	0.959	0.951	0.956
Ship Hull	0.043	0.012	0.020	**0.054**	0.044	0.026	0.053	0.019
Truss	**0.958**	0.946	0.954	0.954	0.937	0.888	0.869	0.891
Welded Beam	0.958	0.937	0.967	**0.968**	0.921	0.954	0.927	0.939

Table 19: F1 scores for likelihood-based models on engineering problems. **Best** is bolded and next two best are underlined. Higher (↑) is better. Median scores over three runs are reported.

	VAE	VAE-CC	VAE-CL	VAE-Rej	DDPM	DDPM-CL	DDPM-G	DDPM-Rej
Ashby Chart	**0.953**	NA	0.953	0.950	0.859	0.855	0.876	0.856
Bike Frame	**0.899**	NA	0.897	0.890	0.780	0.778	0.786	0.754
Cantilever Beam	**0.966**	0.958	0.958	0.959	0.935	0.939	0.929	0.928
Compression Spring	0.946	**0.957**	0.954	0.956	0.795	0.785	0.795	0.794
Car Impact	0.955	**0.960**	0.947	0.958	0.938	0.939	0.927	0.938
Gearbox	0.958	**0.965**	0.963	0.963	**0.969**	0.968	0.863	0.965
Heat Exchanger	**0.965**	0.952	0.962	0.963	0.899	0.902	0.871	0.879
Pressure Vessel	0.956	**0.960**	0.954	0.959	0.887	0.884	0.872	0.884
Reinforced Concrete	0.946	0.945	**0.953**	0.940	0.846	0.851	0.837	0.848
Ship Hull	0.033	**0.906**	0.033	0.034	0.879	0.887	0.871	0.873
Truss	0.952	0.959	**0.961**	0.948	0.876	0.880	0.793	0.874
Welded Beam	0.953	0.956	0.954	**0.956**	0.875	0.866	0.867	0.870

H Details on Topology Optimization Experiments

In this appendix, we include extra details on the datasets, models, training, and results of the topology optimization (TO) experiments.

H.1 Dataset Details

The GAN was trained exclusively on 32436 valid (connected) topologies generated through iterative optimization (SIMP) (Bendsøe & Kikuchi, 1988). The GAN-MC variants are trained on a medley of disconnected topologies generated by iterative optimization (2564), and either procedurally-generated synthetic topologies (35000) or GAN-generated disconnected topologies (92307). Synthetic topologies were sourced directly from the classification dataset of (Mazé & Ahmed, 2023). The GAN used to generate disconnected topologies for rejection was the exact GAN benchmarked in the paper. Topologies were checked for continuity and rejected samples were added to the negative dataset. All positive and negative data was multiplied by 8 using simple data augmentation consisting of rotation and flips before training any model. The various data sources are visualized below

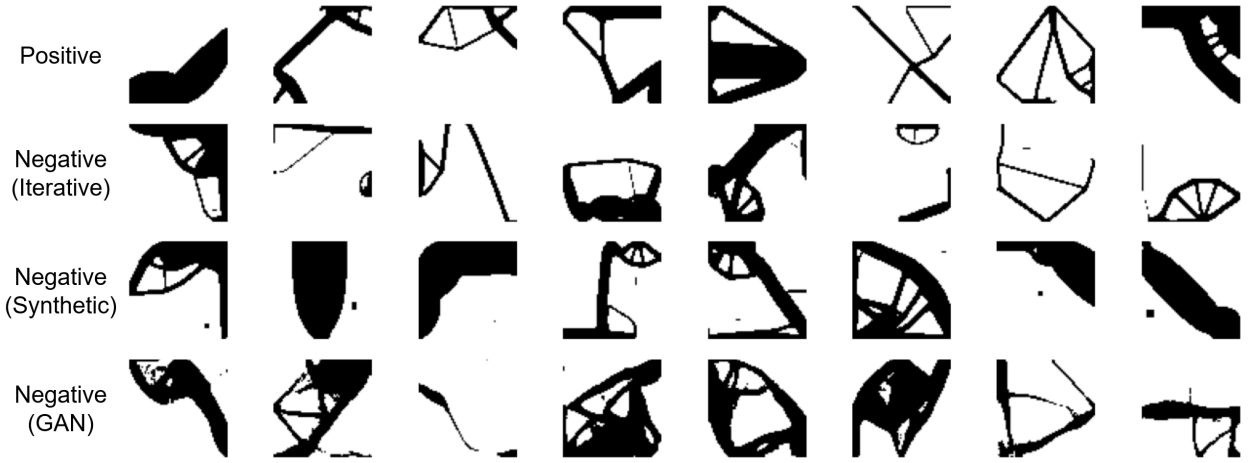


Figure 15: Visualization of positive data and various sources of negative data used to train GAN and GAN-MC on TO problems.

H.2 Model Details

The model architectures of the GAN and GAN-MC are identical except for the final output dimension of the descriminator. Both generator and descriminator are simple 5-layer convolutional neural networks. The generator has 3.6M parameters, while the discriminator has 2.8M parameters. For more architectural details, we refer the reader to the codebase. The latent dimension is 100, batch size is 128, and learning rate for both models is 2e-4, using the adam optimizer.

H.3 Visualization

We visualize several samples generated by GAN and GAN-MC, annotating constraint violations. Note that several violations are circled in some topologies. Each floating section contributes to the pixel count invalidity score, but is not double-counted for binary invalidity score. The topologies generated by GAN-MC have visibly fewer invalidities compared to the topologies generated by the GAN.

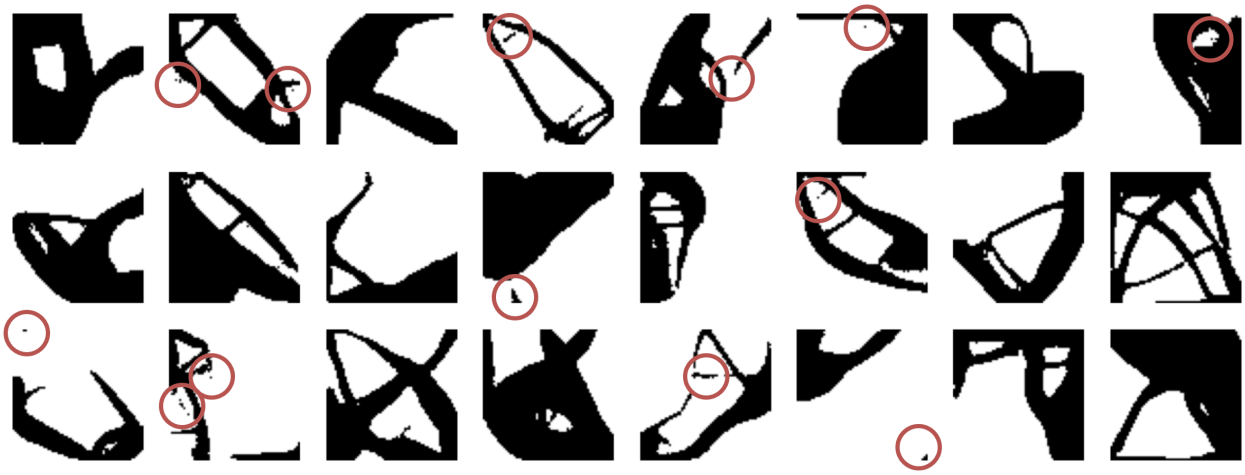


Figure 16: Topologies generated by GAN with constraint violations annotated.

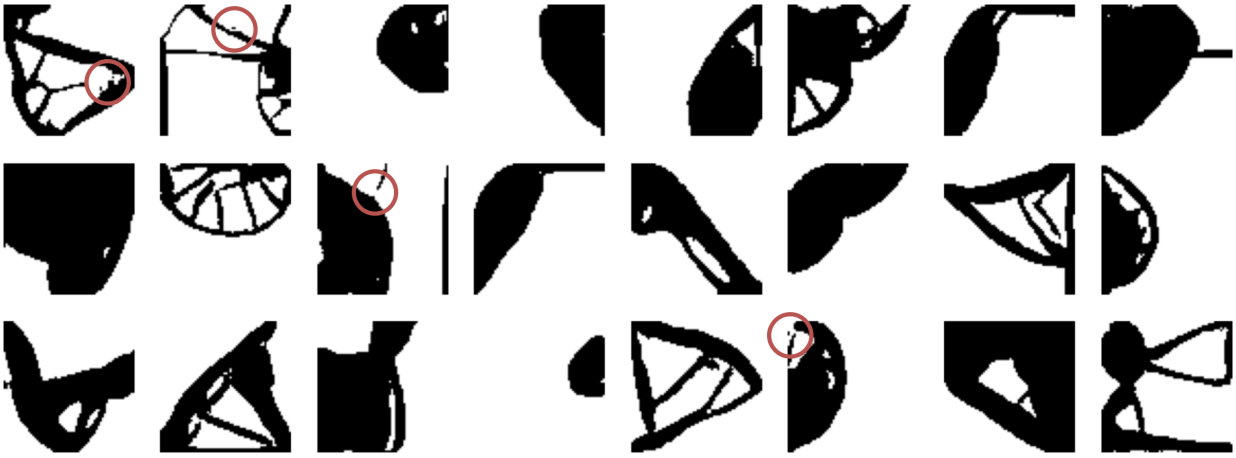


Figure 17: Topologies generated by GAN-MC trained on rejected negative data with constraint violations annotated.