CcDPM: A Continuous Conditional Diffusion Probabilistic Model for Inverse Design

Yanxuan Zhao, Peng Zhang, Guopeng Sun, Zhigong Yang, Jianqiang Chen, Yueqing Wang

Computational Aerodynamics Institute, China Aerodynamics Research and Development Center, Mianyang, China
zzz1014751733@mail.nwpu.edu.cn, {zp14123456,gpsguopeng,yangzhigong2012,yqwang2013}@163.com, jq-chen@263.net

Abstract

Engineering design methods aim to generate new designs that meet desired performance requirements. Past work has directly introduced conditional Generative Adversarial Networks (cGANs) into this field and achieved promising results in single-point design problems (one performance requirement under one working condition). However, these methods assume that the performance requirements are distributed in categorical space, which is not reasonable in these scenarios. Although Continuous conditional GANs (CcGANs) introduce Vicinal Risk Minimization (VRM) to reduce the performance loss caused by this assumption, they still face the following challenges: 1) CcGANs can not handle multi-point design problems (multiple performance requirements under multiple working conditions). 2) Their training process is time-consuming due to the high computational complexity of the vicinal loss. To address these issues, A Continuous conditional Diffusion Probabilistic Model (CcDPM) is proposed, which the first time introduces the diffusion model into the engineering design area and VRM into the diffusion model. CcDPM adopts a novel sampling method called multi-point design sampling to deal with multi-point design problems. Moreover, the k-d tree is used in the training process of CcDPM to shorten the calculation time of vicinal loss and speed up the training process by 2-300 times in our experiments. Experiments on a synthetic problem and three real-world design problems demonstrate that CcDPM outperforms the state-of-the-art GAN models.

Introduction

Engineering design aims to generate a set of designs that meet specific performance requirements. Numerous studies have been conducted in this area over the last few years with encouraging outcomes (Buede and Miller 2016; Hubka 2015). However, these methods are time-consuming due to the following reasons: 1) these studies search for the optimal design in a large space using a trial-and-error process, 2) during the trial-and-error process, it is necessary to continuously adopt numerous physical simulations to verify if obtained designs meet the performance requirements.

To accelerate existing methods, researchers introduce the genetic algorithm, topology optimization (Duysinx and Bendsøe 1998; Bendsoe and Sigmund 2003), and adjoint optimization (Anderson and Venkatakrishnan 1999) into engineering design. However, their efforts have not fundamentally solved the time-consuming problem because the search space is still very large and the time of a single numeric simulation is still long. Furthermore, in situations where performance is evaluated by non-analytical models (*e.g.*, experiments or expert assessments), some gradient-based methods fail because they require differentiable physical solvers.

To further improve the efficiency, researchers proposed inverse design methods, which regard the performance requirements as inputs, and directly generate designs that meet specific performance requirements, to skip the trial-and-error process. Many works have directly applied cGANs (Mirza and Osindero 2014) and conditional Variational autoencoders (cVAEs) (Sohn, Lee, and Yan 2015) to the inverse design (Yilmaz and German 2020; Achour et al. 2020; Wang, Shimada, and Farimani 2021). These methods are intuitive and effective, but they have a critical limitation that they assume the empirical distributions of the performance requirements are discrete, and estimated through the Dirac delta function. This assumption is unreasonable because the distributions of performance requirements of designs are continuous in this scenario.

To reduce the performance loss caused by this discrete assumption, Continuous conditional GAN (CcGAN) (Ding et al. 2022) and Performance conditioned Diverse Generative Adversarial Network (PcDGAN) (Heyrani Nobari, Chen, and Ahmed 2021), introduce Vicinal Risk Minimization (VRM) (Sain 1996; Chapelle et al. 2000) into conditional GAN (cGAN). Although relaxing the assumption, CcGAN and PcDGAN still face some challenges. First, GANs (Goodfellow et al. 2020) adopt zero-sum game theory to train the networks, which are difficult to train and prone to mode collapse. Second, they cannot work if the input performance requirements belong to different working conditions in multi-point design problems (Piperni and Rahman 2021; Mangano and Martins 2021; Li, Bai, and Qu 2022). For example, they can only generate new aircraft designs given lift coefficient (CL) and drag coefficient (CD) under only one working condition. Third, the continuous empirical distributions of input conditions lead to the increased computational complexity of the cost function, resulting in a time-consuming training process.

Recently, a novel generative method named the diffusion probabilistic model (DPM) (Sohl-Dickstein et al. 2015; Ho, Jain, and Abbeel 2020) has been extensively studied. Compared with GANs, it has a more stable training process and can generate more realistic results. Therefore, DPM is selected as our basic model, and a novel Continuous conditional DPM is proposed, called CcDPM. CcDPM introduces the VRM into its loss function to adapt the continuity of input working conditions and performance requirements and adopts the k-dimensional tree (k-d tree) (Bentley 1975) to reduce the calculation time of vicinal loss in the training process. Moreover, CcDPM can take the performance requirements under multiple working conditions as its inputs and generate the corresponding designs.

The main contributions are summarized as follows:

(1) A new sampling method is proposed to deal with multi-point design problems, which enables diffusion models to accept any number of performance requirements under different working conditions as inputs and generate corresponding designs while training with only one condition.

(2) The k-d tree is introduced into our training process to reduce the calculation time of vicinal loss, thus greatly speeding up the training process. This method is very useful in training large-scale datasets and can be applied to the training process of CcGAN. Compared with CcDPM without a k-d tree, the training speed of CcDPM with a k-d tree on different datasets increases by 2-300 times.

(3) A novel method is proposed to solve the inverse design problem, called CcDPM. As far as we know, it is the first work to apply DPM to the inverse design area, and also the first work to introduce vicinal loss into DPM.

(4) CcDPM has been applied to several real-world inverse design problems, such as airfoil inverse design and rocket inverse design. Experimental results show that our method achieves state-of-the-art, which validates the effectiveness and advantages of the proposed CcDPM.

Background and Related Work

In this section, we provide a concise background on three topics explored in this work: inverse design, diffusion models, and vicinal risk minimization.

Engineering Design

The purpose of engineering design is to create a design with specified performance requirements under specific working conditions. For example, designing a city bus with a fuel consumption of $25L/100KM$ (performance requirement) on the urban road (working condition) is referred to as a single-point design problem. Designing a car that has a fuel consumption of $10L/100KM$ (performance requirement 1) on the urban road (working condition 1) and a fuel consumption of $5L/100KM$ (performance requirement 2) on the highways (working condition 2), respectively, is referred to as a multi-point design problem.

Inverse Design

Inverse design is a kind of engineering design method, which aims to end the trial-and-error process in traditional engineering design methods. They regard the performance requirements and working conditions as inputs, and directly generate designs that meet specific performance requirements under corresponding working conditions, so inverse design problems can be regarded as conditional generative problems. Due to the significant achievements of cGANs and cVAEs in generative tasks like image and video generation, many works have directly applied them to the inverse design (Yilmaz and German 2020; Achour et al. 2020; Wang, Shimada, and Farimani 2021).

Diffusion Probabilistic Model

Diffusion probabilistic models are a class of latent variable generative models (Sohl-Dickstein et al. 2015; Ho, Jain, and Abbeel 2020; Nichol and Dhariwal 2021) and have been used in many fields (Ramesh et al. 2022; Liu et al. 2022). They can generate data with a similar distribution as the training data. In specific, diffusion models work by destroying training data through the successive addition of Gaussian noise and then learning to restore the data by reversing the noising process. After that, the diffusion models can generate data by simply passing randomly sampled noise through the trained denoising process.

Continuous Empirical Probability Density Function

VRM is an alternative rule to empirical risk minimization (ERM). VRM assumes that a sample point shares the same label with other samples in its vicinity. The cGANs and conditional diffusion models learn the conditional distribution of samples with some auxiliary information. However, their empirical loss function fails in the continuous scenario. It is because the empirical loss function of cGANs and conditional diffusion models rely on a large number of samples for each distinct label as ERM approaches demand (Mohri, Rostamizadeh, and Talwalkar 2018; Shalev-Shwartz and Ben-David 2014), however corresponding samples for some labels may not be found when the label is continuous. To address the continuous challenge in cGAN, CcGAN (Ding et al. 2022) uses a new loss function for the discriminator, called vicinal loss, which is based on VRM.

Methodology

In this section, A novel diffusion model with vicinity-based loss, which uses a k-d tree for acceleration and a novel sampling method adapted to multi-point design is proposed. The details of each part are described in this section.

Problem Definition

Let $\boldsymbol{x} \in \mathbb{R}^d$ denotes the designs we need, $\{\boldsymbol{w}_{\boldsymbol{c}_1}, \boldsymbol{w}_{\boldsymbol{c}_2}, \cdots, \boldsymbol{w}_{\boldsymbol{c}_m}\}$ is the set of m working conditions, and $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m\}$ means the set of the performance requirements corresponding to m working conditions, where m is a positive integer, $\boldsymbol{y}_i \in \mathbb{R}^K$, and K is the dimensions of each $\boldsymbol{y}_i$. In the inverse design problem, we try to find $\boldsymbol{x}$ given $\{\boldsymbol{w}_{\boldsymbol{c}_1}, \boldsymbol{w}_{\boldsymbol{c}_2}, \cdots, \boldsymbol{w}_{\boldsymbol{c}_m}\}$ and $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m\}$ which means the new designs $\boldsymbol{x}$ should have performance $\boldsymbol{y}_1$ under $\boldsymbol{w}_{\boldsymbol{c}_1}$, performance $\boldsymbol{y}_2$

under $\boldsymbol{w}_{\boldsymbol{c}_2}$, and so on. Therefore, the problem can be regarded as solving a conditional probability distribution $q(\boldsymbol{x}| < \boldsymbol{w}_{\boldsymbol{c}_1}, \boldsymbol{y}_1 >, < \boldsymbol{w}_{\boldsymbol{c}_2}, \boldsymbol{y}_2 >, \cdots, < \boldsymbol{w}_{\boldsymbol{c}_m}, \boldsymbol{y}_m >)$.

Given the dataset $\mathcal{D} = \{(\boldsymbol{x}^1, \boldsymbol{w}_{\boldsymbol{c}}^1, \boldsymbol{y}^1), (\boldsymbol{x}^2, \boldsymbol{w}_{\boldsymbol{c}}^2, \boldsymbol{y}^2), \cdots, (\boldsymbol{x}^N, \boldsymbol{w}_{\boldsymbol{c}}^N, \boldsymbol{y}^N)\}$ (each data pair $(\boldsymbol{x}^i, \boldsymbol{w}_{\boldsymbol{c}}^i, \boldsymbol{y}^i)$ means design $\boldsymbol{x}^i$ has a performance $\boldsymbol{y}^i$ under working condition $\boldsymbol{w}_{\boldsymbol{c}}^i$), our target is to learn a distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}| < \boldsymbol{w}_{\boldsymbol{c}_1}, \boldsymbol{y}_1 >, < \boldsymbol{w}_{\boldsymbol{c}_2}, \boldsymbol{y}_2 >, \cdots, < \boldsymbol{w}_{\boldsymbol{c}_m}, \boldsymbol{y}_m >)$ from $\mathcal{D}$ that is as close as possible to the approximate $q(\boldsymbol{x}| < \boldsymbol{w}_{\boldsymbol{c}_1}, \boldsymbol{y}_1 >, < \boldsymbol{w}_{\boldsymbol{c}_2}, \boldsymbol{y}_2 >, \cdots, < \boldsymbol{w}_{\boldsymbol{c}_m}, \boldsymbol{y}_m >)$ with changeable m in one model.

Continuous Conditional Diffusion Probabilistic Model

The diffusion probabilistic model has achieved great success in many generative tasks (Dhariwal and Nichol 2021; Gong et al. 2022; Li et al. 2022). Compared with GANs, its training process is more stable, and it can generate more realistic samples. In our work, a novel diffusion model called the Continuous conditional Diffusion Probabilistic Model (CcDPM) is proposed. CcDPM introduces the vicinal loss into its loss function. To reduce the calculation time of vicinal loss, the k-d tree is adopted in our training process. Moreover, CcDPM can take performance requirements under multiple working conditions as its inputs and generate the corresponding designs. In this section, the loss function, the training process, and the testing process will be introduced respectively.

Loss Function of CcDPM As mentioned before, the traditional GANs and DPMs commonly assume that the empirical distributions of conditions are discrete and estimate the empirical distributions through the Dirac delta function. This assumption is unreasonable in the inverse design area because the distributions of performance requirements are continuous. Motivated by CcGAN, the vicinal loss is introduced into the DPM. The vicinal loss is essentially a smooth estimation of the marginal distribution $p(\boldsymbol{w}_{\boldsymbol{c}}, \boldsymbol{y})$ (only smooth $p(\boldsymbol{y})$ is also reasonable) by the Gaussian kernel density estimation (KDE) (Davis, Lii, and Politis 2011; Hastie et al. 2009). The empirical probability density function $p(\boldsymbol{x}, \boldsymbol{u})$ of ERM is shown in Eq.1.

$$\hat{p}^{\delta}(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}^i)\delta(\boldsymbol{u} - \boldsymbol{u}^j) \tag{1}$$

where $\boldsymbol{u}$ denotes the combination of $\boldsymbol{w}_{\boldsymbol{c}}$ and $\boldsymbol{y}$, N denotes the number of samples in the dataset, $\boldsymbol{x}^i$ is the real sample in the training dataset, $\boldsymbol{u}^i$ is the condition-performance pair in the training dataset, and δ represents the Dirac function centered at 0. Therefore, the $\hat{p}^{\delta}(\boldsymbol{x}, \boldsymbol{u})$ is equal to zero when $\boldsymbol{u}$ is not in the condition-performance collection in the dataset. In CcDPM, Gaussian kernel KDE is used to replace the Dirac delta estimation and to smooth the marginal distribution $p(\boldsymbol{u})$. The new estimate for $p(\boldsymbol{x}, \boldsymbol{u})$ is termed the soft vicinal estimate (SVE) as Eq.2.

Algorithm 1: Training Algorithm of CcDPM

Input: The dataset $\boldsymbol{\mathcal{D}} = \{(\boldsymbol{x}^1, \boldsymbol{w}_{\boldsymbol{c}}^1, \boldsymbol{y}^1), \cdots, (\boldsymbol{x}^N, \boldsymbol{w}_{\boldsymbol{c}}^N, \boldsymbol{y}^N)\}$, and its corresponding condition-performance set $\boldsymbol{\mathcal{C}} = \{(\boldsymbol{w}_{\boldsymbol{c}}^1, \boldsymbol{y}^1), \cdots, (\boldsymbol{w}_{\boldsymbol{c}}^N, \boldsymbol{y}^N)\}$
Output: The trained weights $\boldsymbol{\theta}$.

```
 1: Build a k-d tree $\boldsymbol{\mathcal{K}}$ of $\boldsymbol{\mathcal{D}}$ using $\boldsymbol{\mathcal{C}}$.
 2: while $step < total_{epoch}$ do
 3:     Initialize $t \sim Uniform(1, \cdots, T)$
 4:     Randomly sample $\boldsymbol{u}^i = (\boldsymbol{w}_{\boldsymbol{c}}^i, \boldsymbol{y}^i)$ from space of $\boldsymbol{\mathcal{C}}$.
 5:     Add Gauss noise to $\boldsymbol{u}^i$, and get $\hat{\boldsymbol{u}}^i = (\boldsymbol{u}^i + \boldsymbol{\varepsilon}')$.
 6:     Find $k$ nearest neighbors of $\hat{\boldsymbol{u}}^i$ from $\boldsymbol{\mathcal{K}}$ in a distance
        threshold of $\kappa$, and obtain the corresponding subset
        $\boldsymbol{\mathcal{D}}_i = \{(\boldsymbol{x}^{i_1}, \boldsymbol{w}_{\boldsymbol{c}}^{i_1}, \boldsymbol{y}^{i_1}), \cdots, (\boldsymbol{x}^{i_k}, \boldsymbol{w}_{\boldsymbol{c}}^{i_k}, \boldsymbol{y}^{i_k})\}$ from
        $\boldsymbol{\mathcal{D}}$.
 7:     Randomly choose a sample $\boldsymbol{\mathcal{D}}_i^j = (\boldsymbol{x}^{i_j}, \boldsymbol{w}_{\boldsymbol{c}}^{i_j}, \boldsymbol{y}^{i_j})$
        from $\boldsymbol{\mathcal{D}}_i$, $\boldsymbol{u}^{i_j} = (\boldsymbol{w}_{\boldsymbol{c}}^{i_j}, \boldsymbol{y}^{i_j})$.
 8:     Calculate $W$ using $\hat{\boldsymbol{u}}^i$ and $\boldsymbol{u}^{i_j}$ by Eq.3.
 9:     Use $(\boldsymbol{x}^{i_j}, \hat{\boldsymbol{u}}^i)$ and $t$ to calculate $\hat{\mathcal{L}}_{simple}^{SVL}(\boldsymbol{\theta})$ by Eq.4
10:     Compute the gradients and update $\boldsymbol{\theta}$.
11: end while
12: return $\boldsymbol{\theta}$.
```

$$\hat{p}^{SVE}(\boldsymbol{x}, \boldsymbol{u}) = \frac{C}{N} \left[\sum_{i=1}^{N} \exp^{(-\frac{1}{2}(\boldsymbol{u}-\boldsymbol{u}^i)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{u}-\boldsymbol{u}^i))}\right] \left[\frac{\sum_{j=1}^{N} W(\boldsymbol{u}^j, \boldsymbol{u})\delta(\boldsymbol{x}-\boldsymbol{x}^j)}{\sum_{k=1}^{N} W(\boldsymbol{u}^k, \boldsymbol{u})}\right] \tag{2}$$

$$W(\boldsymbol{u}^i, \boldsymbol{u}) = e^{-\nu||\boldsymbol{u}^i - \boldsymbol{u}||^2} \tag{3}$$

where ν is a Hyper-parameter, C is constant. Through Eq.2 a novel loss is used in CcDPM by using VRM called vicinal loss in Eq.4. In Eq.4 $\boldsymbol{\varepsilon}$ denotes the noise in difussion model, $\boldsymbol{\varepsilon}'$ denotes the noise in Algorithm 1.

$$\hat{L}_{simple}^{SVL}(\boldsymbol{\theta}) = \frac{C}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} E_{\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}'}\left[\frac{W(\boldsymbol{u}^i, \boldsymbol{u}^j + \boldsymbol{\varepsilon}')}{\sum_{k=1}^{N} W(\boldsymbol{u}^k, \boldsymbol{u}^j + \boldsymbol{\varepsilon}')} ||\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{u}^j + \boldsymbol{\varepsilon}', t)||^2\right] \tag{4}$$

Training Process Algorithm 1 shows our training algorithm. In the calculation of the loss function, k neighbors in a distance threshold of κ near the specified $\hat{\boldsymbol{u}}^i$ will be searched on the whole dataset. Consequently, the search complexity is $\mathbb{O}(N)$, where N is the number of training samples. When the training dataset is very large, the search time will be very long. Therefore, the k-d tree (Bentley 1975) is adopted to accelerate this process. The k-d tree is a data structure used for searching neighbors, and its average search complexity is $\mathbb{O}(\log n)$. Therefore, a k-d tree is constructed using $\boldsymbol{\mathcal{C}}$ at the beginning of the training process. Then $\boldsymbol{\theta}$ can be updated iteratively. In each training step, first randomly sample in

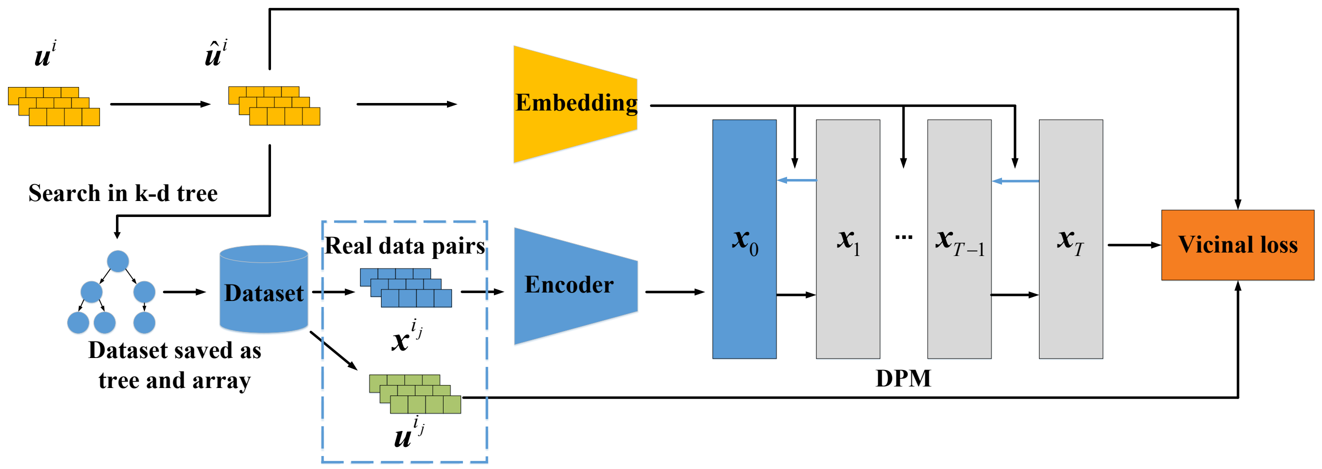

Figure 1: Training process of CcDPM.

space $\boldsymbol{C}$ to get a working condition $\boldsymbol{w}_{\boldsymbol{c}}^i$ and its corresponding performance requirements $\boldsymbol{y}^i$. Next, Gaussian noise is added to them to get $\hat{\boldsymbol{u}}^i$, and then use the k-d tree to find k nearest neighbors in a distance threshold of κ. After that, the loss function can be computed by using $\hat{\boldsymbol{u}}^i$ and one of its neighbors. Finally, the gradient will be calculated and θ can be updated.

Multi-Point Design Sampling After the training process, the model should be applied to generation problems. In real scenarios, the engineering design problem can be a multi-point design problem (Piperni and Rahman 2021; Mangano and Martins 2021; Li, Bai, and Qu 2022), which means the generated designs should have different specific performances under different working conditions. For example, we want the generated designs $\boldsymbol{x}$ concurrently satisfies both $\boldsymbol{y}_1$ under working condition $\boldsymbol{w}_{\boldsymbol{c}_1}$ and $\boldsymbol{y}_2$ under working condition $\boldsymbol{w}_{\boldsymbol{c}_2}$. CcGAN, PcDGAN, and DPM cannot handle the generative problems under multiple working conditions.

In our method, $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{w}_{\boldsymbol{c}}, \boldsymbol{y})$ can be obtained from the trained CcDPM, which means CcDPM only accepts one performance requirement under one working condition as its input. To enable it to handle situations with multiple input conditions, the sampling algorithm should be changed, as shown in Fig. 2 and Algorithm 2.

We define the Markovian noising process $\hat{q}$ conditioned on multiple performance requirements under multiple working conditions, which is similar to the unconditional Markovian noising process q (Dhariwal and Nichol 2021). The conditional reverse process can be expressed as Eq.5.

$$\begin{aligned}\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, <\boldsymbol{w}_{\boldsymbol{c}_1}, \boldsymbol{y}_1>, \cdots, <\boldsymbol{w}_{\boldsymbol{c}_m}, \boldsymbol{y}_m>) = \\ Zq(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})\hat{q}(<\boldsymbol{w}_{\boldsymbol{c}_1}, \boldsymbol{y}_1>, \cdots, <\boldsymbol{w}_{\boldsymbol{c}_m}, \boldsymbol{y}_m>|\boldsymbol{x}_t)\end{aligned} \tag{5}$$

where Z is a normalizing constant, $\hat{q}(<\boldsymbol{w}_{\boldsymbol{c}_1}, \boldsymbol{y}_1>, \cdots, <\boldsymbol{w}_{\boldsymbol{c}_m}, \boldsymbol{y}_m>|\boldsymbol{x}_t)$ can be calculated by estimator $p_{\boldsymbol{\varphi}}(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{w}_{\boldsymbol{c}})$. Working condition is known during generation and working condition $\boldsymbol{w}_{\boldsymbol{c}}$ and design $\boldsymbol{x}$ are independent, and assuming that the performance of the same design is independent under different working conditions, with

Algorithm 2: Sampling Algorithm of CcDPM

Input: The trained weights $\boldsymbol{\theta}$, the working conditions and performance requirements set $\{(\boldsymbol{w}_{\boldsymbol{c}_1}, \boldsymbol{y}_1), \cdots, (\boldsymbol{w}_{\boldsymbol{c}_m}, \boldsymbol{y}_m)\}$, and denote $(\boldsymbol{w}_{\boldsymbol{c}_i}, \boldsymbol{y}_i)$ to $\boldsymbol{u}_i$.
Output: New design $\boldsymbol{x}$.

```
1: $t = T$
2: **while** $t > 0$ **do**
3:     Obtain $\boldsymbol{\varepsilon_\theta}(\boldsymbol{x}_t)$ using $\boldsymbol{x}_t$ and $\boldsymbol{\theta}$.
4:     for j = 1 to $m$ do
5:         Obtain $\boldsymbol{\varepsilon_\theta}(\boldsymbol{x}_t, \boldsymbol{u}_j)$ using $\boldsymbol{x}_t$, $\boldsymbol{u}_j$, $\boldsymbol{\theta}$, and $t$.
6:     Compute $\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_m)$ using $\boldsymbol{\varepsilon_\theta}(\boldsymbol{x}_t)$,
       $\boldsymbol{\varepsilon_\theta}(\boldsymbol{x}_t, \boldsymbol{u}_1), \cdots, \boldsymbol{\varepsilon_\theta}(\boldsymbol{x}_t, \boldsymbol{u}_m)$ by Eq.12
7:     Using $\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_m)$ sampling $\boldsymbol{x}_{t-1}$ using
       DDPM or another sampler such as DDIM.
8:     $t = t - 1$
9: **end while**
10: $\boldsymbol{x} = \boldsymbol{x}_0$
11: **return** new design $\boldsymbol{x}$.
```

Bayesian Formula Eq.6 can be obtained, more detailed proof is provided in Appendix B.

$$\hat{q}(\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_m|\boldsymbol{x}_t) \propto \prod_{i=1}^{m} p_{\boldsymbol{\varphi}}(\boldsymbol{y}_i|\boldsymbol{x}_t, \boldsymbol{w}_{\boldsymbol{c}_i}) \tag{6}$$

where $p_{\boldsymbol{\varphi}}$ is an estimator. Then the sampling method used to inverse design for multi-point design is obtained. The sampling method with estimator is described as Eq.7.

$$\hat{\boldsymbol{p}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, \boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m) = \mathcal{N}(\hat{\boldsymbol{\mu}}_{\boldsymbol{\theta}} + \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\boldsymbol{g}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}) \tag{7}$$

where $\boldsymbol{g}$ and $\hat{\boldsymbol{\mu}}_{\boldsymbol{\theta}}$ are described in Eq.8 and Eq.9.

$$\boldsymbol{g} = \nabla_{\boldsymbol{x}_t} \log \prod_{i=1}^{m} p_{\boldsymbol{\varphi}}(\boldsymbol{y}_i|\boldsymbol{x}_t, \boldsymbol{w}_{\boldsymbol{c}_i})|_{\boldsymbol{x}_t=\boldsymbol{\mu}} \tag{8}$$

$$\begin{aligned}\hat{\boldsymbol{\mu}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m) = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t) \\ +\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t) \sum_{i=1}^{m} s_i \nabla_{\boldsymbol{x}_t} \log p_{\boldsymbol{\varphi}}(\boldsymbol{y}_i|\boldsymbol{x}_t, \boldsymbol{w}_{\boldsymbol{c}_i})\end{aligned} \tag{9}$$

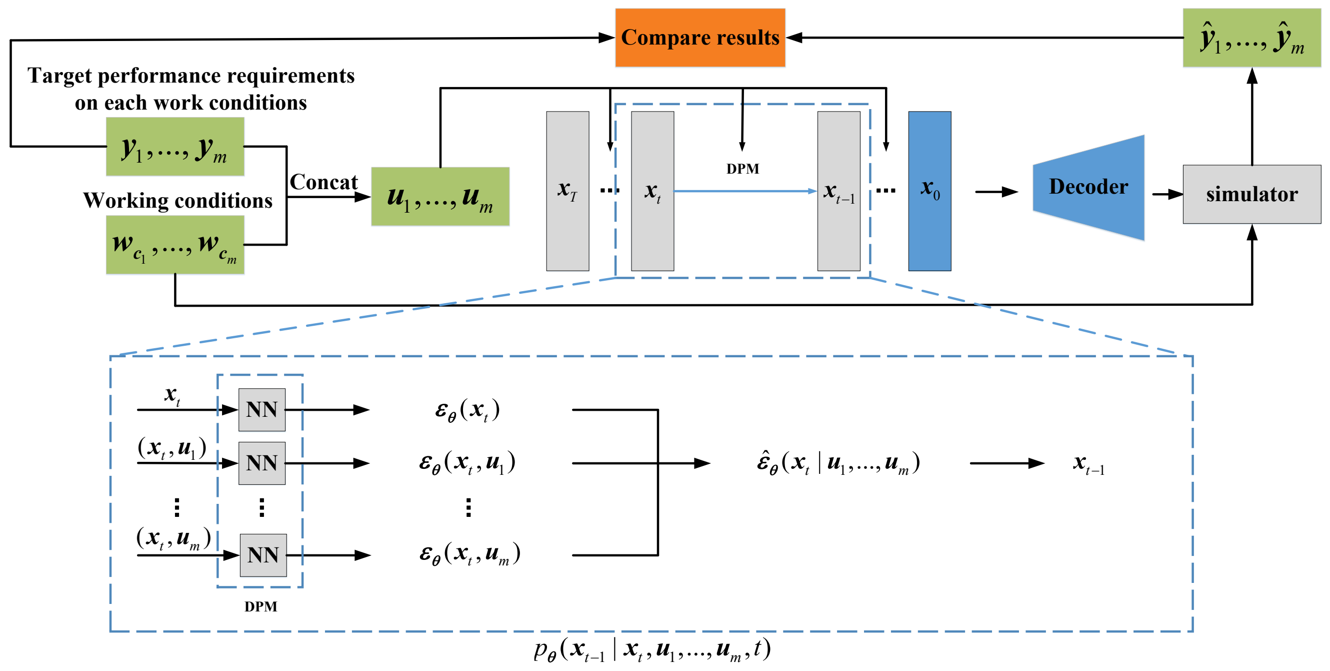
Compare results
$\hat{y}_1,\dots,\hat{y}_m$
Target performance requirements on each work conditions
$y_1,\dots,y_m$
Concat
$u_1,\dots,u_m$
Working conditions
$w_{c_1},\dots,w_{c_m}$
x_T
x_t
DPM
x_{t-1}
x_0
Decoder
simulator
x_t
(x_t,u_1)
(x_t,u_m)
NN
DPM
$\varepsilon_\theta(x_t)$
$\varepsilon_\theta(x_t,u_1)$
$\varepsilon_\theta(x_t,u_m)$
$\hat{\varepsilon}_\theta(x_t \mid u_1,\dots,u_m)$
x_{t-1}
$p_\theta(x_{t-1} \mid x_t,u_1,\dots,u_m,t)$

Figure 2: Sampling method of CcDPM.

In Eq.9, the $p_{\boldsymbol{\varphi}}(\boldsymbol{y}_i|\boldsymbol{x}_t,\boldsymbol{w}_{\boldsymbol{c}_i})$ is difficult to compute, and it requires another model to learn. In our CcDPM, the estimator free sampling method (Ho and Salimans 2022) is used. The implicit estimator is given in Eq.10.

$$p^i(\boldsymbol{y}|\boldsymbol{x}_t,\boldsymbol{w}_{\boldsymbol{c}}) \propto \frac{p(\boldsymbol{x}_t|\boldsymbol{y},\boldsymbol{w}_{\boldsymbol{c}})}{p(\boldsymbol{x}_t)} \tag{10}$$

Compute the gradient terms of both sides in Eq.10, then Eq.11 can be obtained.

$$\begin{gathered}\nabla_{\boldsymbol{x}_t}\log p^i(\boldsymbol{y}_i|\boldsymbol{x}_t,\boldsymbol{w}_{\boldsymbol{c}_i}) \propto \\ \nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t|\boldsymbol{y}_i,\boldsymbol{w}_{\boldsymbol{c}_i}) - \nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t) \propto \\ \boldsymbol{\varepsilon}(\boldsymbol{x}_t|\boldsymbol{y}_i,\boldsymbol{w}_{\boldsymbol{c}_i}) - \boldsymbol{\varepsilon}(\boldsymbol{x}_t)\end{gathered} \tag{11}$$

Then the sampling method of CcDPM is shown in Eq.12.

$$\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{u}_1,...,\boldsymbol{u}_m) = \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t) + \sum_{i=1}^{m} s_i(\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,\boldsymbol{u}_i) - \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t)) \tag{12}$$

where $\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t)$ is the unconditional output of the model, m is the number of specific working conditions, and s_i is the guiding scaling of specific performance under specific working conditions. After obtaining $\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,\boldsymbol{u}_1,\cdots,\boldsymbol{u}_m)$, DDPM (Ho, Jain, and Abbeel 2020) (or another sampler such as DDIM (Song, Meng, and Ermon 2020)) can be used to sample $\boldsymbol{x}_{t-1}$.

Experiments

In this section, we introduce one synthetic dataset and three real-world datasets and report the results that CcDPM applied to them.

Datasets

Four datasets are used in our experiments to evaluate our CcDPM.

- **Synthetic Uneven Dataset:** This dataset is an uneven synthetic example (Heyrani Nobari, Chen, and Ahmed 2021). There are six peaks in the performance space. 50% of the data are located in a circle around one of the peaks, the remaining 50% of the data is evenly distributed throughout the space. 10,000 data points are used for training. The performance metrics as a density function of an un-normalized Gaussian mixture:

$$\boldsymbol{y}(\boldsymbol{x}) = \sum_{k=1}^{K} \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{\mu_k})^T(\boldsymbol{x}-\boldsymbol{\mu_k})}{2\sigma^2}\right) \tag{13}$$

  where $\boldsymbol{\mu}_k$ is the mode of the k-th mixture component and σ is the standard deviation.
- **Xfoil Dataset:** This dataset is an airfoil dataset (Heyrani Nobari, Chen, and Ahmed 2021) under single working condition. Based on the UIUC airfoil library, 48,503 airfoils are generated by BézierGAN (Chen, Chiu, and Fuge 2019), and calculated by Xfoil (Drela 1989) at Mach number 0.01 and angle of attack 0 degrees, and the lift-drag ratio CL/CD is used as performance. Since there is only one working condition, the working condition is not included in the calculation in each model. Performance metrics $\boldsymbol{y} = [CL/CD]$, designs $\boldsymbol{x} =$ airfoil shape parameters.
- **CFD Dataset:** This dataset is an airfoil dataset have multiple working conditions. Different from the Xfoil dataset, this dataset used a CFD solver which is more

model	single-point design				multi-point design	
	Uneven	Xfoil	CFD	Rocket	CFD	Rocket
CcGAN	0.0256	0.0663	0.0425	0.0732	-	-
PcDGAN	0.0186	0.0515	0.0756	0.0738	-	-
CcLSGAN	0.0257	0.1220	0.0411	0.0660	-	-
DPM	0.0174	0.0501	0.0411	0.0442	0.1010	0.0891
CcDPM	**0.0148**	**0.0459**	**0.0376**	**0.0425**	**0.0986**	**0.0871**

Table 1: The label error of single-point design and multi-point design.

model	single-point design				multi-point design	
	Uneven	Xfoil	CFD	Rocket	CFD	Rocket
CcGAN	-85.55	-31.40	-16.37	-29.77	-	-
PcDGAN	-87.75	-29.76	**-5.94**	-26.96	-	-
CcLSGAN	-84.60	-31.28	-13.64	-29.79	-	-
DPM	-64.78	-29.78	-11.21	-27.05	-10.77	-32.67
CcDPM	**-64.63**	**-29.28**	-12.42	**-26.70**	**-10.50**	**-31.87**

Table 2: The diversity comparison of single-point design and multi-point design.

complex and close to reality (Wang et al. 2021). 10,366 airfoils are obtained by using the UIUC airfoil library after CST perturbation (Nadarajah, Castonguay, and Mousavi 2007; Kulfan 2008). The working conditions are the Mach number is from 0.2 to 0.7 and the angle of attack is from 2 to 10, thus working condition $\boldsymbol{w_c} = [Ma, AOA]^T$, totaling 31098 data pairs. Using lift coefficient as performance metrics, thus $\boldsymbol{y} = [CL]$, $\boldsymbol{x} =$ airfoil shape parameters.

- **Rocket Dataset:** This dataset is a rocket dataset have multiple working conditions. 3311 rocket shapes are calculated with simulation software at Mach number from 0.6 to 1.6, roll angle from 2 to 20 degrees, and attack angle from 2 to 20 degrees, thus working condition $\boldsymbol{w_c} = [Ma, RA, AOA]^T$, totaling 3,311,000 data pairs. Using the axial force coefficient, the normal force coefficient, and the pressure center as performance metrics, thus $\boldsymbol{y} = [CA, CN, Xp]^T$, $\boldsymbol{x} =$ rocket shape parameters.

Only the CFD dataset and Rocket dataset are used for generating multi-point design tasks (multiple performance requirements under corresponding working conditions) because the Uneven dataset and Xfoil dataset have only one working condition.

Metrics

In our work, the performance of models is evaluated by how well the generated designs meet the given performance requirements and how diverse the designs are. Mean absolute error (MAE) between the performance requirements and the real performance of generated designs is used to evaluate how well the generated designs meet the given performance requirements, named "label error". To measure diversity of designs, the log determinant of the similarity matrix (Borodin 2009) is used in our experiments.

$$s_{div} = \frac{1}{n} \sum_{i=1}^{n} \log \det(\boldsymbol{L}_{s_i}) \tag{14}$$

where n is the number of times diversity is evaluated, S_i is a random subset of generated designs and L is the similarity kernel.

Results

The generated designs of all models are calculated by the corresponding simulation solver to avoid invalid generation. In our experiments, CcGAN does not use the embedding proposed in the original paper (Ding et al. 2022), which will lead to invalid generation in these scenarios. CcLSGAN is the LSGAN (Mao et al. 2017) with vicinal loss. Hyperparameters and more detailed results are provided in Appendix C and Appendix D.

The performance comparison of single-point design and multi-point design is shown in Table 1. In multi-point design, working conditions are set to a normalized distance of 0.5 or more. From Table 1, it can be seen that: 1) compared with GANs and diffusion models, the diffusion models can generate designs that meet design requirements better in all datasets, 2) diffusion model with VRM outperforms diffusion model with ERM in all datasets.

The diversity comparison of single-point design and multi-point design is shown in Table 2. It is difficult for DPM and CcDPM to use Bézier parameterization, so the CST parameterization is used, which makes the comparison of diversity calculated in CST space in case Xfoil. The calculation of diversity only considers valid samples. From Table 2, it can be seen that diffusion models can generate more diverse designs than GANs. Although PcDGAN has larger diversity in case CFD, its label error is higher and will generate more invalid designs.

model	Uneven	Xfoil	CFD	Rocket
Without k-d tree	1h24min	13h30min	151h	530h
With k-d tree	**31min**	**34min**	**47min**	**1h30min**
Speedup Ratio	2.7	23	192	353

Table 3: The speed comparison of CcDPM with and without the k-d tree.

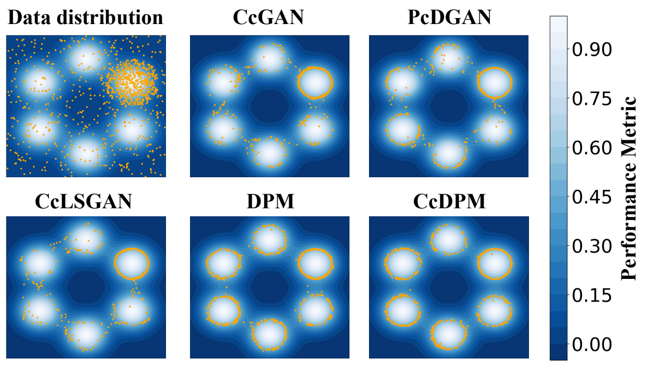

Figure 3: Data distribution (only a sub-sample of 1,000 of the 10,000 data points is shown for clarity) and comparison of the output distribution of 1,000 generated data points with the input performance requirement of 0.5, shown by orange dots.

Table 3 shows the speed comparison of CcDPM with and without the k-d tree. For the speed test, we train the models for 200,000 training steps with a batch size of 256. From Table 2, it can be seen that as the size of the dataset increases, the acceleration effect of the k-d tree becomes more apparent, making it possible to calculate neighborhood loss on large datasets. This k-d tree strategy can also be applied to any other models with vicinal loss such as CcGAN, PcDGAN and has a similar effect.

Fig. 3 shows the distribution of the Uneven dataset and the conditional generation results of the Uneven dataset, with a performance requirement set to 0.5. From Fig. 3, it can be seen that the generation results of CcGAN and CcLSGAN are not uniform enough. As the loss function adopts Determinantal Point Process (DPP) (Borodin 2009), resulting in the generated points of PcDGAN tend to be far away from each other and leave blank space near the coordinate origin. DPM and CcDPM generate better results, while the results of CcDPM are more accurate.

Fig.4 shows the generated designs of the CFD case with normalized performance requirement range from 0.29 to 0.96 under working condition $\boldsymbol{w_c} = [Ma, AOA]^T = [0.3, 8.0]^T$. The performance requirement in this case means lift coefficient of airfoil. Generally speaking, the larger the curvature of the trailing edge of an airfoil, the greater the lift coefficient of the airfoil. All models have captured this feature. Although PcDGAN has higher diversity, it generates more invalid designs (0.49,0.56 in Fig.4).

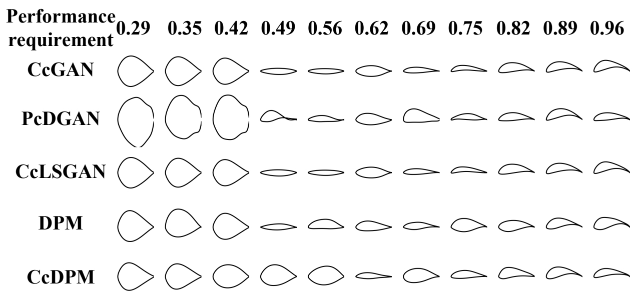

Figure 4: Generated designs of the CFD case with performance requirement in single-point design.

Conclusion and Limitations

In this paper, we introduce diffusion models to the field of engineering inverse design for the first time. Specifically, we propose a novel diffusion model called CcDPM by incorporating VRM into the conditional diffusion model. Through experiments with both synthesized data and real engineering design applications, we demonstrate that CcDPM outperforms the current state-of-the-art approach in tackling conditional generation problems under continuous conditions. Moreover, we accelerate the calculation of vicinal loss by utilizing k-d tree, which reduces the computational complexity from O(n) to O(log n) and makes vicinal loss applicable to large-scale datasets. Additionally, we propose a novel sampling method in the diffusion model to solve multi-point inverse engineering design problems. While we apply this sampling method to DPM and CcDPM, it can be implemented in other diffusion models as well. Overall, this work presents a significant contribution to the field of engineering inverse design by introducing and improving diffusion models. The proposed CcDPM and the accelerated calculation and sampling methods offer promising solutions to complex conditional generation problems and pave the way for further research in this area.

Limitations The diffusion model is difficult to use Bézier parameterization because the same airfoil shape can be represented by different Bézier parameters, which makes it difficult to directly get Bézier parameters from existing airfoil shapes, while GAN can use Bézier parameterization only in the generator part, which avoids the conversion of airfoil shape to Bézier parameters. Besides, the method in this paper is to use a latent space diffusion model to generate the design's encoding, and then restore the design based on the decoder. This method cannot solve inverse design problems where designs cannot be encoded/parameterized such as topology generation.

Ethics Statement

This paper proposes a universal technique for shape inverse design which is not designed for weapon systems. The dataset we use are suitable for the design of airliner airfoils and commercial recyclable rocket shapes. There is a risk of misuse of this technique, and we do not agree to apply it to weapon system design or any designs that may “directly facilitate injury to living beings”.

Acknowledgements

Thank to my sensible friends Yifei Chen and Tong Li, who have made indelible contributions to this paper. This work was supported by NSAF, China (Grant No. U2230208) and the National Numerical Windtunnel Project, China.

References

Achour, G.; Sung, W. J.; Pinon-Fischer, O. J.; and Mavris, D. N. 2020. Development of a conditional generative adversarial network for airfoil shape optimization. In *AIAA Scitech 2020 Forum*, 2261.

Anderson, W. K.; and Venkatakrishnan, V. 1999. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers & Fluids*, 28(4-5): 443–480.

Bendsoe, M. P.; and Sigmund, O. 2003. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media.

Bentley, J. L. 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9): 509–517.

Borodin, A. 2009. Determinantal point processes. *arXiv preprint arXiv:0911.1153*.

Buede, D. M.; and Miller, W. D. 2016. The engineering design of systems: models and methods.

Chapelle, O.; Weston, J.; Bottou, L.; and Vapnik, V. 2000. Vicinal risk minimization. *Advances in neural information processing systems*, 13.

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

Davis, R. A.; Lii, K.-S.; and Politis, D. N. 2011. Remarks on some nonparametric estimates of a density function. *Selected Works of Murray Rosenblatt*, 95–100.

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

Ding, X.; Wang, Y.; Xu, Z.; Welch, W. J.; and Wang, Z. J. 2022. Continuous conditional generative adversarial networks: Novel empirical losses and label input mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Drela, M. 1989. XFOIL: An analysis and design system for low Reynolds number airfoils. In *Low Reynolds Number Aerodynamics: Proceedings of the Conference Notre Dame, Indiana, USA, 5–7 June 1989*, 1–12. Springer.

Duysinx, P.; and Bendsøe, M. P. 1998. Topology optimization of continuum structures with local stress constraints. *International journal for numerical methods in engineering*, 43(8): 1453–1478.

Gong, S.; Li, M.; Feng, J.; Wu, Z.; and Kong, L. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144.

Hastie, T.; Tibshirani, R.; Friedman, J. H.; and Friedman, J. H. 2009. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Heyrani Nobari, A.; Chen, W.; and Ahmed, F. 2021. Pcdgan: A continuous conditional diverse generative adversarial network for inverse design. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 606–616.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33: 6840–6851.

Ho, J.; and Salimans, T. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Hubka, V. 2015. *Principles of engineering design*. Elsevier.

Kulfan, B. M. 2008. Universal parametric geometry representation method. *Journal of aircraft*, 45(1): 142–158.

Li, L.; Bai, J.; and Qu, F. 2022. Multipoint aerodynamic shape optimization of a truss-braced-wing aircraft. *Journal of Aircraft*, 59(5): 1179–1194.

Li, X.; Thickstun, J.; Gulrajani, I.; Liang, P. S.; and Hashimoto, T. B. 2022. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35: 4328–4343.

Liu, J.; Li, C.; Ren, Y.; Chen, F.; and Zhao, Z. 2022. Diffsinger: Singing voice synthesis via shallow diffusion mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 11020–11028.

Mangano, M.; and Martins, J. R. 2021. Multipoint aerodynamic shape optimization for subsonic and supersonic regimes. *Journal of Aircraft*, 58(3): 650–662.

Mao, X.; Li, Q.; Xie, H.; Lau, R. Y.; Wang, Z.; and Paul Smolley, S. 2017. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, 2794–2802.

Mirza, M.; and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Mohri, M.; Rostamizadeh, A.; and Talwalkar, A. 2018. *Foundations of machine learning*. MIT press.

Nadarajah, S.; Castonguay, P.; and Mousavi, A. 2007. Survey of shape parameterization techniques and its effect on three-dimensional aerodynamic shape optimization. In *18th AIAA computational fluid dynamics conference*, 3837.

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

Piperni, P.; and Rahman, S. M. 2021. Singlepoint and Multipoint Robust Design of Airfoils using CST Functions. In *AIAA Scitech 2021 Forum*, 1359.

Ramesh, A.; Dhariwal, P.; Nichol, A.; Chu, C.; and Chen, M. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Sain, S. R. 1996. The nature of statistical learning theory.

Shalev-Shwartz, S.; and Ben-David, S. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2256–2265. PMLR.

Sohn, K.; Lee, H.; and Yan, X. 2015. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28.

Song, J.; Meng, C.; and Ermon, S. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.

Wang, Y.; Deng, L.; Wan, Y.; Yang, Z.; Yang, W.; Chen, C.; Zhao, D.; Wang, F.; and Guo, Y. 2021. An Intelligent Method for Predicting the Pressure Coefficient Curve of Airfoil-Based Conditional Generative Adversarial Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15.

Wang, Y.; Shimada, K.; and Farimani, A. B. 2021. Airfoil gan: Encoding and synthesizing airfoils foraerodynamic-aware shape optimization. *arXiv preprint arXiv:2101.04757*.

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