PaDGAN: Learning to Generate High-Quality Novel Designs

Wei Chen[1]
Siemens Corporate Technology,
Princeton, NJ 08540
e-mail: chen.wei@siemens.com

Faez Ahmed
Department of Mechanical Engineering,
Northwestern University,
Evanston, IL 10601
e-mail: faez@northwestern.edu

Deep generative models are proven to be a useful tool for automatic design synthesis and design space exploration. When applied in engineering design, existing generative models face three challenges: (1) generated designs lack diversity and do not cover all areas of the design space, (2) it is difficult to explicitly improve the overall performance or quality of generated designs, and (3) existing models generally do not generate novel designs, outside the domain of the training data. In this article, we simultaneously address these challenges by proposing a new determinantal point process-based loss function for probabilistic modeling of diversity and quality. With this new loss function, we develop a variant of the generative adversarial network, named "performance augmented diverse generative adversarial network" (PaDGAN), which can generate novel high-quality designs with good coverage of the design space. By using three synthetic examples and one real-world airfoil design example, we demonstrate that PaDGAN can generate diverse and high-quality designs. In comparison to a vanilla generative adversarial network, on average, it generates samples with a 28% higher mean quality score with larger diversity and without the mode collapse issue. Unlike typical generative models that usually generate new designs by interpolating within the boundary of training data, we show that PaDGAN expands the design space boundary outside the training data towards high-quality regions. The proposed method is broadly applicable to many tasks including design space exploration, design optimization, and creative solution recommendation. [DOI: 10.1115/1.4048626]

Keywords: design automation, design optimization, design representation, generative design, deep learning, simulation-based design

1 Introduction

A designer wants good design solutions, which are creative and meet the performance requirements. The term design here refers to any man-made components that serve certain functionality and can be represented by a set of parameters (i.e., design variables). Examples range from chairs to turbine blades. By manually and iteratively exploring design ideas using experience and heuristics, the designers take the risks of (1) wasting time on evaluating unfavorable or even invalid design candidates and (2) not having sufficient width/depth for exploration/exploitation. An ideal design space exploration tool should ensure that, with low cost, one can exploit high-performance solutions in a design space and explore all feasible alternatives.

Design synthesis is the area of research that focuses on developing guidelines, methods, and tools for supporting creation of designs [1]. While recent advances in machine learning assisted automatic design synthesis and design space exploration are promising, the current methods are still far from this ideal picture. To model a design space, researchers have used deep generative models like variational autoencoders (VAEs) [2] and generative adversarial networks (GANs) [3], as they can learn the distribution of existing designs. The hope is that by learning an underlying *latent space*, one can automatically synthesize new designs from the low-dimensional *latent vectors* and will make design exploration more efficient due to the reduced dimensionality [4–7]. However, unlike image generation tasks where these generative models are commonly applied, engineering design problems have one or more performance (or quality) measures. The quality measures how well a design achieves its intended goals and is defined based on the specific problem. For example, beam design problems often define quality based on the compliance value (single objective) [8] or

both compliance and natural frequency (multi-objective) [9]. For aerodynamic design, quality can be defined as the lift-to-drag ratio [6] or the inverse of the drag coefficient [10]. Current state-of-the-art generative models have no mechanism of explicitly promoting high-quality design generation. One may spend huge effort to train a generative model only to find that many generated designs are infeasible or do not meet design requirements. One way of working around this problem is to exclude low-quality data while training [10]. However, such an approach may affect model performance due to the reduced training sample size. This creates a need to explicitly embed the quality measurement into a generative model, so that it can learn to generate high-quality designs by making use of full data and their quality measurements.

In this work, we focus on addressing the problem of simultaneously maximizing diversity and quality of generated designs. Specifically, we develop a new loss function, based on determinantal point processes (DPPs) [11], for generative models to encourage both high-quality and diverse design synthesis. By using this loss function, we develop a new variant of GAN, named performance augmented diverse generative adversarial network (PaDGAN). We show that it can generate high-quality new samples with a good coverage of the design space. More importantly, we found that PaDGAN can expand the existing boundary of the design space toward high-quality regions, which indicates its ability of generating novel high-quality designs.

With the ability of generating high-quality and diverse designs from a (reduced) latent representation, the proposed PaDGAN can then be used for improving the efficiency in design space exploration. While it is interesting to see how exploring the low-dimensional latent space of the PaDGAN can accelerate exploration or improve the performance of the optimal solution, we leave that to the future work. In this article, we focus on the architecture of PaDGAN and its performance in design synthesis.

2 Background and Related Work

Our work produces generative models that synthesize diverse designs from latent representations. There are primarily two streams of related research: (1) design synthesis and (2) diversity measurement. Within these two fields, we provide a brief background on two techniques we use in this paper—GANs and DPPs—and their applications in design. Readers interested in a more comprehensive understanding of their background are advised to read Kulesza and Taskar's work [11] for DPPs and the chapter on "Deep Generative Models" in Ref. [12].

2.1 Deep Generative Model-Based Design Synthesis.
To achieve automatic design synthesis, past researchers have used approaches based on shape grammar [13–15], graph enumeration [16,17], functional models [18], analogy [1], and constraint programming [19,20]. These methods often need to encode expert knowledge as either grammar rules, functional basis, or constraints. In recent years, data-driven design synthesis has become increasingly popular. Different from traditional design synthesis methods, data-driven methods do not necessarily require expert knowledge and can learn to generate plausible new designs from a database [4,6,21–23].

In the last few years, deep generative models have gained traction due to their ability to learn complex feature representations. The family of deep generative models contains various methods, among which VAEs and GANs are the two most commonly used deep generative models for solving engineering design problems. For example, they have been used in applications like design exploration [4,5,24], surrogate modeling [25], and material microstructure design [26,27].

Applications of Deep Generative Models in Design Synthesis. Many design applications have huge collections of unstructured design data (computer-aided design (CAD) models, images, microstructures, etc.) with hundreds of features and multiple functionalities. To learn from these complex datasets, deep generative models have increasingly been employed. For instance, Chen et al. [6,28] proposed a BézierGAN model for airfoil parameterization and synthesis and demonstrated significantly faster convergence to the optimum when optimizing over the latent space. Yang et al. [27] used a GAN to generate microstructures and performed design optimization over the latent space. Chen and Fuge [5] proposed a hierarchical GAN architecture to synthesize designs with inter-part dependencies. Oh et al. [29] integrated topology optimization and generative models to generate designs that are optimized for engineering performance. These methods either do not explicitly consider the quality of generated designs or use a separate optimization process to search for high-quality designs. Burnap et al. [30] used a VAE to generate new highly rated automotive images, which are aesthetically pleasing. Shu et al. [10] proposed a GAN-based model to generate high-quality 3D designs, where they improve the quality of generated samples by retraining the model on an updated dataset with low performing designs removed. In contrast, our method improves the quality of generated designs while training the deep generative model, without retraining or discarding any samples in the training data. Also, to the best of our knowledge, there is no generative model that simultaneously encourages diversity and quality. While the methods we develop in this work are applicable to most deep generative models, we use GANs to demonstrate our results and will describe them next.

Generative Adversarial Networks. GANs [3] model a game between a generative model (*generator*) and a discriminative model (*discriminator*). The generative model maps an arbitrary noise distribution to the data distribution (i.e., the distribution of designs in our scenario) and thus can generate new data, while the discriminative model tries to perform classification, i.e., to distinguish between real and generated data. The generator G and the discriminator D are usually built with deep neural networks. As D improves its classification ability, G also improves its ability to generate data that fools D. Thus, a vanilla GAN (standard GAN with no bells and whistles) has the following objective function, which comprises a discriminator loss term and a generator loss term:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim P_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim P_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]$$

(1)

where $\mathbf{x}$ is sampled from the data distribution P_{data}, $\mathbf{z}$ is sampled from the noise distribution $P_{\mathbf{z}}$, and $G(\mathbf{z})$ is the generator distribution. A trained generator thus can map from a predefined noise distribution to the distribution of designs. The noise input $\mathbf{z}$ is considered as the latent representation of the data, which can be used for design synthesis and exploration.

Problems in Using Generative Adversarial Networks for Design Synthesis. Learning in GANs can be difficult in practice, which may be one of the reasons that they are less widely used in design compared to VAEs. Despite an enormous amount of recent work in the machine learning community, GANs are notoriously unstable to train, and it has been observed that they often suffer from *mode collapse* [31], in which the generator network learns how to generate samples from a few modes of the data distribution but misses many other modes. For instance, when training on multiple categories of designs, a GAN model would sometimes generate designs only for a single category [32].

Recent approaches [33–35] tackled mode collapse in one of two different ways: (1) modifying the learning of the system to reach a better convergence point or (2) explicitly enforcing the models to capture diverse modes or map back to the true-data distribution. Solutions to the mode collapse problem range from designing a reconstructor network in VEEGAN [34] to matching the similarity matrix of generated samples with data [36]. However, these approaches do not directly optimize diversity. Their objective, which is often improving data fit along with training stability, indirectly promotes diversity as a byproduct, which is not necessarily guaranteed. In contrast, PaDGAN explicitly enforces diversity in generated samples, where we embed the diversity measure in the loss function. This allows direct control on generated samples' diversity and avoids other adjustments—e.g., adding any extra trainable parameters or changing the learning paradigm [36]. This is desirable for problems where the focus is generating diverse samples and not just capturing all the modes of the data. It also addresses the mode collapse problem by virtue of promoting generation of diverse solutions, which encourages samples to cover different modes. It is important to note that promoting diversity will always ensure that all modes are captured, while the reverse is not true. We later discuss how our method contrasts with the state-of-art approach of explicitly capturing diversity.

2.2 Measuring Design Coverage.
Massive highly redundant sources of audio, video, speech, text documents, and sensor data have become commonplace and are expected to become larger and more preponderant in the future [37]. This brings a need to measure diversity of a set of items, such that redundancy in data can be reduced and machine learning models can be trained using data with a smaller sample size and which are not biased in favor of a few classes. *Diversity* (also called coverage or variety) is a measure of how different a set of items are from each other. Quantitatively, it is measured using two predominant ways—submodular functions or DPPs. Submodular functions are set functions with diminishing marginal gain property, which naturally model notions of coverage and diversity. They achieved among the top results on common automatic document summarization benchmarks (e.g., at the Document Understanding Conference [38]). In design, too, researchers have used submodular function-based diversity measures to understand design space exploration using

terms like *variety* [39–41]. These functions have helped designers sift through large sets of ideas by ranking them [42] or selecting a diverse subset [43]. Ahmed et al. [42] compared DPPs [11] with certain commonly used submodular functions. They concluded that unlike submodular functions, DPPs are more flexible, since they only need a valid similarity kernel as an input rather than an underlying Euclidean space or clusters. In this article, we will use DPPs as a measure of diversity, which is described next.

Determinantal Point Processes. DPPs, which arise in quantum physics, are probabilistic models that model the likelihood of selecting a subset of diverse items as the determinant of a kernel matrix. Viewed as joint distributions over the binary variables corresponding to item selection, DPPs essentially capture negative correlations and provide a way to elegantly model the trade-off between often competing notions of quality and diversity. The intuition behind DPPs is that the determinant of a kernel matrix roughly corresponds to the volume spanned by the vectors representing the items. Points that "cover" the space well should capture a larger volume of the overall space and thus have a higher probability. As shown by Kulesza and Taskar [44], one of DPPs' advantages is that computing marginals, computing certain conditional probabilities, and sampling can all be done in polynomial time. In this article, we focus on another advantage of DPPs, which is the decomposition of DPP kernels into quality and similarity terms.

For the purposes of modeling real data, the most relevant construction of DPPs is through L-ensembles [45]. An L-ensemble defines a DPP via a positive semidefinite matrix L indexed by the elements of a subset S. The kernel matrix L defines a global measure of similarity between pairs of items, so that more similar items are less likely to co-occur. The probability of a set S occurring under a DPP is calculated as follows:

$$\mathbb{P}_L(S) = \frac{\det(L_S)}{\det(L + I)} \tag{2}$$

where $L_S \equiv [L_{ij}]_{ij \in S}$ denotes the restriction of L to the entries indexed by elements of S, I is an $N \times N$ identity matrix, and N is the total number of items. For any set size, the most probable subset under a DPP will have the maximum likelihood over $\mathbb{P}_L(S)$ or (equivalently) the highest determinant (the denominator can be ignored for maximizing the determinant of a fixed set size). Similar to submodular functions, one of the main applications of DPP is extractive document summarization, where it provided state-of-the-art results.

In Sec. 3, we show how the decomposition of DPP kernels can be used to design a DPP-based loss function, which promotes the quality and the diversity of generated samples in a generative model.

2.3 Comparison With State-of-the-Art and Our Contributions. The work closest to ours is the generative determinantal point processes (GDPP) method by Elfeki et al. [36]. The authors devised an objective term that encourages the GAN to synthesize data with diversity similar to the training data. PaDGAN differs from their method in three aspects. First, PaDGAN is stable against scaling of data; while on validating GDPP for multiple test problems, we found that their method does not work for problems with training data at different scales. Second, while PaDGAN aims to maximize the diversity of generated samples, GDPP aims to achieve a similar diversity value as the training data. By avoiding the goal of mimicking the diversity of the training data, PaDGAN will generate diverse samples even when the original training dataset is biased in favor of a few modes, while GDPP is designed to mimic the bias in generated samples. Finally, we maximize the quality of generated samples, whereas GDPP does not have such consideration. This feature of PaDGAN is helpful for design exploration as it can help discover novel high-quality designs (demonstrated in Sec. 5.2).

The scientific contributions and novelty of this work are as follows:

(1) We propose a novel design synthesis method that simultaneously encourage synthesis of diverse and high-performance designs.
(2) We find that PaDGAN can expand the design space boundary toward high-quality regions that it had not seen from the existing data.
(3) We propose a way to control the trade-off between quality and diversity in DPPs. Our method extends past work on decomposing a DPP kernel by providing a way to tune the relative importance of quality over diversity.
(4) We provide easy-to-verify test cases and metrics to validate any generative models, whose goal is to maximize sample quality and/or coverage over a dataset with multiple modes.
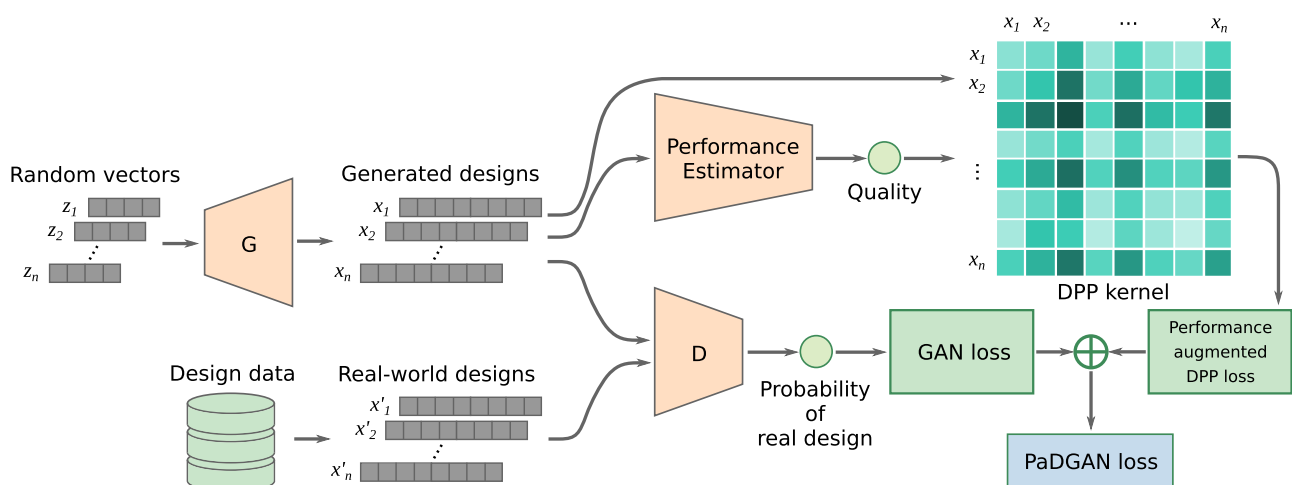

Fig. 1 Architecture of PaDGAN. Built upon a vanilla GAN, the generator *G* generates designs given random vectors (drawn from a prior distribution) as inputs; while the discriminator *D* gets designs as inputs and predicts whether they are generated by *G* or sampled from the dataset. Different from the vanilla GAN, PaDGAN considers the diversity and the quality of generated designs by adding a performance augmented DPP loss. This is done by constructing a DPP kernel that accounts for both diversity and quality of designs. The quality is computed or predicted by a performance estimator.

3 Methodology

Built on a standard GAN architecture, PaDGAN introduces a *performance augmented DPP loss,* which measures the diversity and the quality of a batch of generated designs during training. The overall model architecture of PaDGAN is shown in Fig. 1. In this section, we begin by describing how to decompose a DPP kernel, then proceed on how to create a DPP loss, which augments high performing designs, and finally provide a method to balance the diversity and quality using a quality dial. We also add a note on improving training stability at the end.

3.1 Decomposition of a Determinantal Point Process Kernel.
DPP kernels can be decomposed into quality and diversity parts [11]. The (i, j)th entry of a positive semidefinite DPP kernel L can be expressed as follows:

$$L_{ij} = q_i \, \phi_i^T \, \phi_j \, q_j \tag{3}$$

We can think of $q_i \in \mathbb{R}^+$ as a scalar value measuring the quality (or performance) of an item i and $\phi_i^T \phi_j$ as a signed measure of similarity between items i and j. The decomposition enforces L to be positive semidefinite. Suppose we select a subset S of samples, then this decomposition allows us to write the probability of this subset S as the square of the volume spanned by $q_i \phi_i$ for $i \in S$ using the following equation:

$$\mathbb{P}_L(S) \propto \prod_{i \in S} (q_i{}^2) \det (K_S) \tag{4}$$

where K_S is the similarity matrix of S.

The first term increases with the quality of the selected items, and the second term increases with the diversity of the selected items. As item i's quality q_i increases, so do the probabilities of sets containing item i. As two items i and j become more similar, $\phi_i^T \phi_j$ increases and the probabilities of sets containing both i and j decrease. From a geometric intuition, the determinant of L_Y is equal to the squared volume of the parallelepiped spanned by the vectors $q_i \phi_i$ for $i \in Y$. We show an illustration of this intuition in Fig. 2. The magnitude of the vector representing item i is q_i, and its direction is ϕ_i. It shows how DPPs decomposed into quality and diversity naturally balance the two objectives of high quality and high diversity.

When selecting a subset S of items, without the diversity term, we would choose high-quality items, but we would tend to choose similar high-quality items over and over. Without the quality term, we would get a very diverse set, but we might fail to include the most important items in S, focusing instead on low-quality outliers. By combining the two models, we can achieve a more balanced result. The key intuition of PaDGAN is that if we can find a way to add the term from Eq. (4) to the objective function of any generative model, then while training it will be encouraged to generate high probability subsets, which will be both diverse and high quality. In Sec. 3.2, we define such a loss function.
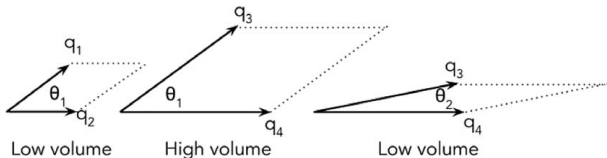

Fig. 2 Visualization of volume in 2D space, where q_i represent the quality of an item and θ_i shows how similar they are. Comparing the leftmost figure to the central figure, we observe that the similarity is same, but the quality magnitude increases, leading to a higher volume. From the central figure to the rightmost figure, we observe that the quality magnitude is the same, but the similarity increases, leading to a lower volume (or diversity) encompassed by the two vectors.

While the authors used this decomposition to find quality and similarity terms from a known kernel, we reverse this procedure to create the kernel L for a sample of points generated by PaDGAN from known inter-sample similarity values and quality. Note that in a DPP model, the quality or performance of an item is a scalar value, like compliance, displacement, drag-coefficient, and so on. The quality can be estimated using an external model (like a physics-based simulator) or by finding the distance of current performance of a design from a target performance. For multidimensional cases, the quality can be derived by taking the norm or weighted sum of multiple dimensions. The similarity terms $\phi(i)^T \phi(j)$ can be derived using any similarity kernel, which we represent using $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(i)^T \phi(j)$ and $\|\phi(i)\| = \|\phi(j)\| = 1$. Here, $\mathbf{x}_i$ is a vector representation of a design.

3.2 Performance Augmented Determinantal Point Processes Loss.
Our performance augmented DPP loss models diversity and quality simultaneously and gives a lower loss to sets of designs, which are both high quality and diverse. Specifically, we construct a kernel matrix L_B for a generated batch B based on Eq. (3). For each entry of L_B, we have

$$L_B(i, j) = k(\mathbf{x}_i, \mathbf{x}_j)\big(q(\mathbf{x}_i)q(\mathbf{x}_j)\big)^{\gamma_0} \tag{5}$$

where $\mathbf{x}_i, \mathbf{x}_j \in B$, $q(\mathbf{x})$ is the quality value at $\mathbf{x}$ and $k(\mathbf{x}_i, \mathbf{x}_j)$ is the similarity kernel between $\mathbf{x}_i$ and $\mathbf{x}_j$. We add γ_0 term as a dial to control the weight of quality, which is further explained in Sec. 3.3.

The performance augmented DPP loss is expressed as follows:

$$\mathcal{L}_{\text{PaD}}(G) = -\frac{1}{|B|} \log \det (L_B) = -\frac{1}{|B|} \sum_{i=1}^{|B|} \log \lambda_i \tag{6}$$

where λ_i is the ith eigenvalue of L_B. Note that computing Eq. (6) can be expensive when the size of B is large as the complexity of calculating the determinant is $O(n^3)$. However, as we can train the model with small mini-batches, the computational cost of computing Eq. (6) is small. Also note here we only optimize the generator G as the purpose of $\mathcal{L}_{\text{PaD}}$ is to promote high diversity and quality for generated designs, which is independent of the discriminator's objective. By adding this loss to the vanilla GAN's objective from Eq. (1), the problem becomes

$$\min_G \max_D V(D, G) + \gamma_1 \mathcal{L}_{\text{PaD}}(G) \tag{7}$$

where γ_1 controls the weight of $\mathcal{L}_{\text{PaD}}(G)$. To update any weight θ_G^i in the generator in terms of $\mathcal{L}_{\text{PaD}}(G)$, we descend its gradient based on the chain rule:

$$\frac{\partial \mathcal{L}_{\text{PaD}}(G)}{\partial \theta_G^i} = \sum_{j=1}^{|B|} \left(\frac{\partial \mathcal{L}_{\text{PaD}}(G)}{\partial q(\mathbf{x}_j)} \frac{dq(\mathbf{x}_j)}{d\mathbf{x}_j} + \frac{\partial \mathcal{L}_{\text{PaD}}(G)}{\partial \mathbf{x}_j} \right) \frac{\partial \mathbf{x}_j}{\partial \theta_G^i} \tag{8}$$

where $\mathbf{x}_j = G(\mathbf{z}_j)$.

Equation (8) indicates a need for $dq(\mathbf{x})/d\mathbf{x}$, which is the gradient of the quality function. In practice, this gradient is accessible when the quality is evaluated through any performance estimator that is differentiable, like adjoint-based solver methods. If the gradient of a performance estimator is not available, one can either use numerical differentiation or approximate the quality function using a differentiable surrogate model (e.g., a neural network-based surrogate model). In our experiments in Sec. 5.2, we use a neural network-based surrogate model. We will explore the possibility of using an automatic differentiation enabled simulator (e.g., an adjoint solver) as the performance estimator in future studies.

3.3 Introducing a Quality Dial for Determinantal Point Process Kernels.
Note that we modified the original objective to introduce γ_0 as a parameter. We found that traditional DPP decomposition does not allow us to change the importance of quality versus diversity within a given kernel. This means that if we fix

the quality scores and similarity scores, the trade-off between the two cannot be controlled. A naive way to increase the importance of quality would be to multiply the quality scores by a large constant and expect it to increase its importance relative to diversity. However, with careful observation, one would realize that this approach would not work. By using the geometric interpretation of the DPPs, this would be equivalent to scaling all lengths by the same factor, which will not affect the relative value of volumes. As quality and diversity objectives are multiplied together to get the probability of the set (Eq. (4)), to change the relative importance, we need to adjust the dynamic range of the quality scores. We do this by using an exponent to change the distribution of the quality. When $\gamma_0 = 0$, all quality scores collapse to one and the resultant PaDGAN model only generates diverse designs. In contrast, for large values of γ_0, the highest quality scores have the largest probability mass and PaDGAN only generates the highest quality designs, ignoring diversity. This method of balancing diversity and quality provides more flexibility to PaDGAN and in general can be used for many applications of DPPs.

3.4 Improving PaDGAN Stability. Stabilization of GAN learning remains an open problem, and in this section, we provide a heuristic method to improve GAN stability, when using a data-driven surrogate model for evaluating the quality. Note that in Eq. (8), the quality gradient is used in the back propagation step. If the quality gradients are not accurate, the generator learning can go astray. This is not a problem when the quality estimator is a simulator that can reasonably evaluate (even with low fidelity) any design in the design space, irrespective of the designs being invalid or unrealistic. However, it creates problems when we use a data-driven surrogate model. A data-driven surrogate model is normally trained only on realistic designs and hence may perform unreliably on unrealistic ones. In the initial stages of training, a GAN model will not always generate realistic designs during training. This makes it difficult for the surrogate model to correctly guide the generator's update and may cause stability issues. To avoid this problem, we propose two small modifications to PaDGAN:

(1) *Realisticity weighted quality.* Specifically, we weight the predicted quality at $\mathbf{x}$ by the probability of $\mathbf{x}$ being the real design (predicted by the discriminator):

$$q(\mathbf{x}) = D(\mathbf{x})q'(\mathbf{x})$$

where $q'(\mathbf{x})$ is the predicted quality (by a surrogate model for example) and $D(\mathbf{x})$ is the discriminator's output at $\mathbf{x}$.

(2) An *escalating schedule* for setting γ_1 (the weight of the performance augmented DPP loss). A GAN is more likely to generate unrealistic designs in its early stage of training. Thus, we initialize γ_1 at 0 and increase it during training, so that PaDGAN focuses on learning to generate realistic designs at the early stage and takes quality into consideration later when the generator can produce more realistic designs. The schedule is set as follows:

$$\gamma_1 = \gamma_1'\left(\frac{t}{T}\right)^p$$

where γ_1' is the value of γ_1 at the end of training, t is the current training step, T is the total number of training steps, and p is a factor controlling the steepness of the escalation.

We can also consider the uncertainty of the quality estimation and put a lower weight on the quality score when the uncertainty is high. However, we only consider the aforementioned two modifications in this article and leave others to future work. Note that these modifications are only needed if one is using a performance estimator (e.g., a surrogate model), which gives unreliable quality predictions for unrealistic designs.

4 Experiment

So far, we have shown how the mathematical components of PaDGAN will encourage it to generate high-quality and diverse samples. In this section, we will describe experiments, which can help us validate our claims. These experiments are carefully designed such that the outcome of any generative models can be verified easily. This section introduces the experimental settings for each example. To show the merit of modeling quality and diversity simultaneously, we compare the PaDGAN with alternative models where those two attributes are modeled separately. In the following sections, we show that for three multi-modal synthetic problems, PaDGAN outperforms all other methods by achieving both high quality and high diversity. Finally, after showing that the claims hold on three test cases, we apply PaDGAN on a real-world airfoil synthesis problem. We find that PaDGAN can discover new regions of high-quality designs, which are outside the design domain over which it was trained.

4.1 Data and Quality Measure

Synthetic Example I. The purpose of creating 2D synthetic examples is to test the performance of PaDGAN given known ground truth and visualize the results in terms of diversity and quality. These examples are analogical to any 2D design problem, where designs are represented by two variables. In this synthetic example I, we generate a ring-shaped dataset, with 10,000 samples uniformly distributed between two origin-centered circles of 0.25 and 0.5 in radius, respectively (Fig. 3). We use a synthetic density function to define the quality of each sample. Specifically, we use the following density function of an unnormalized Gaussian mixture as the quality function:

$$q(\mathbf{x}) = \sum_{k=1}^{K} \exp\left(-\frac{(\mathbf{x} - \mu_k)^T(\mathbf{x} - \mu_k)}{2\sigma^2}\right) \tag{9}$$

where μ_k is the mode of the kth mixture component and σ is the standard deviation. The centers $\mu_1, \ldots, \mu_K$ are evenly spaced around a circle centered at the origin and with a radius of 0.4. We set $K = 6$ and $\sigma \approx 0.1$. Hence, there are six peaks of quality and points are evenly spread between two concentric circles in the training data. Samples that receive a higher value from the quality function are considered to be of higher quality. While the top row of Fig. 3 shows the design space, the areas of high-quality samples (performance space) is shown by light color areas in the bottom row. Ideally, by simultaneously maximizing diversity and quality, we expect generating more samples near the six local optima (i.e., modes) of the quality function, and those samples should be spread out and evenly distributed among all six mixture components.
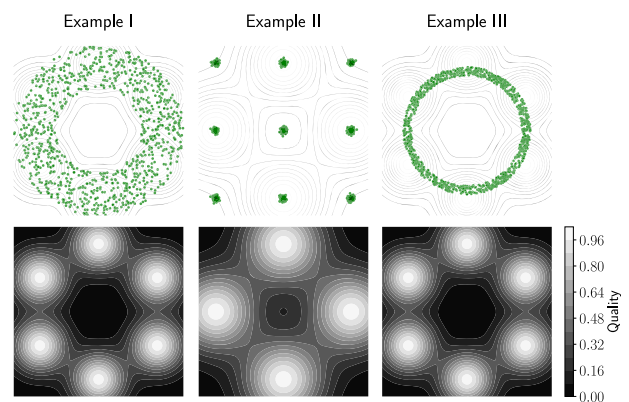

Fig. 3 Data and quality functions in synthetic examples. The dots in the top plots represent data and the contours represent quality functions.
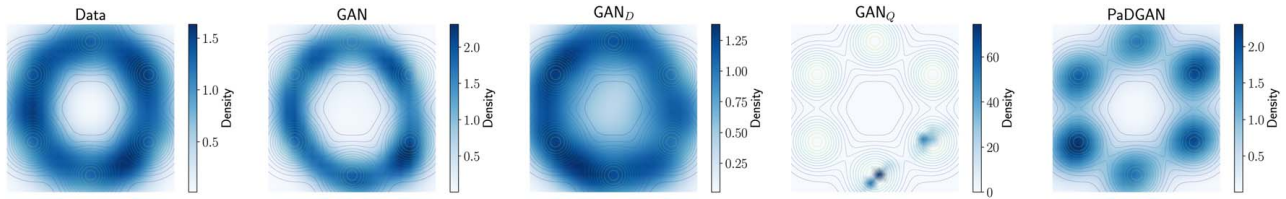
Fig. 4 Results on synthetic example I. The leftmost plot show the density of training data. The rest of the plots show the density of samples generated by different models.
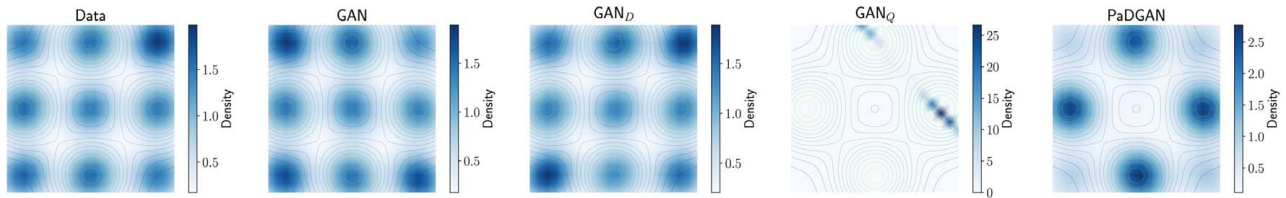

Fig. 5 Results on synthetic example II. The leftmost plot show the density of training data. The rest of the plots show the density of samples generated by different models.
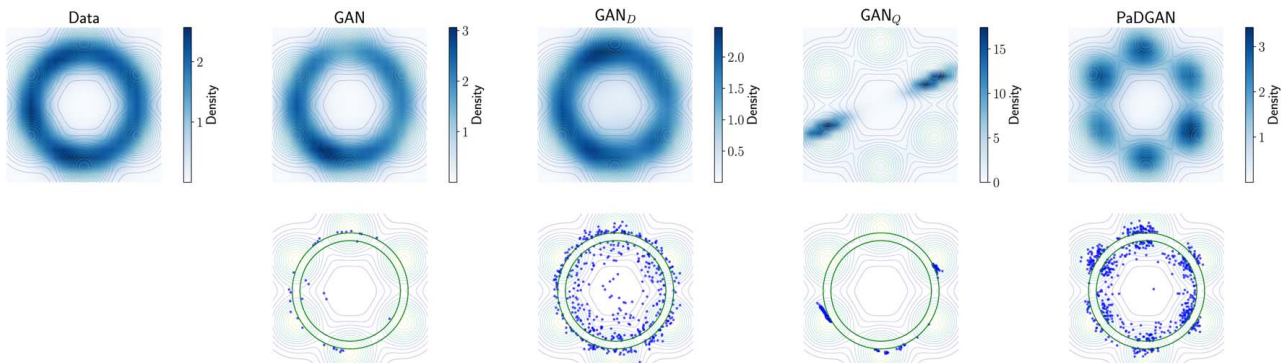

Fig. 6 Results on synthetic example III. The leftmost plot show the density of training data. The remaining plots in the top row show the density of samples generated by different models. We observe that GAN and *GAN_D* generate samples similar to the data, while *GAN_Q* suffers with mode collapse and only generates samples for two clusters. PaDGAN generates more samples in the high-quality region, ignoring the low-quality areas of the training data. Plots in the bottom row show only the generated samples (dots), which are "outside" the region of training data (indicated by two circles). We observe that PaDGAN generates unseen data in the high-quality areas, while other methods either do not explore outside the domain or generate low-quality samples.

Synthetic Example II. The data in this example have nine clusters placed on a 3×3 grid (Fig. 3). The sample size is 10,000. Similar to synthetic example I, we use Eq. (9) as the quality function. Here, we set $K = 4$ and $\sigma \approx 0.16$. Four of nine clusters (modes) of the data overlap with local optima of the quality function. We expect that if both diversity and quality are considered, the generator should produce most samples in all the four high-quality clusters and few samples in other clusters (instead of generating most samples from a single high-quality cluster).

Synthetic Example III. This example is the same as example I, except that data are bounded within two origin-centered circles of 0.325 and 0.375 in radius (Fig. 3). The purpose of decreasing the coverage of data is to demonstrate PaDGAN's capability of extrapolating in the high-quality regions (i.e., expanding the boundary of existing design space toward the high-quality regions). The sample size is also 10,000.

Airfoil Example. An airfoil is the cross-sectional shape of a wing or a propeller/rotor/turbine blade. In this example, we use the UIUC airfoil database[2] as our data source. It provides the geometries of

[2]http://m-selig.ae.illinois.edu/ads/coord_database.html

nearly 1,600 real-world airfoil designs. We preprocessed and augmented the dataset based on Ref. [6] to generate a dataset of 38,802 airfoils. Each design is represented by 192 discrete 2D coordinates along their suction (upper) and pressure (lower) surfaces, which leads to a design space dimensionality of 384. The lift-to-drag ratio C_L/C_D is a common objective in aerodynamic design optimization problems. Thus we used C_L/C_D as the performance measure, which can be computed using XFOIL software [46]. To provide the gradient of the quality function for Eq. (8), we trained a neural network-based surrogate model on all 38,802 airfoils to approximate the quality. Note that for all the examples, we scaled the quality scores between 0 and 1. We show a subset of 100 randomly chosen example airfoils from the training data in the left plot of Fig. 9.

4.2 Model Configuration and Training. To demonstrate the effectiveness of the PaDGAN, we compare it with the following three models:

(1) GAN: a vanilla GAN with the objective of Eq. (1).
(2) GAN_D: PaDGAN with $\gamma_0 = 0$ in Eq. (5), i.e., which only optimizes for diversity and ignores the quality.
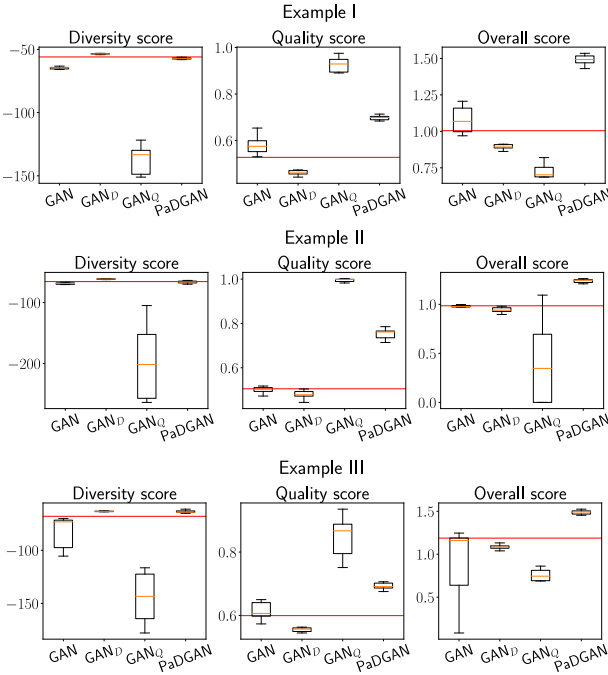
Fig. 7 Scores for synthetic examples. The horizontal line denotes the scores of the training data. The box plots show the statistics of ten models for each method.
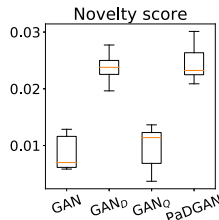

Fig. 8 Novelty scores for example III. The box plots show the statistics of ten models for each method.

(3) GAN$_Q$: a vanilla GAN, which ignores diversity and only optimizes for the quality using the following additional term $\mathcal{L}_Q(G) = -\frac{1}{|B|}\sum_{i=1}^{|B|} q(\mathbf{x}_i)$. The training objective is then set to:

$$\min_G \max_D V(D, G) + \gamma_2 \mathcal{L}_Q(G)$$

where γ_2 controls the weight of the quality objective.

To find similarity between designs, we use a radial basis function (RBF) kernel with a bandwidth of 1.0 when constructing L_B in Eq. (5), i.e., $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-0.5\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$. This gives a value between 0 and 1, with a higher value for more similar designs. In synthetic examples, we set $\gamma_0 = 2$ and $\gamma_1 = 0.5$ for PaDGAN and $\gamma_2 = 10$ for GAN$_Q$. We conduct a parametric study to show how γ_0 and γ_1 affect PaDGAN's performance and include the results in Appendix B. The generators and discriminators are fully connected neural networks. In the airfoil example, we set $\gamma_0 = 2$ and $\gamma_1 = 0.2$ for PaDGAN. We used a residual neural network (ResNet) [47] as the surrogate model and a BézierGAN [6,28] to generate airfoils. For simplicity, we refer to the BézierGAN as a vanilla GAN and the BézierGAN with loss $\mathcal{L}_{PaD}$ as a PaDGAN in the airfoil example in the rest of the article. For all the experiments, we use Adam [48] as the optimizer to train neural networks and set the learning rate of both G and D to 0.0001. The batch size is 32.

Weights are initialized from a uniform distribution. Detailed network architecture and hyperparameter settings can be found in our open-source code.[3]

4.3 Evaluation. We use the *diversity score* and the *quality score* of generated samples to measure the performance of generative models. The diversity score is expressed as the mean log determinant of the similarity matrix:

$$s_{\text{div}} = \frac{1}{n}\sum_{i=1}^{n} \log \det(L_{S_i}) \tag{10}$$

where n is the number of times diversity is evaluated, $S_i \subseteq Y$ is a random subset of Y (the set of generated samples), and L_{S_i} is the similarity matrix of S_i with entries $L_{S_i}(j, k) = k(\mathbf{x}_j, \mathbf{x}_k)$ for each $\mathbf{x}_j$, $\mathbf{x}_k \in S_i$. The quality score is computed by taking the average quality of generated samples:

$$s_{\text{qa}} = \frac{1}{|Y|}\sum_{i=1}^{|Y|} q(\mathbf{x}_i) \tag{11}$$

where $\mathbf{x}_i \in Y$ is a randomly generated design.

For synthetic examples, we define the *overall score* to measure the overall performance by combining measures for diversity and quality of generated samples:

$$s_{\text{overall}} = -\sum_k \frac{m_k}{|Y|} \log\left(\frac{m_k}{|Y|}\right) \tag{12}$$

where m_k is the number of generated samples within the one-sigma interval of the kth mixture component of the quality function. The overall score is affected by both the amount of high-quality samples and the spread of those samples. The highest score occurs when there are the same number of generated samples within the one-sigma interval of each mixture component and no samples are outside those intervals.

We use the *novelty score* to evaluate how different generated samples are from the training data. Specifically, for each generated sample, novelty is measured by the distance from its nearest training sample. The novelty score is computed by taking the average of those nearest distances:

$$s_{\text{novel}} = \frac{1}{|Y|}\sum_{i=1}^{|Y|} \min_{\mathbf{x}'_j \in Y'} D(\mathbf{x}_i, \mathbf{x}'_j) \tag{13}$$

where Y' is the set of training samples and D is a distance or dissimilarity measure. We set D to be the Euclidean distance in the synthetic example and Hausdorff distance in the airfoil example.

In the experiments, we set $|Y| = 1000$, $|S_i| = 10$, and $n = 1000$. To take into consideration the stochasticity of the model training, for each type of model (PaDGAN, GAN, GAN$_D$, and GAN$_Q$), we train them ten times for each experimental setting and report the performance statistics for all those ten models (Figs. 7 and 11). We report and discuss the results in Sec. 5.

5 Results and Discussion

In this section, we compare the performance of PaDGAN with its alternatives (i.e., GAN, GAN$_D$, and GAN$_Q$) and discuss the implication of these results.

5.1 Synthetic Examples. Figures 4–6 show the density plots of generated samples for each model, which represents their *generative distribution*. Ideally, when we sample designs from the

---

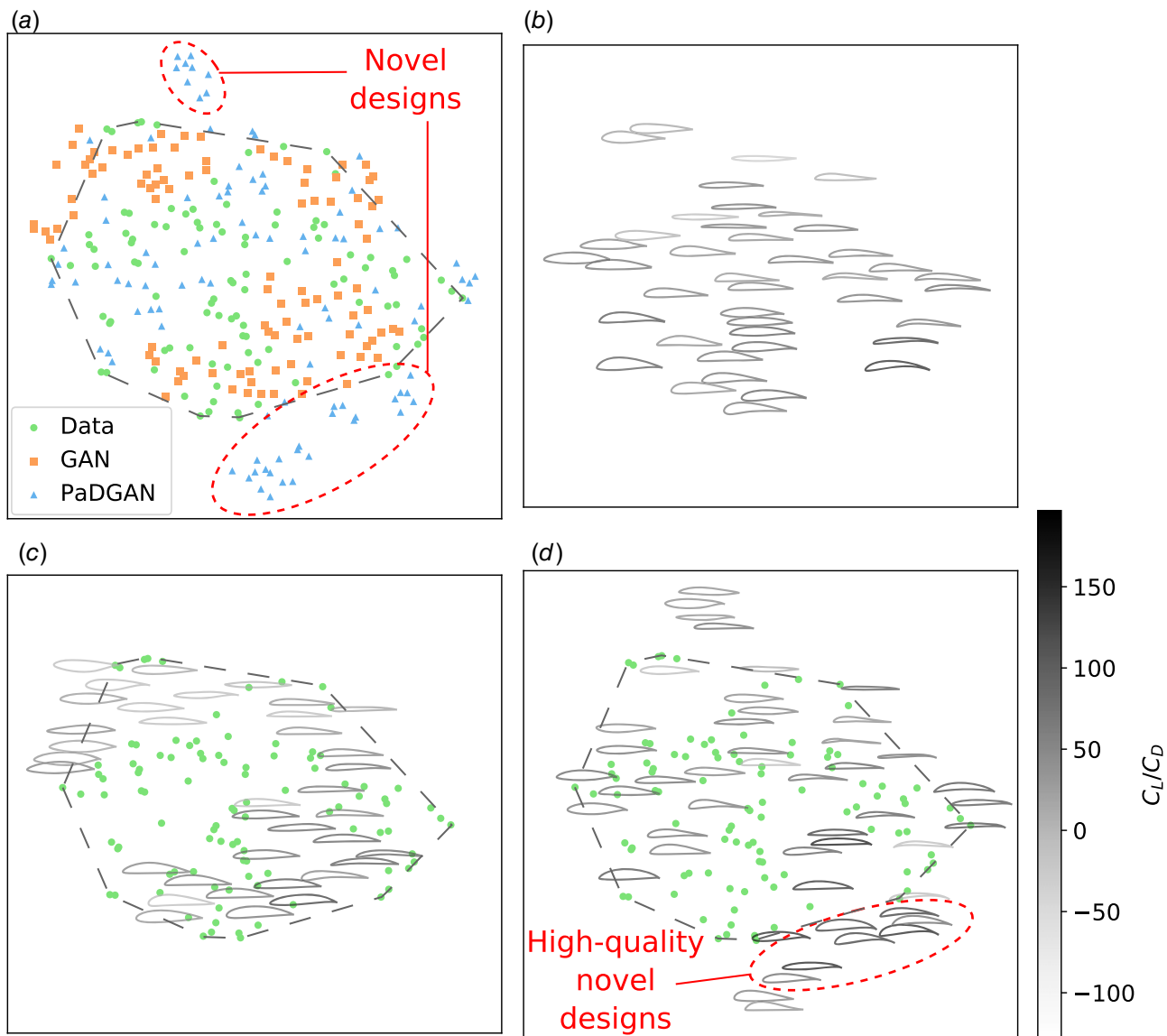[3]https://github.com/wchen459/PaDGAN

Fig. 9 To compare the distribution of real and synthetic airfoils, we map airfoil designs sampled randomly from training data, vanilla GAN, and PaDGAN through t-SNE into the same 2D space (shown in (a)). (b)–(d) The airfoil geometries, where the shades represent quality (i.e., C_L/C_D). The dots in (c) and (d) represent the training data. We label the convex hull of the sampled training data in (a), (c), and (d), which roughly indicates the boundary of the original design space.

generator, we want these designs to have a good coverage over real-world designs (i.e., the training data) and most of them should have high quality. In Fig. 4, the generative distribution learned by a vanilla GAN fails to cover the entire training data (nonuniform contours) (Fig. 7). However, in both examples I and II, the generative distribution of GAN_D has a good coverage of the training data due to its diversity objective. This shows that the diversity objective by itself is capable of avoiding mode collapse. By replacing the diversity objective with a quality objective, GAN_Q only generates samples near one of the optima of the quality functions, ignoring the others. In practice, this will give many high-quality samples, but they all look very similar to each other. In contrast, the generative distribution of PaDGAN exhibits has a higher density near high-quality regions and also good coverage of the design space.

Example III is intentionally created to demonstrate the ability of PaDGAN for expanding the boundary of training data. Figure 6 shows that both GAN_D and PaDGAN generates samples outside the training data's boundary. Particularly, PaDGAN expands the boundary toward high-quality regions. If these samples represent designs, it basically indicates that PaDGAN can expand the

boundary of existing designs and generate completely novel designs. We will further demonstrate this with a real design problem later. Figure 8 compares the novelty scores of different models for example III and shows that GAN_D and PaDGAN have much higher novelty scores than the vanilla GAN and GAN_Q, which is consistent with Fig. 6. This promising result indicates that by diversifying generated samples, PaDGAN is capable of expanding the design space toward the direction of high-quality regions. Note that this is not only filling the "holes" of the design space by interpolation but also *extrapolation* on the right direction. It is not surprising that the generator knows which direction to expand since it receives from the performance estimator the information of quality gradients.

Figure 7 shows the statistics of ten trained models for each method. For all three synthetic examples, GAN_D has the best performance in the diversity score and the worst performance in the quality score. GAN_Q generates the highest quality samples, but has the lowest diversity scores, showing that all the samples very similar to each other. PaDGAN has the highest overall score in all examples, which shows that it generates high-quality samples
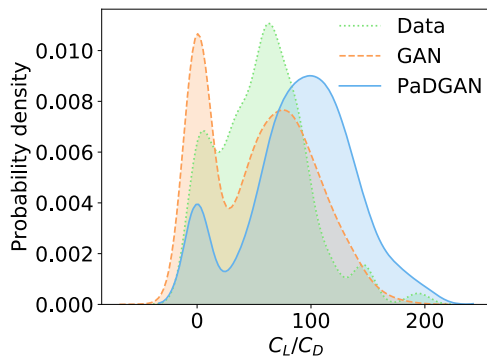
Fig. 10 The quality (C_L/C_D) distributions for training data, vanilla GAN, and PaDGAN. The Kullback–Leibler divergence is 0.7245 between the quality distributions of data and vanilla GAN's samples and is 1.1013 between the quality distributions of data and PaDGAN's samples.
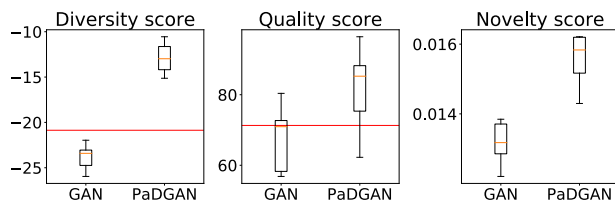

Fig. 11 Scores for the airfoil example. The horizontal line denotes the scores of the training data (the novelty score for training data is 0 based on Eq. (13)). The box plots show the statistics of ten models for each method.

that spread over different optima. The lowest variance indicates a consistent performance over multiple runs of PaDGAN training.

5.2 Airfoil Example. We synthesized 100 airfoil designs from a vanilla GAN and 100 from a PaDGAN, computed their quality (C_L/C_D values) using XFOIL[4], and used the t-distributed stochastic neighbor embedding (t-SNE) to map these designs onto the same two-dimensional space, as shown in Fig. 9. The quality is indicated by the shades of plotted designs, where dark shaded airfoils are of higher quality. We also show 100 designs from the training data in the left most figure to represent the original design space. Both the GAN and the PaDGAN generate realistic airfoil designs. We observe that the vanilla GAN (middle figure) generates a few airfoils that fill in the gaps of the training data (i.e., interpolation). However, PaDGAN discovers new high-quality designs, which are outside the boundary of the training data. We mark these regions in by ellipses in the leftmost part of Fig. 9. This shows that the diversity promoting part of PaDGAN encourages it to discover new unseen design areas, while the quality promoting part helps it find areas where high-quality designs are found, as is also demonstrated by synthetic example III. In future work, we will explore if PaDGAN can be used as a tool to assist in design discovery by generating novel high-quality designs for more complex design domains.

We show the quality (i.e., C_L/C_D) distributions of training data and generated designs by vanilla GAN and PaDGAN in Fig. 10. We observe that the quality distribution of data has two modes (large number of samples)—one near 0 and one near 70. The vanilla GAN's quality distribution mimics these two modes but has a larger probability mass near 0. Comparing with both the training data and the vanilla GAN, PaDGAN's quality distribution has a larger mass over the higher quality region. This shows that

---

[4]We set $C_L/C_D = 0$ when the simulation fails.

PaDGAN generates most samples which are of significantly higher quality than the training data.

Figure 11 shows the statistics of quality, diversity, and novelty scores over ten runs of model training. The PaDGAN's diversity score is always higher than the training data's (shown by a horizontal line), whereas the vanilla GAN almost always has a lower diversity score than the data. The quality scores of most PaDGAN models are higher than the vanilla GAN models. PaDGAN also has higher novelty scores than the vanilla GAN. These results demonstrate the effectiveness of PaDGAN as a design exploration tool.

To show the evaluation scores in Figs. 7 and 11 more clearly, we list the means and 95% confidential intervals of all scores in Appendix A.

6 Conclusion and Future Work

In this article, we proposed a new loss function for generative models based on determinantal point processes. With this loss function, we developed a new GAN model, named PaDGAN. To the best of authors' knowledge, this is the first GAN model that can simultaneously encourage the generation of diverse and high-quality designs. We use both synthetic and real-world examples to demonstrate the effectiveness of PaDGAN and show that by diversifying generated samples, PaDGAN expands the existing boundary of the design space toward high-quality regions. This model is particularly useful when we want to thoroughly explore different high-quality design alternatives or discover novel solutions. For example, when performing design optimization, one may accelerate the search for global optimal solutions by sampling start points from the proposed model. Also, this method can be a tool in the early conceptual design stage to aid the creative process. It can generate new designs that are learnt from previous generations of designs, while introducing novelty and taking into account the desired quality metrics. The resultant designs can be used as inspirations to steer designers in exploring novel designs. Although we demonstrated the effectiveness of our method via a GAN-based model, the proposed framework also generalizes to other generative models like variational autoencoders and can be used for various design synthesis problems.

Note that by trying to mimic the training data, PaDGAN captures design constraints implicitly. For instance, in Fig. 6 (example III), it captures the inner and the outer ring of the training data and generates the majority of the points inside the two circular rings. However, we still observe a few points outside the rings, as we do not explicitly define this as a constraint boundary. To explicitly capture design constraints, one can train a differentiable classifier (e.g., a neural network-based classifier), which predicts constraint satisfaction and use it as a second discriminator. However, this approach of explicitly capturing the constraints is outside the scope of this work.

In this work, we only model quality as a scalar. When the quality is indicated by multiple factors, we can convert those factors into a single factor using approaches like scalarization. However, this only pushes generated designs along one direction toward the Pareto front. In the future, we will extend this work to model multidimensional quality and allow generated designs to be pushed toward the entire Pareto front. The performance augmented DPP loss added to the GAN loss resembles a weighted sum of two objectives of a multi-objective optimization problem. However, the relationship between the two terms is not necessarily conflicting and depends on the distribution of samples in the design and performance space. While GAN loss functions often use a weighted sum approach, it may have a few drawbacks like: (1) the weighted approach gives a single trade-off solution, and one has to re-train the model to increase or decrease the importance of the performance augmented DPP loss; and (2) setting a numerical weight between

the two terms by a practitioner is difficult due to the dependence of values on the data.

Theoretically, the performance augmented DPP loss proposed in this work can be added as a regularization term in the loss function when training any deep generative models with two requirements—there should be a method to quantify similarity between items and each item should have a differentiable quality or performance model. However, this does not guarantee that, in practice, this regularization term would not introduce convergence/stability issues to the training. Particularly, one of our heuristics for improving training stability is to weight the quality by the probability predicted by the discriminator (Sec. 3.4). This practical consideration is specific to GANs and will not be compatible if using another deep generative model such as the VAE or the flow-based generative model.

There are also parallels between determinant of the kernel matrix and other coverage metrics (like hypervolume indicator, convex hull), which can also be considered for improving the diversity of solutions. While a measure like hypervolume indicator is more accurate that the parallelopiped volume in measuring the volume of a set of points, it is impractical for GAN training due two reasons. First, the computational complexity of hypervolume indicator calculation is $O(n \log n + n^{d/2})$, where n is the sample size and d is the dimensionality of each sample. Specifically, the dimensionality d has a high impact on the complexity. In practice, d is usually large (e.g., $d = 384$ for our airfoil example and can be thousands for more complex designs). This makes using hypervolume indicator for diversity measurement impractical for GAN training. In contrast, the complexity of determinant calculation is $O(n^3)$, which only depends on the number of samples in a batch ($n = 32$ in our examples). Thus, the computational cost is acceptable. Second, the DPPs allow for an mathematically elegant way of balancing quality and diversity and enable efficient ways of computing marginals, computing conditional probabilities, and sampling in polynomial time.

While we developed this method for engineering design applications, it can generalize to many other domains, where quality and coverage over a domain are needed. For example, in molecule discovery, our model can be integrated with the generative model developed by Gómez-Bombarelli et al. [49], who combined a generative model with the search over latent space to generate new molecules. In 3D shape synthesis, our model can be trained on large datasets like ShapeNet and used as a recommender system within CAD software. The loss function we develop can also be integrated with human face synthesis methods to generate new human faces, which are high quality (depending on any criteria like beauty) and from different groups (regions, race, gender, age, etc.). Overall, the method provides a new direction of research, where generative models focus on the unbiased generation of high-quality items.

Conflicts of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request. The data and information that support the findings of this article are freely available at: https://github.com/wchen459/PaDGAN. The authors attest that all data for this study are included in the paper. Data provided by a third party listed in Acknowledgment.

Nomenclature

q = quality function
$\mathbf{x}$ = design variables
$\mathbf{z}$ = noise vector
B = a batch of generated samples draw from Y
D = discriminator
G = generator
Y = the set of generated samples
L_S = DPP kernel matrix for a set S

$P_{\mathbf{z}}$ = noise distribution
P_{data} = data distribution
γ_0 = weight of quality in the performance augmented DPP loss
γ_1 = weight of the performance augmented DPP loss in the PaDGAN loss

Appendix A: Table of Evaluation Metrics

We list the means and 95% confidence intervals of all evaluation scores (Figs. 7 and 11) in Table 1. It shows that PaDGAN received best overall score for all cases and at least the second best score for both diversity and quality in all four examples.

Appendix B: Parametric Study

While our main results use a fixed value of γ_0 (quality dial) and γ_1 (DPP dial), practitioners may wonder how the performance of PaDGAN is impacted by change in the value of these parameters. To show this, we conduct a parametric study. In the first

Table 1 Diversity and quality scores for all experiments

	Model	Diversity score	Quality score	Overall score	Novelty score
Example I	GAN	-69.0885 ± 27.4357	0.5826 ± 0.0620	1.0678 ± 0.2938[a]	N/A
	GAN$_D$	-53.6183 ± 0.3601[b]	0.4672 ± 0.0162	0.9060 ± 0.0414	N/A
	GAN$_Q$	-152.1945 ± 77.5582	0.9131 ± 0.1065[b]	0.5913 ± 0.5875	N/A
	PaDGAN	-57.2489 ± 1.16202[a]	0.6955 ± 0.0269[a]	1.4897 ± 0.0624[b]	N/A
Example II	GAN	-68.2451 ± 3.9807	0.5004 ± 0.0447	0.9752 ± 0.0531[a]	N/A
	GAN$_D$	-61.8685 ± 0.7375[b]	0.4811 ± 0.0239	0.9473 ± 0.0269	N/A
	GAN$_Q$	-197.0243 ± 119.0978	0.9914 ± 0.0280[b]	0.4261 ± 0.8780	N/A
	PaDGAN	-65.5501 ± 1.53222[a]	0.7176 ± 0.0344[a]	1.2117 ± 0.0320[b]	N/A
Example III	GAN	-93.1237 ± 67.4991	0.6219 ± 0.0847	0.9019 ± 0.8333	0.0138 ± 0.0276
	GAN$_D$	-63.5478 ± 0.4637[b]	0.5560 ± 0.0152	1.0811 ± 0.0468[a]	0.0230 ± 0.0050[a]
	GAN$_Q$	-150.3896 ± 65.4173	0.8492 ± 0.1219[b]	0.6356 ± 0.6366	0.0107 ± 0.0114
	PaDGAN	-63.9063 ± 1.76832[a]	0.6968 ± 0.0248[a]	1.4993 ± 0.0624[b]	0.0238 ± 0.0051[b]
Airfoil	GAN	-23.8758 ± 2.4789[a]	63.7383 ± 34.2794[a]	N/A	0.0135 ± 0.0024[a]
	PaDGAN	-12.4373 ± 4.8516[b]	77.2618 ± 43.7569[b]	N/A	0.0162 ± 0.0032[b]

[a]The second best score for each example.
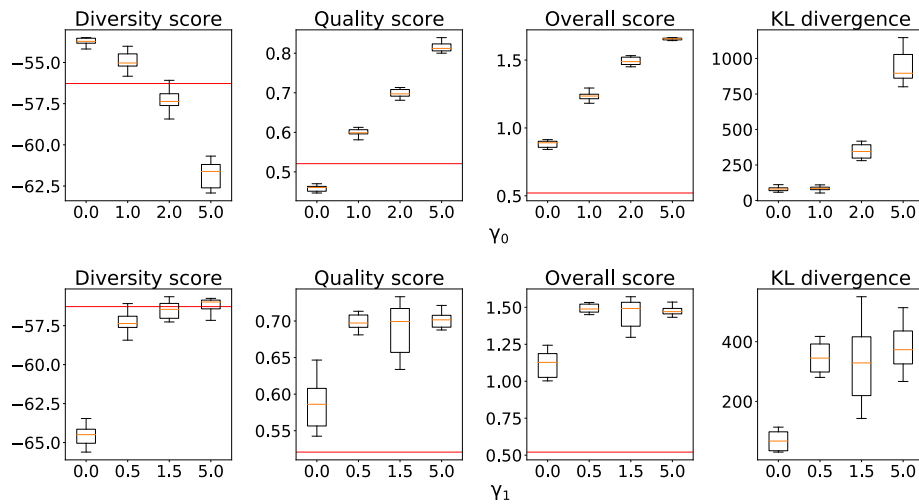[b]The best score for each example.

Fig. 12 Effect of γ_0 (top) and γ_1 (bottom) on performance indicators for synthetic example I. The horizontal line denotes the scores of the training data. The box plots show the statistics of ten models for each method.
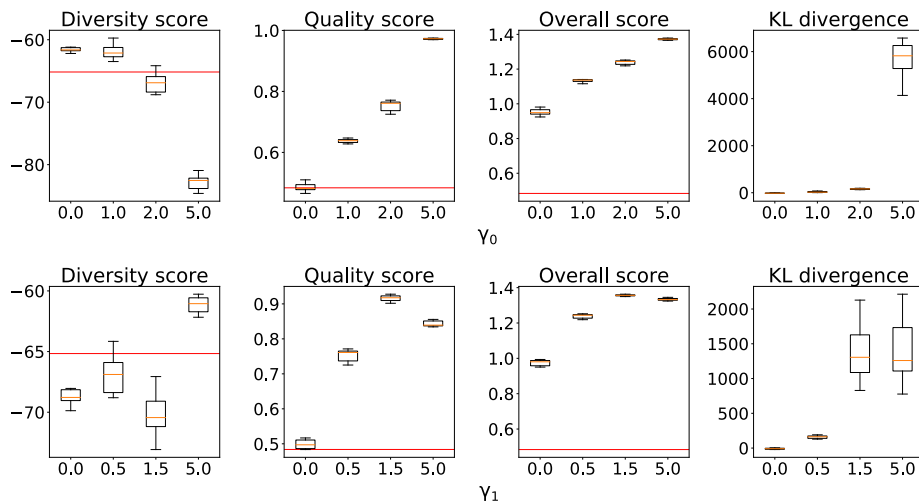

Fig. 13 Effect of γ_0 (top) and γ_1 (bottom) on performance indicators for synthetic example II. The horizontal line denotes the scores of the training data. The box plots show the statistics of ten models for each method.
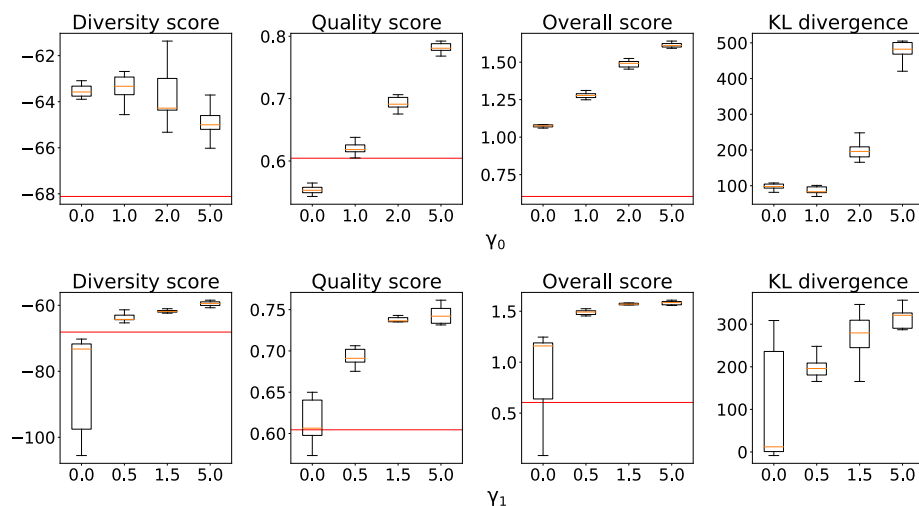

Fig. 14 Effect of γ_0 (top) and γ_1 (bottom) on performance indicators for synthetic example III. The horizontal line denotes the scores of the training data. The box plots show the statistics of ten models for each method.
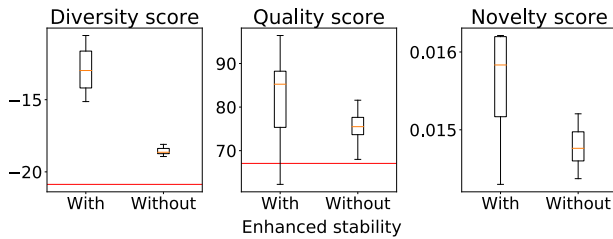
Fig. 15 Comparison of scores with or without enhancing stability introduced in Sec. 3.4. The horizontal line denotes the scores of the training data (the novelty score for training data is 0 based on Eq. (13)).

experiment, γ_1 is fixed at 0.5 (same as our experiments) and γ_0 is varied; while in the second experiment, γ_0 is fixed at 2 and γ_1 is varied. The results are shown in Figs. 12–14. Since γ_0 controls the weight of quality in the performance augmented DPP loss, increasing it decreases the diversity score but increases the quality score and the overall score. However, when setting it to very large values ($\gamma_0 > 5$), training became unstable due to exploding gradients. Meanwhile, γ_1 controls the weight of the performance augmented DPP loss over the standard GAN loss. Thus, in general, all scores increase with an increase in γ_1 until a point, where we either see a plateau or a decrease of scores. This behavior depends on how diversity and quality interact with the fit to data and whether increase in the former is detrimental to the latter. We observe that setting $\gamma_1 > 5$ in most cases also leads to unstable training. This is because too much focus on the performance augmented DPP loss brings convergence issues to the standard GAN's objective.

We also measured the KL divergence between the quality distributions of data and each model's generated samples. The effects of γ_0 and γ_1 on KL divergence share a similar pattern with their effects on the quality score or the overall score. This is expected since the mode of the quality distribution is shifted toward higher quality regions when the quality score is higher.

Appendix C: Effects of Enhancing PaDGAN Stability

With the airfoil design example, we demonstrate the effects of the realisticity weighted quality and the escalating schedule for γ_1 introduced in Sec. 3.4. These two considerations are for the purpose of stabilizing PaDGAN's training when using a data-driven surrogate model for quality prediction. Figure 15 shows that without those considerations, all three scores are worse in most cases, which indicates a necessity to incorporate those two settings while training a PaDGAN.

References

[1] Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Hernandez, N. V., and Wood, K. L., 2011, "Computer-Based Design Synthesis Research: An Overview," ASME J. Comput. Inf. Sci. Eng., **11**(2), p. 021003.
[2] Kingma, D. P., and Welling, M., 2014, "Auto-Encoding Variational Bayes," 2nd International Conference on Learning Representations, Banff, AB, Canada, Apr. 14–16.
[3] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2014, "Generative Adversarial Nets," Advances in Neural Information Processing Systems, Montreal, Quebec, Canada, Dec. 8–13, pp. 2672–2680.
[4] Chen, W., Fuge, M., and Chazan, J., 2017, "Design Manifolds Capture the Intrinsic Complexity and Dimension of Design Spaces," ASME J. Mech. Des., **139**(5), p. 051102.
[5] Chen, W., and Fuge, M., 2019, "Synthesizing Designs With Interpart Dependencies Using Hierarchical Generative Adversarial Networks," ASME J. Mech. Des., **141**(11), p. 111403.
[6] Chen, W., Chiu, K., and Fuge, M., 2019, "Aerodynamic Design Optimization and Shape Exploration Using Generative Adversarial Networks," AIAA SciTech Forum, San Diego, CA, Jan. 7–11.
[7] Chen, W., Chiu, K., and Fuge, M., 2020, "Airfoil Design Parameterization and Optimization Using Bézier Generative Adversarial Networks," AIAA J.
[8] Bendsoe, M. P., and Sigmund, O., 2004, Topology Optimization: Theory, Methods and Applications, Springer, New York.
[9] Ahmed, F., Deb, K., and Bhattacharya, B., 2016, "Structural Topology Optimization Using Multi-Objective Genetic Algorithm With Constructive Solid Geometry Representation," Appl. Soft. Comput., **39**, pp. 240–250.
[10] Shu, D., Cunningham, J., Stump, G., Miller, S. W., Yukish, M. A., Simpson, T. W., and Tucker, C. S., 2020, "3d Design Using Generative Adversarial Networks and Physics-Based Validation," ASME J. Mech. Des., **142**(7), p. 071701.
[11] Kulesza, A., and Taskar, B., 2012, "Determinantal Point Processes for Machine Learning," Found. Trends Mach. Learn., **5**(2–3), pp. 123–286.
[12] Goodfellow, I., Bengio, Y., and Courville, A., 2016, Deep Learning, MIT Press, Cambridge, MA.
[13] Gmeiner, T., and Shea, K., 2013, "A Spatial Grammar for the Computational Design Synthesis of Vise Jaws," ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Portland, OR, Aug. 4–7.
[14] Königseder, C., Stanković, T., and Shea, K., 2016, "Improving Design Grammar Development and Application Through Network-Based Analysis of Transition Graphs," Design Sci., **2**, p. e5.
[15] Shea, K., Aish, R., and Gourtovaia, M., 2005, "Towards Integrated Performance-Driven Generative Design Tools," Auto. Construction, **14**(2), pp. 253–264.
[16] Herber, D. R., Guo, T., and Allison, J. T., 2017, "Enumeration of Architectures With Perfect Matchings," ASME J. Mech. Des., **139**(5), p. 051403.
[17] Kamesh, V. V., Mallikarjuna Rao, K., Rao, S., and Balaji, A., 2017, "Topological Synthesis of Epicyclic Gear Trains Using Vertex Incidence Polynomial," ASME J. Mech. Des., **139**(6), p. 062304.
[18] Bryant, C. R., Stone, R. B., McAdams, D. A., Kurtoglu, T., and Campbell, M. I., 2005, "Concept Generation From the Functional Basis of Design," ICED 05: 15th International Conference on Engineering Design: Engineering Design and the Global Economy, Melbourne, Australia, Aug. 15–18, pp. 280–281.
[19] Wyatt, D. F., Wynn, D. C., Jarrett, J. P., and Clarkson, P. J., 2012, "Supporting Product Architecture Design Using Computational Design Synthesis With Network Structure Constraints," Res. Eng. Design, **23**(1), pp. 17–52.
[20] Wijkniet, J., and Hofman, T., 2018, "Modified Computational Design Synthesis Using Simulation-Based Evaluation and Constraint Consistency for Vehicle Powertrain Systems," IEEE Trans. Vehicular Technol., **67**(9), pp. 8065–8076.
[21] Chen, X., Diez, M., Kandasamy, M., Zhang, Z., Campana, E. F., and Stern, F., 2015, "High-Fidelity Global Optimization of Shape Design by Dimensionality Reduction, Metamodels and Deterministic Particle Swarm," Eng. Optim., **47**(4), pp. 473–494.
[22] D'Agostino, D., Serani, A., Campana, E. F., and Diez, M., 2017, "Nonlinear Methods for Design-Space Dimensionality Reduction in Shape Optimization," Machine Learning, Optimization, and Big Data – Third International Conference, Volterra, Italy, Sept. 14–17, pp. 121–132.
[23] D'Agostino, D., Serani, A., Campana, E. F., and Diez, M., 2018, "Deep Autoencoder for Off-Line Design-Space Dimensionality Reduction in Shape Optimization," 2018 AIAA/ASCE/AHS/ASC, Structures Structural Dynamics, and Materials Conference, Kissimmee, FL, Jan. 8–12.
[24] Burnap, A., Liu, Y., Pan, Y., Lee, H., Gonzalez, R., and Papalambros, P. Y., 2016, "Estimating and Exploring the Product Form Design Space Using Deep Generative Models," ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Charlotte, NC, Aug. 21–24.
[25] Cunningham, J. D., Simpson, T. W., and Tucker, C. S., 2019, "An Investigation of Surrogate Models for Efficient Performance-Based Decoding of 3d Point Clouds," ASME J. Mech. Des., **141**(12), p. 121401.
[26] Cang, R., Vipradas, A., and Ren, Y., 2017, "Scalable Microstructure Reconstruction With Multi-Scale Pattern Preservation," ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Cleveland, OH, Aug. 6–9.
[27] Yang, Z., Li, X., Catherine Brinson, L., Choudhary, A. N., Chen, W., and Agrawal, A., 2018, "Microstructural Materials Design Via Deep Adversarial Learning Methodology," ASME J. Mech. Des., **140**(11), p. 111416.
[28] Chen, W., and Fuge, M., 2018, "Béziergan: Automatic Generation of Smooth Curves From Interpretable Low-Dimensional Parameters," Preprint arXiv:1808.08871.
[29] Oh, S., Jung, Y., Kim, S., Lee, I., and Kang, N., 2019, "Deep Generative Design: Integration of Topology Optimization and Generative Models," ASME J. Mech. Des., **141**(11), p. 111405.
[30] Burnap, A., Hauser, J. R., and Timoshenko, A., 2019, "Design and Evaluation of Product Aesthetics: A Human-Machine Hybrid Approach," CoRR, abs/1907.07786.
[31] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X., 2016, "Improved Techniques for Training GANs," Advances in Neural Information Processing Systems, Barcelona, Spain, Dec. 5–10, pp. 2226–2234.
[32] Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S., 2017, "Least Squares Generative Adversarial Networks," Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, Oct. 22–29, pp. 2813–2821.
[33] Bang, D., and Shim, H., 2018, "Mgan: Solving Mode Collapse Using Manifold Guided Training," Preprint arXiv:1804.04391.
[34] Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., and Sutton, C., 2017, "VEEGAN: Reducing Mode Collapse in Gans Using Implicit Variational

Learning," Advances in Neural Information Processing Systems, Long Beach, CA, Dec. 4–9, pp. 3308–3318.

[35] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P., 2016, "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets," Advances in Neural Information Processing Systems, Barcelona, Spain, Dec. 5–10, pp. 2172–2180.

[36] Elfeki, M., Couprie, C., Riviere, M., and Elhoseiny, M., 2019, "GDPP: Learning Diverse Generations Using Determinantal Point Processes," International Conference on Machine Learning, Long Beach, CA, June 9–15, pp. 1774–1783.

[37] Dube, A., and Helkkula, A., 2016, "Customer Approach to the Use of Big Data: Wearables for Service," Proceedings of SERVSIG 2016 Conference, Maastricht, The Netherlands, June 17–19.

[38] Lin, H., and Bilmes, J., 2011, "A Class of Submodular Functions for Document Summarization," Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies–Volume 1, Portland, OR, June 19–24, pp. 510–520.

[39] Shah, J. J., Kulkarni, S. V., and Vargas-Hernandez, N., 2000, "Evaluation of Idea Generation Methods for Conceptual Design: Effectiveness Metrics and Design of Experiments," ASME J. Mech. Des., 122(4), pp. 377–384.

[40] Fuge, M., Stroud, J., and Agogino, A., 2013, "Automatically Inferring Metrics for Design Creativity," International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Portland, OR, Aug. 4–7.

[41] Ahmed, F., Ramachandran, S. K., Fuge, M., Hunter, S., and Miller, S., 2019, "Measuring and Optimizing Design Variety Using Herfindahl Index," ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Anaheim, CA, Aug. 18–21.

[42] Ahmed, F., and Fuge, M., 2017, "Ranking Ideas for Diversity and Quality," ASME J. Mech. Des., 140(1), p. 011101.

[43] Ahmed, F., Fuge, M., and Gorbunov, L. D., 2016, "Discovering Diverse, High Quality Design Ideas From a Large Corpus," ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Charlotte, NC, Aug. 21–24.

[44] Kulesza, A., and Taskar, B., 2011, "k-dpps: Fixed-Size Determinantal Point Processes," Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Scheffer, T., and Getoor, L., eds. Bellevue, WA, June 28–July 2, Omnipress, pp. 1193–1200.

[45] Borodin, A., 2009, The Oxford Handbook of Random Matrix Theory, Oxford University Press, Oxford, UK.

[46] Drela, M., 1989, "Xfoil: An Analysis and Design System for Low Reynolds Number Airfoils," Low Reynolds Number Aerodynamics, T. J. Mueller, ed., Springer, New York, pp. 1–12.

[47] He, K., Zhang, X., Ren, S., and Sun, J., 2016, "Deep Residual Learning for Image Recognition," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, June 27–30, pp. 770-778.

[48] Kingma, D. P., and Ba, J., 2015, "Adam: A Method for Stochastic Optimization," 3rd International Conference on Learning Representations, San Diego, CA, May 7–9.

[49] Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A., 2018, "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules," ACS Central Sci., 4(2), pp. 268–276.