Inverse Design of 2D Airfoils using Conditional Generative Models and Surrogate Log-Likelihoods

Qiuyi Chen[1]*, Jun Wang[1], Phillip Pope[2], Wei (Wayne) Chen[3], Mark Fuge[1]
[1]Department of Mechanical Engineering, University of Maryland, College Park, Maryland 20742
[2]Department of Computer Science, University of Maryland, College Park, Maryland 20742
[3]Department of Mechanical Engineering, Northwestern University, Evanston, IL 60201

*This paper shows how to use conditional generative models in 2D airfoil optimization to probabilistically predict good initialization points within the vicinity of the optima given the input boundary conditions, thus warm starting and accelerating further optimization. We accommodate the possibility of multiple optimal designs corresponding to the same input boundary condition and take this inversion ambiguity into account when designing our prediction framework. To this end, we first employ the conditional formulation of our previous work BézierGAN—Conditional BézierGAN (CBGAN)—as a baseline, then introduce its sibling conditional entropic BézierGAN (CEBGAN), which is based on optimal transport regularized with entropy. Compared with CBGAN, CEBGAN overcomes mode collapse plaguing conventional GANs, improves the average lift-drag (C_l/C_d) efficiency of airfoil predictions from 80.8% of the optimal value to 95.8%, and meanwhile accelerates the training process by 30.7%. Furthermore, we investigate the unique ability of CEBGAN to produce a log-likelihood lower bound that may help select generated samples of higher performance (*e.g.*, aerodynamic performance). In addition, we provide insights into the performance differences between these two models with low-dimensional toy problems and visualizations. These results and the probabilistic formulation of this inverse problem justify the extension of our GAN-based inverse design paradigm to other inverse design problems or broader inverse problems.*

1 INTRODUCTION

Design synthesis with shape optimization is often time-consuming. After setting up a forward (analysis) model of the objective function under a set of boundary conditions or requirements, you have to specify an initial set of design variables and embark on some (typically) wearisome and costly iterative algorithm to minimize that objective. This swings to its extreme when the analysis model is formulated as nonlinear differential equations that need to be solved iteratively, as, in the case of airfoil design, the Navier-Stokes equation governs the system. Optimizing a design for a single set of boundary conditions can take days or weeks, depending on the number of design variables and complexity of the objective function or forward model.

Therefore, it is tempting to construct a mapping that leads us directly from the input problem parameters—*e.g.*, boundary conditions, constraints, or other problem-specific requirements—to the optimal design variables. Alternatively, to be more pragmatic, at least to the vicinity of the optima to short-circuit or accelerate a subsequent optimization process. Researchers refer to such mappings as *Inverse Design*.

The quest for such a mapping induces several challenges. Apart from seeking a model with enough complexity and regularity to approximate this complicated mapping with precision, another inevitable conundrum is the *inversion ambiguity* that inverse design problems usually confront. This is when there are multiple potential, near-optimal designs corresponding to the same input condition. This stymies traditional bijective regression models.

In this work, we attempt to address these challenges in the context of 2D airfoil optimization, in which, given the input boundary conditions, we predict the corresponding airfoils with near-maximal lift-drag (C_l/C_d) efficiency. Specifically, after casting this inverse design problem from a probabilistic perspective, we introduce two recently developed probabilistic generative models—conditional BézierGAN (CBGAN) and conditional entropic BézierGAN (CEBGAN)—to realize airfoil inverse design in a probabilistic and data-driven manner. Our result shows that CBGAN and CEBGAN can respectively produce airfoils with 80.8% and 95.8% of the average optimal airfoil performance. Their predictions can then serve as warm start initialization points to accelerate further optimization. This paper's key contributions are as follows:

1. Our conditional GANs (*i.e.*, CBGAN and CEBGAN) take the freestream conditions and the target property, including the Mach number, Reynolds number, and the

*corresponding author. e-mail: qchen88@umd.edu

Copyright © by ASME

target lift coefficient as input, then directly predict airfoils of near-optimal C_l/C_d efficiency (80.8% and 95.8% of optimal C_l/C_d, respectively) that can short-circuit optimization. We measure the optimality gap in both instantaneous and cumulative senses as their performance metrics.
2. Our CEBGAN incorporates the recent advances in computational optimal transport to accelerate its training by 30.7% compared with CBGAN while achieving better final performance (higher C_l/C_d efficiency).
3. CEBGAN (or more generally CEGAN—conditional entropic GAN) enables evaluating the surrogate log-likelihood of samples for decision making during prediction. We develop the formulation of the surrogate log-likelihood for CEGAN and prove its validity in the Supplementary Material. We further investigate its practicability by examining its correlation with actual sample performance (*i.e.*, aerodynamic efficiency).
4. We compare the performance of CGAN and CEGAN on low-dimensional toy problems to provide more insight into each model's behavior and illustrate the paper's key results on an understandable example.
5. We create a dataset of optimal airfoils and the algorithm for generating it for future studies. This dataset and the corresponding code to replicate the paper is located at https://github.com/IDEALLab/CEBGAN_JMD_2021.

2 BACKGROUND AND RELATED WORK

This section first provides some background context on inverse problems, inverse airfoil design, and mainstream generative models to lay the foundation for how we cast inverse airfoil design from a probabilistic viewpoint. Then we enumerate some recent work incorporating conditional generative models to solve inverse design problems or broader inverse problems.

2.1 Inverse Problems

Inverse design falls under the scope of inverse problems [1, 2], in which we need to retrieve the corresponding system parameters $\mathbf{x}$ from the observations $\mathbf{y}$ governed by a forward problem:

$$\mathbf{y} = F(\mathbf{x}) + \mathbf{e} \quad \text{or} \quad R(\mathbf{x},\mathbf{y}) = \mathbf{e} \tag{1}$$

where for the explicit formulation on the left, $F : X \to Y$ is a forward operator mapping parameters to observed data and $\mathbf{e}$ is the observation noise or tolerance; for the implicit one on the right, $R : X \times Y \to \mathbb{R}^m$ is a residual operator measuring X and Y's observation of the governing equations such as PDEs. The explicit form can be converted to the implicit form via $R(\mathbf{x},\mathbf{y}) \leftarrow \mathbf{y} - F(\mathbf{x})$. Although most forward problems are well-posed, their inversions are usually not, with none or multiple $\mathbf{x}$ corresponding to the same $\mathbf{y}$. This cripples deterministic or straightforward attempts at inversion. Many works rely on regularization to escape this pitfall [2, 3].

In contrast to regularization, Bayesian inference tackles this issue by taking all possible solutions into account and quantifying the uncertainty of each with probability measure [1, 2]. From the Bayesian perspective, if we take $\mathbf{x}$ and $\mathbf{y}$ both as the realizations of certain random variables, solving the forward problem is equivalent to sampling a dataset $\{\mathbf{x},\mathbf{y}\}$ from $p(\mathbf{x},\mathbf{y}) = p(\mathbf{y} \mid \mathbf{x})\, p(\mathbf{x})$ where the likelihood $p(\mathbf{y} \mid \mathbf{x})$ is assumed to exist and can be formulated from the deterministic forward problem. The prior $p(\mathbf{x})$ can be uniform, Gaussian, or any other distributions depending on the prior information to embed. The inverse problem then corresponds to deriving the posterior $p(\mathbf{x} \mid \mathbf{y})$ of this probabilistic model.

2.2 Inverse Design and Inverse Airfoil Design

Based on the Bayesian view of inverse problems, we can regard inverse design as retrieving the posterior $p(\mathbf{x} \mid \mathbf{y}, O)$ of the optimal design variables $\mathbf{x}$ given the desired problem parameter $\mathbf{y}$, which are all governed by certain optimality condition in the form of $R(\mathbf{x},\mathbf{y}) = \mathbf{0}$. Specifically, we propose to formulate the inverse design problem in general as:

$$\begin{aligned} p(\mathbf{x} \mid \mathbf{y}, O) &\propto p(O \mid \mathbf{x},\mathbf{y})\, p(\mathbf{x}) \\ &\propto \exp(-\|R(\mathbf{x},\mathbf{y})\|^2/\varepsilon^2)\, p(\mathbf{x}) \end{aligned} \tag{2}$$

where O stands for being optimal and ε controls the tolerance of error. This probabilistic formulation is reasonable because it assigns a higher posterior probability to design variables $\mathbf{x}$ whose residuals are closer to $\mathbf{0}$.

Our inverse airfoil design problem requires the airfoil to satisfy $\mathbf{y}$ and be optimal under a chosen criterion. This criterion is typically computed via some numerical forward model or simulator. In our airfoil example, the airfoil shape is first processed by some shape parameterization algorithm to obtain design variables $\mathbf{x}$ and passed to a mesh generator M for meshing. The generated mesh is then fed together with the desired property $\mathbf{y}$ and boundary conditions $\mathbf{b}$ into a Computational Fluid Dynamics (CFD) solver to produce a chosen objective J. Ideally, the second-order optimality condition of J—$\nabla_{\mathbf{x}} J(M(\mathbf{x}),\mathbf{y},\mathbf{b}) = \mathbf{0}$ and $\mathbf{H}_{\mathbf{x}} J(M(\mathbf{x}),\mathbf{y},\mathbf{b}) \succeq 0$—is used to examine whether $\mathbf{x}$ has properties $\mathbf{y}$ under $\mathbf{b}$ and is also optimal in terms of certain requirements. Following Eqn. (2), we formulate the inverse airfoil design problem as:

$$\begin{aligned} p(\mathbf{x} \mid \mathbf{y},\mathbf{b}, O) &\propto p(O \mid \mathbf{x},\mathbf{y},\mathbf{b})\, p(\mathbf{x}) \\ &\propto \exp\left(-\frac{\|R(\mathbf{x},\mathbf{y},\mathbf{b})\|^2}{\varepsilon^2}\right) p(\mathbf{x}). \end{aligned} \tag{3a}$$

$$R(\mathbf{x},\mathbf{y},\mathbf{b}) = \begin{cases} \nabla_{\mathbf{x}} J(M(\mathbf{x}),\mathbf{y},\mathbf{b}) & \mathbf{H}_{\mathbf{x}} J(M(\mathbf{x}),\mathbf{y},\mathbf{b}) \succeq 0 \\ \infty & \mathbf{H}_{\mathbf{x}} J(M(\mathbf{x}),\mathbf{y},\mathbf{b}) \prec 0 \end{cases} \tag{3b}$$

We try to formulate the inverse airfoil design problem as generically as possible here such that it can extend to other

Copyright © by ASME

domains and tasks, and so we did not specify the components of Eqn. (3a, 3b). However, for the specific problems addressed in this paper, the methodology section details these choices.

The analytical form of the posterior $p(\mathbf{x} \mid \mathbf{y})$ is in general infeasible by virtue of the likelihood's complexity. However, one practical work-around is to use any universal approximator [4] with adequate model capacity—such as any of the mainstream generative models introduced next—to learn $p(\mathbf{x} \mid \mathbf{y})$ from the generated dataset $\{\mathbf{x}, \mathbf{y}\}$ on a data-driven basis.

2.3 Generative Models

Traditional generative models of limited complexity like Gaussian Mixture Models are in general insufficient to approximate real-world high-dimensional target distributions $p_r(\mathbf{x})$. As of writing, the three most commonly used generative models including their variations are generative adversarial networks (GANs) [5, 6, 7, 8, 9], variational autoencoders (VAEs) [10, 11, 12, 13, 14, 15] and flow-based models [16, 17, 18]. They all share a deep neural network generator $G : Z \to X$ in common that serves the same purpose—implicitly representing a distribution $p_g(\mathbf{x})$ via the transformation of latent prior distribution $p(\mathbf{z})$ of noise. They mainly differ in the way they drive p_g towards p_r:

1. GANs (*e.g.*, vanilla GAN [5]) achieve this by either exactly or approximately minimizing some statistical distances between p_g and p_r, as the next section elucidates. In general, GANs do not provide information about the sample likelihood $p_g(\mathbf{x})$. This is one of their primary weaknesses.
2. VAEs achieve this by indirectly maximizing the sample likelihood via its lower bound. Since G here is not designed to be invertible and merely represents a low-dimensional manifold of measure zero in high-dimensional data space, likelihood maximization becomes possible by introducing Gaussian-like noise in the data space and via variational inference.
3. Flow-based models align the dimension of $\mathbf{x}$ and $\mathbf{z}$, and carefully design the generator G to make it invertible such that the analytical form of the density function can be retrieved for direct likelihood maximization.

Currently, GANs often generate higher quality samples than VAE or Flow models. This indicates GANs' good convergence to at least some modes of p_r. VAE's convergence issues are usually attributed to posterior collapse [11, 12, 13], whereas flow-based models currently need to sacrifice their expressivity on the altar of invertibility.

Our work will focus on GANs' learning of the posterior $p(\mathbf{x} \mid \mathbf{y})$, for which the generator now takes the conditional form $G : Y \times Z \to X$ to induce its dependency on the condition $\mathbf{y}$, forming an approximate distribution $p_g(\mathbf{x} \mid \mathbf{y})$.

2.4 Recent Work on Inverse Design

Inverse design based on conditional generative models is an emerging area without a significant body of existing work. So far, within the realm of mechanical engineering, it is almost universally applied to the design of nanomaterials and microstructures, including nano-photonics [19,20,21,22,23], nanoelectronics [24], cellular structures [25], porous materials [26], and crystals [27], etc. It has also been applied to the synthesis of kinematic linkages recently [28]. Outside of those isolated areas, it has been employed by inverse molecular design [29], medical imaging [30, 31], gene expression inference [32], image processing [33] among others. Other than using conditional generative models, inverse design was also enabled by using a traditional numerical regime (*i.e.*, a design optimization framework) [34] and other machine learning algorithms (*e.g.*, Gaussian process and reinforcement learning) [35, 36] for structural optimization, metamaterial design, and flow sculpting. Furthermore, with regard to the generative model's application to unconditional design synthesis, which can potentially be adapted for a conditional configuration, several works have emerged lately. For instance, Chen and Fuge demonstrated design synthesis preserving inter-part dependencies using GANs of hierarchical architecture [37]. Oh *et al.* ran topology optimization and GANs in a loop to explore the design space and synthesize new designs with efficiency [38]. Shu *et al.* generated 3D models using GANs [39].

To the extent of our knowledge, only two existing works studied the conditional synthesis of curves using generative models, both related to airfoil design. In [40], Yilmaz and German used conditional GAN to generate airfoils of specified stall conditions and airfoil drag polars. Achour *et al.* [41] implemented a conditional GAN taking discrete conditions representing four classes as input to generate airfoils falling into the desired quarter of the $[C_l/C_d$ ratio $\times$ shape area$]$ domain. Our work differs from theirs via the key contributions listed in the end of the introduction (Sec. 1).

3 METHODOLOGY

In this section, we present the formulation and technical details of our airfoil prediction frameworks. We first introduce the generation of the airfoil dataset for inverse airfoil design. After that, we shed light on conditional BézierGAN (CBGAN) construction and its evolution to the optimal-transport-based conditional entropic BézierGAN (CEBGAN), which is easier to train and yields information about the sample likelihood. We end the section by describing the metrics we use to measure their performances on our generalized regression problems.

3.1 Dataset Creation via 2D Airfoil Optimization

The conditional GAN models' real dataset consists of optimal shape designs of 2D airfoils. To build the dataset, we perform shape optimization over a range of input boundary conditions to achieve high-performing 2D airfoil designs. We use the SU2[1] solver (an open-source PDE anal-

[1] https://su2code.github.io/. SU2 can deal with different kinds of physical problems by choosing different solvers such as Euler's Navier-Stokes' and Reynolds-averaged Navier-Stokes' equations. In this paper, we optimize

Copyright © by ASME

ysis toolset) to perform gradient-based shape optimization following the general process shown in Fig. 1.

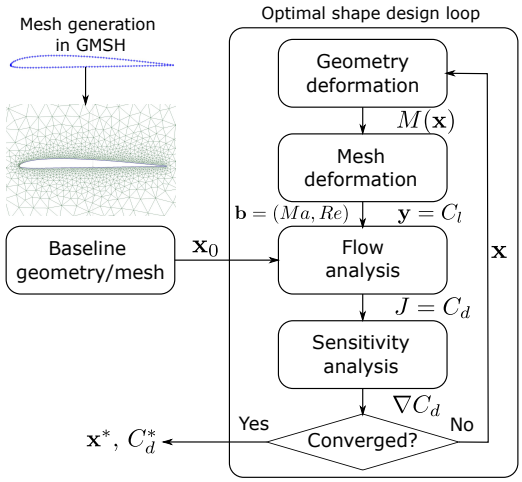


Fig. 1: Flow chart of gradient-based airfoil shape optimization with SU2.

The flow chart starts with a baseline geometry (represented as surface points) and its mesh as input to the design cycle, along with a chosen objective function J and a set of design variables $\mathbf{x}$. SU2 allows users to choose different objectives such as drag coefficient (C_d), lift coefficient (C_l), and efficiency (C_l/C_d). In this paper, we choose C_d as the objective function ($J = C_d$) with a target C_l ($\mathbf{y} = C_l$)—*i.e.*, the optimization is run at a fixed C_l which works by updating the angle of attack (α) during the optimization such that the resulting C_l matches the target C_l value. In SU2, Hicks-Henne bump functions and free-form deformation (FFD) control point approach [42] are used to parameterize 2D airfoils and generate the design variables $\mathbf{x}$. SU2 also provides different optimizers (e.g., SLSQP, CG, BFGS, and POWELL) to do the gradient-based optimization. A chosen gradient-based optimizer will orchestrate the design cycle consisting of the direct flow solver, adjoint solver, and geometry/mesh deformation tools in SU2. The iterative design loop proceeds until an optimum is found or reaching a maximum number of optimizer iterations. We develop an integrated computational pipeline that automates the optimization by merging GMSH[2] (open-source finite element mesh generator) mesher and SU2 optimizer through a Python script.

In this paper, we use the gradient-based SLSQP optimizer and the Hick-Henne parameterization method to minimize C_d such that the efficiency (*i.e.*, C_l/C_d) is maximized at a constant C_l. To find the optimal airfoil design (and mitigate converging to local optima in non-convex problems), for each group of freestream conditions $\mathbf{b}$ and target properties $\mathbf{y}$—*i.e.*, Mach number (Ma), Reynolds number (Re), and target lift coefficient (C_l)—we perform adjoint optimization on eight diverse candidate airfoils and pick the optimized one with the highest efficiency (Fig. 2).

The diverse candidate airfoils are generated by applying Latin Hypercube sampling [43] on the lower-dimensional latent space (of BézierGAN [44]) for diverse coverage (8 samples) and feeding the 8 groups of latent codes into the pretrained BézierGAN model. To generate sufficient data, each of the 8 candidate airfoils is optimized (using adjoint optimization) for 1,000+ steps with the same 1,000+ sets of input freestream conditions (Ma and Re) and target C_l. We select the final adjoint-optimized airfoil with the highest efficiency and store it in our database of samples.

3.2 BézierGAN

BézierGAN is a framework developed by Chen and Fuge [44, 45]. This subsection reviews its crucial components, the underlying mechanism, and conditional formulation, from which entropic BézierGAN is inspired and stems.

3.2.1 Bézier Layer and Regularization

BézierGAN is essentially a specialized InfoGAN [46]. Its only difference to its predecessor is the additional Bézier layer mounted on its generator and the accompanying regularization loss, which ensure the generation of smooth Bézier curves and make it suitable for geometry-related engineering applications.

The Bézier layer is of the mathematical form below [45]:

$$\mathbf{X}_j = \frac{\sum_{i=0}^{n} \binom{n}{i} u_j^i (1-u_j)^{n-i} \mathbf{P}_i w_i}{\sum_{i=0}^{n} \binom{n}{i} u_j^i (1-u_j)^{n-i} w_i}, \qquad j = 0, \ldots, m \tag{4}$$

where $\mathbf{P}$, $\mathbf{w}$ and n are respectively control points, weights, and predetermined degree defining the rational Bézier curve, and $\mathbf{X}$ is the tensor of data points sampled from the Bézier curve according to the $m+1$ parameter variables $\mathbf{u}$ that determine the sampling intervals. For numerical stability, this layer is usually evaluated on a logarithmic scale. It serves as the final output layer of the generator and does not hinder backpropagation, thanks to its differentiability. In the training process of BézierGAN, we can apply additional regularizations to $\mathbf{P}$ and $\mathbf{w}$ to further rectify the quality of the Bézier curve. Apart from this distinction, the other part of BézierGAN's training is the same as those of vanilla GAN and InfoGAN, and can also benefit from their improvements.

3.2.2 Minimax Game and Probabilistic Perspective

Since the learning of disentangled representation is not mandatory in our applications, the mutual information maximization coming from InfoGAN will be suspended in this work. Instead, we focus on the core minimax game inherited from vanilla GAN, which is a min-max optimization of the form below:

$$\min_G \max_D \; \mathbb{E}_{\mathbf{x}\sim p_r(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_Z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{5}$$

airfoils by solving the Reynolds-averaged Navier-Stokes' (RANS) equation.

[2] https://gmsh.info/

Copyright © by ASME

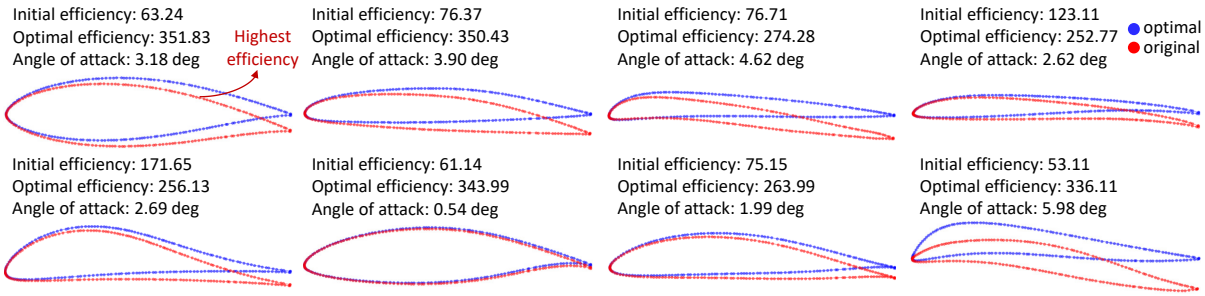


Fig. 2: Candidate baseline airfoils (each consists of 192 surface points) with their optimized designs from SU2. The freestream conditions: $Ma = 0.5$, $Re = 51,000,000$, and target lift coefficient: $C_l = 1.0$.

where D is the discriminator, G is the generator, $\mathbf{z}$ is the noise, and $\mathbf{x}$ are the real samples we are trying to counterfeit using G after running this game several times. Doing so encourages the distribution $p_g(\mathbf{x})$ implicitly represented by G to converge to $p_r(\mathbf{x})$, which is the underlying distribution generating real data. Past work has proven that when the discriminator is trained to optimum before each update of the generator, we are equivalently minimizing the Jensen–Shannon divergence between p_r and p_g, *i.e.*, $\mathrm{JSD}(p_r \,\|\, p_g)$ [5]. In practice, however, because of inefficiency, vanishing gradients, and instability [47], in each epoch, the discriminator is only updated for a few iterations and thus barely arrives at its optimum; so this theory is just a reasonable approximation.

Nevertheless, this probabilistic perspective does provide researchers with many insights and new directions to delve into. Although the mode collapse plaguing vanilla GAN cannot be attributed to JS divergence in full [48, 49], the minimax game based on it still plays a role in this and other defects because several proposals that replace this game assuming JS divergence to be the culprit have achieved significant improvement [6,7,8,9]. One prominent work, if not the most among them, is the Wasserstein GAN (WGAN) [8], which is established on optimal transport theory that we shall introduce in the next subsection.

3.2.3 Conditional Formulation and CBGAN

BézierGAN and vanilla GAN share the same conditional formulation as shown in [50], namely

$$\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_r(\mathbf{x},\mathbf{y})}\{\log D(\mathbf{x},\mathbf{y}) + \mathbb{E}_{\mathbf{z}\sim p_Z(\mathbf{z})}[\log(1-D(G(\mathbf{z},\mathbf{y}),\mathbf{y}))]\} \quad (6)$$

where $\mathbf{y}$ stands for the condition corresponding to $\mathbf{x}$ and both the discriminator and generator additionally take $\mathbf{y}$ as a input, so that G implicitly represents a conditional distribution $p_g(\mathbf{x} \mid \mathbf{y})$. From the same JS divergence point of view, it readily follows that this minimax game is approximately minimizing $\mathrm{JSD}\,(p_r(\mathbf{x},\mathbf{y}) \,\|\, p_g(\mathbf{x} \mid \mathbf{y})\, p_r(\mathbf{y}))$ to lead $p_g(\mathbf{x} \mid \mathbf{y})$ to $p_r(\mathbf{x} \mid \mathbf{y})$.

Fig. 3 presents the architecture of CBGAN for our inverse airfoil design task. Its generator takes the Gaussian noise vector $\mathbf{z}$, the desired property $\mathbf{y}$ which is C_l here, and the freestream conditions $\mathbf{b}$ which are Ma and Re as input, and output the design variables $\mathbf{x}$, which are the 192 data points on the Bézier curve of an airfoil and the angle of attack α. The discriminator, on the other hand, takes $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{b}$ as input and produces the probability of being an optimal airfoil.

3.3 Entropic BézierGAN

Arjovsky and Bottou [47] hypothesize that one critical defect vanilla GANs suffer from is the discontinuity of JS divergence over distributions concentrated on low-dimensional manifolds embedded in high-dimensional spaces. These low-dimensional manifolds are usually misaligned, so it prompts the discriminator to overpower the generator and leads to vanishing gradients. To overcome this convergence issue, WGANs equipped with Wasserstein distance, a special case of optimal transport (OT) distance, came into play. This modification improved the stability of GAN's training process drastically and significantly alleviated mode collapse [8, 9] common in early GAN models. Later, these WGANs were further generalized to a broader range of OT distances, and the Lipschitz smoothness constraint originally enforced with crudeness was also replaced with an entropic soft regularization term in the loss function [51, 52, 53, 54]. These GANs are dubbed Smoothed WGAN, OT GAN, or Entropic GAN (EGAN), and the lower bound estimation of likelihood is also enabled thanks to their special properties [54].

Many real-world datasets indeed reside on low-dimensional manifolds [55] and the airfoil dataset should be no exception, as reflected in [44]. Therefore, we hypothesized that EGAN based on optimal transport might carry corresponding benefits for inverse design. Our entropic sibling of BézierGAN also employs the Bézier layer as the final layer of the generator. Likewise, it has no difference to regular EGANs other than that.

3.3.1 Optimal Transport with Entropic Regularization

For two probability distributions—specifically in our GAN training case, the real data distribution $p_r(\mathbf{x})$ and the generator's approximate distribution $p_g(\mathbf{x})$—the Kantorovich optimal transport distance regularized with entropy

Copyright © by ASME

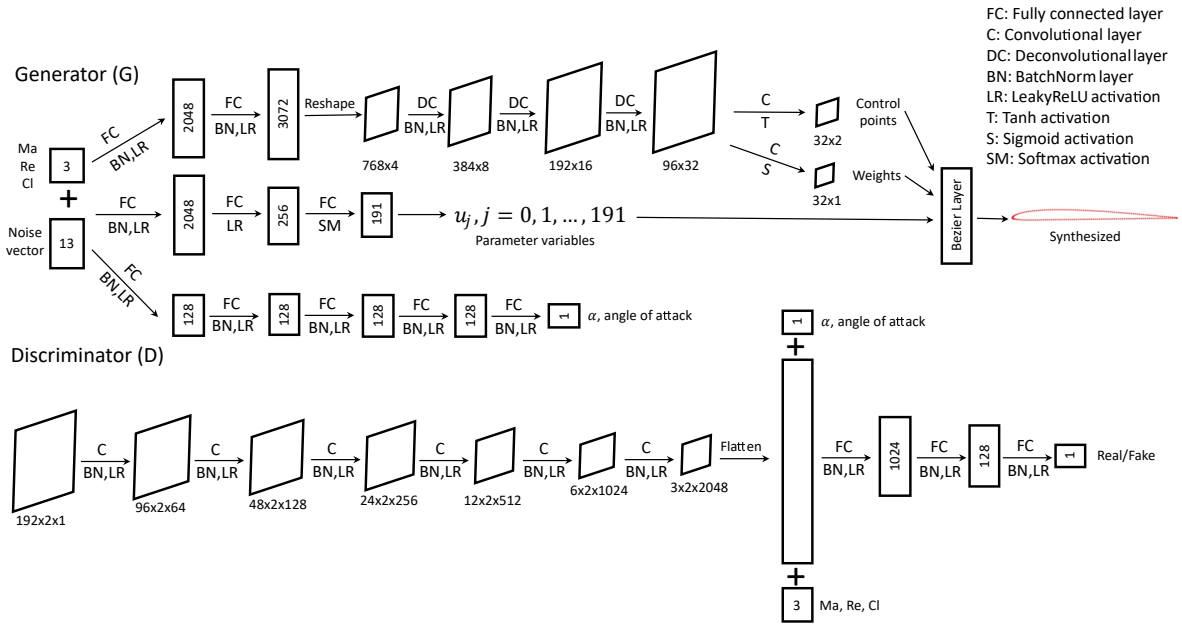


Fig. 3: Overall CBGAN model architecture. CEBGAN shares the same generator architecture and does not need the discriminator.

or equivalently KL divergence has the following expression [56, 57]:

$$\mathrm{OT}_\lambda(p_r, p_g) := \min_{\mathbb{P}_{X,\hat{X}} \in \Pi(p_r, p_g)} \mathbb{E}_{\mathbf{x},\hat{\mathbf{x}} \sim \mathbb{P}_{X,\hat{X}}}[c(\mathbf{x}, \hat{\mathbf{x}})] + \lambda \mathrm{KL}(\mathbb{P}_{X,\hat{X}} \mid p_r \times p_g) \quad (7)$$

where $\Pi(p_r, p_g)$ is the set of joint distributions $\mathbb{P}_{X,\hat{X}}$ whose marginal distributions equal p_r and p_g, c is cost function usually symmetric positive, and $\lambda \geq 0$ is a weight controlling the degree of regularization. When $c = \|\cdot\|_2$ and $\lambda = 0$, this OT distance reduces to the well-known Wasserstein-1 distance used in WGAN [8, 9], minimizing which is how WGAN is trained.

Yet the evaluation of Wasserstein-1 distance is not as easy as its counterpart for which $\lambda > 0$. Though a direct optimization of Eqn. (7) is highly intractable, thanks to the Fenchel-Rockafellar theorem, the strong duality holds so that we can evaluate its dual instead [56, 57, 58]:

$$\mathrm{OT}_\lambda(p_r, p_g) = \max_{f,g} \mathbb{E}_{\mathbf{x} \sim p_r}[f(\mathbf{x})] + \mathbb{E}_{\hat{\mathbf{x}} \sim p_g}[g(\hat{\mathbf{x}})] - \lambda \mathbb{E}_{\mathbf{x},\hat{\mathbf{x}} \sim p_r \times p_g}[\exp(v(\mathbf{x}, \hat{\mathbf{x}})/\lambda) - 1] \quad (8)$$

$$v(\mathbf{x}, \hat{\mathbf{x}}) := f(\mathbf{x}) + g(\hat{\mathbf{x}}) - c(\mathbf{x}, \hat{\mathbf{x}}) \quad (9)$$

where f and g are the Lagrange multipliers that can be optimized through a universal approximator like neural networks or the Sinkhorn algorithm introduced next. Equipped with this OT probability distance, in EGAN, instead of minimizing the JS divergence, we can train the generator G, bringing p_g to p_r via [52, 54]:

$$\min_G \mathrm{OT}_\lambda(p_r, p_g). \quad (10)$$

3.3.2 Sinkhorn Divergence

Though greatly increasing the evaluation efficiency and mitigating the curse of dimensionality [56], the entropic regularization, in consequence, brings about an entropic bias, namely $\mathrm{OT}_\lambda(p, p) \neq 0$, which may induce mode collapse [58]. One practical way to eliminate this bias is to use Sinkhorn divergence defined below composed of OT_λ to evaluate the discrepancy between distributions:

$$\mathrm{S}_\lambda(p_r, p_g) := \mathrm{OT}_\lambda(p_r, p_g) - \frac{1}{2}\mathrm{OT}_\lambda(p_r, p_r) - \frac{1}{2}\mathrm{OT}_\lambda(p_g, p_g) \quad (11)$$

It is proved in [58] that this metric is indeed symmetric, convex, smooth, and positive definite, thus a better option than OT_λ. Now we can train the generator instead by [51]:

$$\min_G \mathrm{S}_\lambda(p_r, p_g). \quad (12)$$

3.3.3 Sinkhorn Algorithm

Evaluating Eqn. (8) via parameterized f and g represented by neural networks, as in [8,9,54], is time-consuming. To accelerate the training process, we can realize this clumsy maximization with the Sinkhorn algorithm [58], which is the coordinate ascent coming from the first-order optimality condition for Eqn. (8). Normally for a λ not too close to 0, the

Copyright © by ASME

Sinkhorn algorithm can converge within milliseconds [56]. Eqn. (11) can then be evaluated by applying the Sinkhorn algorithm to each of its three terms.

3.3.4 Conditional Formulation and CEBGAN

Following the same rationale for constructing conditional BézierGAN, we can approximate $p_r(\mathbf{x} \mid \mathbf{y})$ with $p_g(\mathbf{x} \mid \mathbf{y})$ through minimizing the Sinkhorn divergence $\mathrm{S}_\lambda(p_r(\mathbf{x},\mathbf{y}) \| p_g(\mathbf{x} \mid \mathbf{y}) p_r(\mathbf{y}))$. This indicates we have to design a cost function $c([\mathbf{x},\mathbf{y}],[\hat{\mathbf{x}},\hat{\mathbf{y}}])$ for the prediction-condition bundles. One effortless way is to construct it by $c([\mathbf{x},\mathbf{y}],[\hat{\mathbf{x}},\hat{\mathbf{y}}]) = c_1(\mathbf{x},\hat{\mathbf{x}}) + c_2(\mathbf{y},\hat{\mathbf{y}})$. Specifically, for the evaluation of S_λ in our airfoil design application, we set $\lambda = 5$ and build a shift-invariant cost function c with L_1 norm by

$$c([\mathbf{x},\mathbf{y},\mathbf{b}],[\hat{\mathbf{x}},\hat{\mathbf{y}},\hat{\mathbf{b}}]) = |\mathbf{x}-\hat{\mathbf{x}}| + |\mathbf{y}-\hat{\mathbf{y}}| + |\mathbf{b}-\hat{\mathbf{b}}|. \tag{13}$$

As to the architecture of CEBGAN, since we are using Eqn. (12) for its training, the discriminator is no longer required. We use the same generator as shown in Fig. 3 for inverse airfoil generation. This unity helps compare the performance of CBGAN and CEBGAN.

3.3.5 Conditional Surrogate Log-Likelihood

Similar to [54], it can also be proven, as provided in the supplementary material [See Supplemental Material §1], that this conditional formulation is equivalent to maximizing the sample likelihood of the explicit density model below provided c is shift-invariant:

$$p(\mathbf{x},\mathbf{y}) = \int_{\hat{\mathbf{y}},\mathbf{z}} p(\mathbf{x},\mathbf{y} \mid \hat{\mathbf{y}},\mathbf{z})\, p_r(\hat{\mathbf{y}})\, p(\mathbf{z})\, d\hat{\mathbf{y}}\, d\mathbf{z} \tag{14}$$

$$p(\mathbf{x},\mathbf{y} \mid \hat{\mathbf{y}},\mathbf{z}) \propto \exp\left(-\frac{c([\mathbf{x},\mathbf{y}],[G(\mathbf{z},\hat{\mathbf{y}}),\hat{\mathbf{y}}])}{\lambda}\right). \tag{15}$$

One can then derive the corollary that

$$\begin{aligned}\log p(\mathbf{x},\mathbf{y}) \geq\ & -\frac{1}{\lambda}\mathbb{E}_{\mathbb{P}^\star_{Z|X,Y}}\left[c([\mathbf{x},\mathbf{y}],[G^\star(\mathbf{z},\mathbf{y}),\mathbf{y}])\right] \\ & + \mathbb{E}_{\mathbb{P}_Z}\left[\log p(\mathbf{z})\right] + H\left(\mathbb{P}^\star_{Z|X,Y}\right) + \log p_r(\mathbf{y}) + \text{const}\end{aligned} \tag{16}$$

in which

$$\mathbb{P}^\star_{Z|X,Y} = \mathbb{P}_Z(\mathbf{z}) \exp\left(\frac{v^\star([\mathbf{x},\mathbf{y}],[G^\star(\mathbf{z},\mathbf{y}),\mathbf{y}])}{\lambda}\right). \tag{17}$$

where H is the Shannon-entropy function, $\mathbb{P}_Z$ is the empirical distribution of sampled noise, and $v^\star$ (Eqn. (9)) is evaluated with the optimal $f^\star$ and $g^\star$. Then for a given condition $\mathbf{y}$ and a series of generated samples $\{\mathbf{x}_i = G(\mathbf{z}_i,\mathbf{y})\}_n$, we can use the RHS of Eqn. (16) as the surrogate log-likelihood (SLL) to evaluate the plausibility of the bundles $\{[\mathbf{x}_i,\mathbf{y}]\}_n$ and select the ones with higher SLL as the samples more likely to have good quality. While sample likelihood is not always reliable for this purpose (especially when the generative model is ill-converged [59]) and the SLL only provides a lower bound, it is nevertheless our only resort for evaluating GAN-based sample likelihoods. (Recall from Sec. 2.3 that the lack of explicit exact sample likelihoods is GAN's key weakness compared to VAE- or Flow-based models.) We will investigate its effectiveness statistically in the experiment section.

3.4 Metrics

To evaluate the quality of our conditional GAN's approximation to $p(\mathbf{x} \mid \mathbf{y},\mathbf{b},O)$ for the inverse airfoil design in a quantitative manner, we compute two classes of metrics. First, we compute the kernel maximum mean discrepancy (MMD)—a measure of distributional fit—that measures how well the generative model matches the training data. Second, we compute how the generative model reduces the optimality gap between the ground-truth optimal airfoil and the one produced by the conditional generative model. This directly measures how the generative model affects downstream optimization performance. We evaluate this in both the instantaneous setting and cumulatively, such as when the generative model warm-starts a traditional optimization process. In addition, we calculate the Pearson correlation coefficient between the surrogate log-likelihood of generated airfoils and their actual performance and plot its distribution.

3.4.1 MMD

During the cross-validation phase before the final training, we use the kernel maximum mean discrepancy (MMD) [60] to measure the discrepancy between the two joint distributions $p_r(\mathbf{x},\mathbf{y})$ and $p_g(\mathbf{x} \mid \mathbf{y})p_r(\mathbf{y})$. The MMD between two distributions $p(x)$ and $q(y)$ is defined by

$$\mathrm{MMD}^2(p,q) = \mathbb{E}\left[k(x,x') - 2k(x,y) + k(y,y')\right] \tag{18}$$

where we select Gaussian kernel with $\sigma = 1$ as k in our application as this is a common choice.

Though not as intuitive as traditional regression metrics such as MSE in giving an intuitive estimation of the error magnitude directly in the data space, MMD is much more justified for this work as MSE essentially comes from the KL divergence between the data distribution and the unimodal Gaussian regression model, whereas $p_g(\mathbf{x} \mid \mathbf{y})$ here is no longer such a simplified conditional distribution.

3.4.2 Reduction in Instantaneous Optimality Gap

Regarding the GAN prediction as a one-step optimization, instantaneous optimality gap checks the efficacy of the conditional GANs by comparing the performance (C_l/C_d ratio in our case) of the predicted airfoil to the performance of the optimized airfoil after one step (iteration) of the iterative adjoint method (*i.e.*, $h_{i=1}$ in Eqn. (19)). Regardless of the instant time saving, we want to show that the one-step prediction using conditional GANs also surpasses an adjoint

Copyright © by ASME

step using gradient-based optimization to improve the airfoil performance. If we ignore the time cost of the instant one-step prediction and compare the GAN predicted airfoil to the original design (*i.e.*, h_0 without any optimization) directly, the performance can have a more significant improvement (as demonstrated in Fig. 7).

3.4.3 Reduction in Cumulative Optimality Gap

While the Instantaneous Optimality Gap (often referred to and used in papers on so-called zero-shot optimization) is a useful performance measure, it ignores the fact that an optimizer can further refine a conditional generative model's prediction. That is, a conditional model might *warm-start* a further optimization. This section describes how we calculate the amount of effort the conditional GANs can save in those cases.

Specifically, given an n-iteration optimization history $\{h_i\}_n$ which records the performance h_i (*i.e.*, C_l/C_d ratio) at each optimization iteration i, after calculating the percentage of each h_i with respect to the optimal value h_n in the history, we define the *cumulative optimality gap* (COG) as the area enclosed by the optimization history curve normalized in percentage and the horizontal line of 100% corresponding to h_n (a specific example can be seen in Fig. 9). In other words,

$$\mathrm{COG}(\{h_i\}_n) = \frac{\sum_{i=1}^{n} h_n - h_i}{h_n}. \tag{19}$$

Unlike the instantaneous optimality gap, this index also takes into account the performance of the subsequent optimization steps, assigning lower COG values to the ones that approach the optima faster.

With the COG defined above, for each group of input conditions, if we have the corresponding history $\{h_i^o\}_m$ of an original adjoint optimization (using original airfoils as a start) and the history $\{h_i^r\}_n$ of a restart adjoint optimization (using GAN predicted airfoils as a warm start) accelerated by the conditional GANs (as shown in Fig. 8 and Fig. 9), we can examine the amount of effort the conditional GANs help save via the *relative reduction in COG*, namely

$$\mathrm{RiCOG} = \frac{\mathrm{COG}(\{h_i^o\}_m) - \mathrm{COG}(\{h_i^r\}_n)}{\mathrm{COG}(\{h_i^o\}_m)} \times 100\%. \tag{20}$$

3.4.4 Correlation between Surrogate Log-Likelihood and C_l/C_d Efficiency

To justify the usage of SLL (Eqn. (16)) for selecting airfoil predictions, we need to statistically verify that there is a positive correlation between the airfoil's surrogate value and its performance, which is the C_l/C_d efficiency in our case. This can be accomplished by first generating n airfoils for each of the m input conditions in the test set. Then for each input condition indexed by i we evaluate the corresponding airfoil predictions' surrogate values $\{s_j^{(i)}\}_n$ and C_l/C_d efficiencies $\{e_j^{(i)}\}_n$ and calculate the *Pearson correlation coefficient* r_i between them:

$$r_i = \frac{\sum_{j=1}^{n}\left(s_j^{(i)} - \bar{s}^{(i)}\right)\left(e_j^{(i)} - \bar{e}^{(i)}\right)}{\sqrt{\sum_{j=1}^{n}\left(s_j^{(i)} - \bar{s}^{(i)}\right)^2}\sqrt{\sum_{j=1}^{n}\left(e_j^{(i)} - \bar{e}^{(i)}\right)^2}}. \tag{21}$$

Finally, we demonstrate the distribution of these m coefficients $\{r_i\}_m$ using a histogram. If the SLL is a practical indicator of a sample's optimality, we can expect most correlation coefficients to be at least greater than 0 and, preferably, closer to the ideal or maximum value of 1.

4 EXPERIMENTS, RESULTS AND DISCUSSION

We first use two simple experiments in low-dimensional spaces to visually study and illustrate the ability of and the performance differences between CGAN and CEGAN, specifically in:

1. Ability to converge to complicated conditional distributions.
2. Ability to capture multimodality of the distributions.

Because of the complexity of the posteriors, these two factors are predominantly the foundation of a generative model's good performance in tackling high-dimensional inverse problems. Only a model with these two abilities can generate samples of good fidelity and handle the ubiquitous inversion ambiguity in inverse problems, *i.e.*, having multiple solutions to the same input.

After using these simple illustrative examples to build intuition, we then employ both generative models on the more realistic problem of learning an optimal 2D airfoil manifold, which is intrinsically a high-dimensional conditional distribution. We compare their learning performance through the lens of reducing the optimality gap and time of airfoil CFD optimization, wherein we use the generative models to provide a good warm-start initialization in the neighborhood of the final solution. Moreover, we investigate the effectiveness of SLL in assessing generated sample's quality.

4.1 Revisiting Regression

This experiment aims to illustrate the ability of conditional GANs in approximating complicated conditional distributions and the performance differences between a traditional Gaussian regression model, the vanilla conditional GAN, and the entropic one based on optimal transport.

For this purpose, we define a toy problem: a 1D Gaussian mixture conditional distribution, in the form:

$$p(y \mid x) = \frac{1}{2}\mathcal{N}(y \mid x^2 - 1, 0.05^2) + \frac{1}{2}\mathcal{N}(y \mid x^3, 0.05^2). \tag{22}$$

A dataset $\{x_i, y_i\}$ composed of 200 samples is then sampled from $p(x,y) = p(y \mid x)\,p(x)$ where $p(x) = \mathcal{U}(-1,1)$, a uniform distribution between -1 and 1. These samples are then

Copyright © by ASME

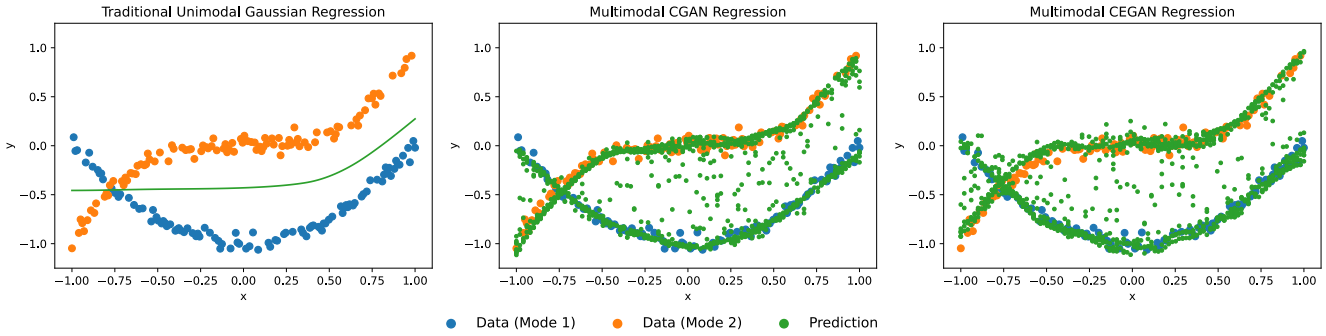


Fig. 4: Learning results of the regression problem of learning a two-mode Gaussian mixture conditional distribution. The orange and blue dots underneath are the data points for training, corresponding to the two different modes, while the green line and the green dots are the predictions made by the regression models.

fed into the generative models for training. They are illustrated in Fig. 4 as orange and blue dots underneath, with each color corresponding to each Gaussian component.

Three regression models are then selected to retrieve this multimodal conditional distribution for comparison. The first one is a traditional unimodal Gaussian regression model $q(y \mid x) = \mathcal{N}(y \mid f(x), \sigma^2)$ whose mean function f is represented by a neural network and trained with MSE loss as usual. The result is shown on the left of Fig. 4, with the mean function f plotted in green. Obviously, it can converge to neither one of the two modes because MSE is only minimized when the prediction approaches the average of the targets. From a probability point of view, this is because MSE originates from the KL divergence—$\mathrm{KL}\left(p(x,y) \| q(y \mid x) p(x)\right)$—while minimizing this divergence can only lead $q(y \mid x)$ to average across all the modes of $p(y \mid x)$ [61]. This further explains why MSE is not an ideal metric for assessing learning results in more complicated applications wherein having mode collapse yet producing good quality samples is usually preferred over a mediocre overall convergence.

This conditional distribution is then approximated by CGAN and CEGAN, both having the same generator structure representing $q(y \mid x)$ with more complexity than the former Gaussian model. After the same amount of training epochs, we can see from the sample points they generate—the green dots in Fig. 4, with the CGAN's prediction in the middle and the CEGAN's on the right—that they both learn to recover the multimodality of the conditional distribution and have comparable convergence. Because of the additional discriminator training phase, each training epoch of CGAN is longer than that of CEGAN under our hyperparameter configuration, not to mention that it takes extra effort in tuning hyperparameters to redress the delicate balance between the discriminator and the generator when training vanilla GANs. These bolster our preference for CEGAN over CGAN, especially when applied to learning complicated conditional distributions.

4.2 Mode Collapse Examination

In this experiment, we demonstrate EGAN's ability to overcome mode collapse and the convergence issues in vanilla GANs and handle the multimodality of the target distribution.

A commonly used toy dataset for mode collapse examination, as in [49], is the 2D Gaussian mixture distribution in Eqn. (23) below with 8 components located on a circle of a certain radius, which equals 15 here:

$$p(\mathbf{x}) = \frac{1}{8} \sum_{n=1}^{8} \mathcal{N}\left(\mathbf{x} \,\middle|\, 15 \left[\cos \frac{n\pi}{4}, \sin \frac{n\pi}{4}\right]^T, \mathbf{I}\right). \tag{23}$$

This is our learning target, mimicking the inverse ambiguity predicament. Eight hundred samples are thereafter sampled from this synthetic distribution, forming a dataset on which both a vanilla GAN and an EGAN with the same generator architecture are trained to see if they can capture all of the 8 modes. Though there is seemingly nothing conditional in this study, we can still take $p(\mathbf{x})$ equivalently as $p_{X|Y=\mathbf{y}}(\mathbf{x})$, *i.e.*, a conditional distribution with its condition Y fixed. Success in learning this distribution is a prerequisite for a high-performing conditional generative model.

The learning results are demonstrated in Fig. 5, with the 800 samples plotted as blue dots underneath and the generated samples of each model as orange dots above. The vanilla GAN fails to sufficiently converge to all modes in this case, while the EGAN trained with Sinkhorn divergence successfully captures all the 8 modes. Similar convergence issues of vanilla GAN are also reported in many works such as [49]. This result indicates that EGAN is a much better choice for solving inverse design problems where inversion ambiguity may exist.

4.3 Airfoil Prediction

After grasping the general performance distinctions between CGAN and CEGAN with the help of those low-dimensional toy problems, we now turn to a more realistic problem. In this final experiment, with the Bézier layer equipped, we investigate CBGAN and CEBGAN's ability to learn the high-dimensional posterior of optimal airfoils conditioned on the freestream conditions and target property. We generate the corresponding near-optimal airfoil on unseen in-

Copyright © by ASME

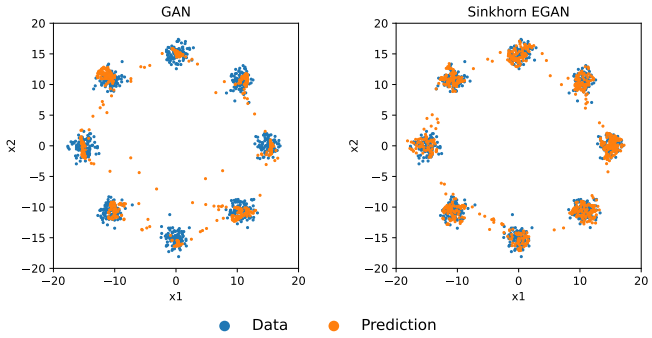


Fig. 5: Result of GAN and EGAN learning Gaussian mixture. The blue dots underneath are the data points for training, while the orange dots are the samples generated by GANs.

put conditions and also use this as a warm-start initialization to short-circuit the subsequent optimizations. We quantify and compare these two GANs' performances using the metrics defined in Sec. 3.4. Regarding CEBGAN, we examine its SLL's efficacy in distinguishing good predictions from the poor ones after CEBGAN converges.

4.3.1 Dataset and Training

Our dataset contains 1,245 optimized airfoils with corresponding input freestream conditions **b** and the target properties **y** (*i.e.*, *Ma*, *Re*, and C_l).

Optimized airfoils: Each airfoil is optimized using the adjoint method described in Sec. 3.1. Each optimized airfoil consists of 192 surface points (192×2 coordinates) along with the optimized angle of attack, α (Fig. 6).

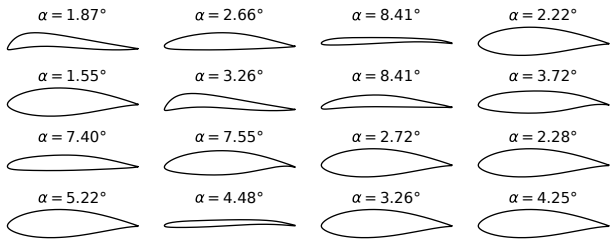


Fig. 6: Samples of the database.

Input conditions: We generate N groups of *Ma*, *Re*, and C_l through the Latin Hypercube sampling strategy to evenly cover the design space. In this paper, we have *Ma* ranging from 0.2 to 0.9, *Re* from 10^7 to 10^8, and C_l from 0.8 to 1.4. We excluded the input conditions leading to airfoil optimization failure (divergence that leads to negative or abnormally high efficiency). The final dataset contains 1,245 groups of input conditions (*Ma*, *Re*, and C_l) and the corresponding 1,245 optimized airfoils. The average time elapse of each SU2 airfoil optimization instance is 55.7 mins on the Deepthought2 HPC[3] at UMD, with 500 instances running in parallel.

The 1,245 airfoils are split into two parts: 995 airfoils for 4-fold cross-validation and training the final generative models, and 250 airfoils reserved for testing. During the cross-validation phase, MMD (Sec. 3.4.1) is used for evaluating GANs' convergence to the distribution of the validation set, against the value of which we adjust the hyperparameters accordingly, setting them to the candidate having the lowest MMD. Once every hyperparameter value is determined, all the 995 training samples are fed into CBGAN and CEBGAN to train the final generative models. The CBGAN is trained with a batch size of 32 for 160,000 iterations, and it takes 3h 51m to finish, while the CEBGAN with a batch size of 128 for 120,000 iterations and takes 2h. Both are trained on NVIDIA Tesla V100 DGXS 32GB GPU. It is hard to directly compare their training speed as they have different hyperparameters, training algorithms, and final performances, but roughly in the sense of time cost per iteration, CEBGAN is about 30.7% faster to train than CBGAN.

4.3.2 Quantitative Performance of Conditional GANs

The freestream conditions (*Ma* and *Re*) and target lift coefficients (C_l) of the 250 testing airfoils are fed into the trained CBGAN and CEBGAN models to generate 250 airfoils with fixed zero noise, respectively. The predicted airfoils' efficiencies are evaluated by the SU2 simulator [62] and benchmarked with the ground truth (*i.e.*, the optimal efficiency from a converged adjoint optimization) to demonstrate the performance of the two conditional GANs.

Specifically, for performance (C_l/C_d) validation of a predicted airfoil, the corresponding input conditions (*Ma*, *Re*, and C_l) of a predicted airfoil (including predicted α) together with other default freestream conditions[4] are wrapped into a SU2 configuration file. GMSH generates a 2D mesh of the predicted airfoil (*i.e.*, 192×2 coordinates) (as shown in Fig. 1), which we then write into a SU2 format. The efficiency is evaluated by solving the RANS equations on the predicted airfoil using air with the corresponding freestream conditions and target C_l.

The ground-truth efficiency value is acquired by performing the iterative adjoint optimization exhibited in Fig. 1. With the same input conditions, the eight candidate airfoils (Fig. 2) are iteratively optimized. The optimized one with the highest efficiency is the ground-truth optimal design, and the highest efficiency is the ground-truth optimal efficiency.

For easy comparison, we normalize the 250 testing samples' efficiencies into percentages. The ground-truth optimal efficiency (*i.e.*, h_n in Eqn. (19)) of each airfoil is represented as 100%, which is used as the comparison baseline. The other types of efficiencies can be represented as percentages by dividing with their corresponding ground-truth efficiency (*i.e.*, h_i/h_n), indicating how close they are to the optimal efficiency.

[3] https://hpcc.umd.edu/hpcc/dt2.html

[4] In this paper, air's freestream pressure is 101,325 Pa and the freestream temperature is 288.15 K.

Copyright © by ASME

Reduction in instantaneous optimality gap: For a more intuitive comparison, we first compute and compare the average reduction in instantaneous optimality gap (Sec. 3.4.2) of the 250 testing airfoils in different scenarios. Fig. 7 illustrates the comparison:

1. The blue dash-dotted line indicates the average ground-truth efficiency of the 250 testing samples. As every ground-truth efficiency is represented as 100%, the average has no variance.
2. The orange dashed line indicates the average efficiency of the 250 predicted airfoils from CBGAN, and the green dashed line indicates the average efficiency of the 250 CEBGAN generated airfoils. Without using any further optimization, the CBGAN predicted airfoils achieve an average of 80.8% of the ground-truth efficiency while the CEBGAN predicted airfoils achieve an average of 95.8% of the ground-truth efficiency.
3. The blue dotted line indicates the average efficiency (22.9%) of the original 250 airfoils before using adjoint optimization.
4. The blue dashed line indicates the average efficiency of the optimized 250 airfoils after taking one step (iteration) of adjoint optimization using the SU2 optimizer. These airfoils achieve an average of 49.3% of the ground-truth efficiency.
5. The box plot underneath indicates the minimum, the first quartile, the median, the third quartile, and the maximum of the 250 testing airfoils at each of the three scenarios (*i.e.*, GAN prediction, one-step adjoint, and initial design). The medians—75.9%, 84.2%, 66.4%, 19.4% from left to right—do not coincide well with the means above, which suggests skewness in the distributions of instantaneous optimality gaps.

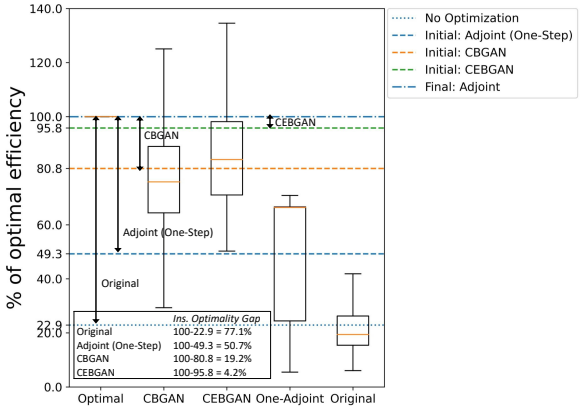


Fig. 7: Illustration of reduction in instantaneous optimality gap using the averaged efficiencies of the 250 testing samples.

Both conditional GANs can quickly (<1 second) predict airfoils whose performance (*i.e.*, efficiency) has been significantly improved (from 22.9% to 80.8% with CBGAN and 95.8% with CEBGAN for these 250 testing samples) compared to the original designs. This represents a reduction in instantaneous optimality gap (*w.r.t.* true optimal C_l/C_d). For CBGAN, the instantaneous gap is reduced by a factor of 4.0× compared to initial airfoils and 2.6× compared to one step of the adjoint method. For CEBGAN, the two numbers are 18.4× and 12.1×, respectively. The average efficiency of the CEBGAN predicted airfoils has been close to the ground-truth one (100%). We have also computed a two-sample t-test on their difference in instantaneous optimality gap. This test resulted in $p = 0.0001$. Though the conditional GANs cannot directly generate the ground-truth optimal airfoils, they surpass an adjoint step and provide a warm start point for a restart optimization, as we show next.

Reduction in cumulative optimality gap: We demonstrate the reduction in cumulative optimality gap (Sec. 3.4.3) by comparing the iterative optimization histories (averaged) of the 250 testing samples between the original adjoint optimization (using original airfoils as a start) and the restart adjoint optimization (using GAN predicted airfoils as a warm start). Fig. 8 illustrates the comparison:

1. The blue dash-dotted line, the blue dotted line, and the three dashed lines (*i.e.*, green, orange, and blue) indicate the same average efficiencies in Fig. 7.
2. The blue solid curve indicates the original adjoint optimization history of the 250 testing samples (in average). The average number of evaluations for the original adjoint optimization is 41. The blue fill indicates the efficiency standard deviation (± StdDev) of the 250 testing samples for each evaluation.
3. The orange and green solid curves indicate the restart adjoint optimization histories of the 250 testing samples (in average) with CBGAN and CEBGAN, respectively. The average number of evaluations of the restart adjoint optimization is 35 for CBGAN and 20 for CEBGAN. The orange and green fills indicate the efficiency standard deviation (± StdDev) of the 250 testing samples for each evaluation.
4. The orange and green dash-dotted lines indicate the average efficiency of the optimal airfoils after the restart optimization with CBGAN and CEBGAN, respectively. As we can see, the restart optimization can lead to a slightly higher efficiency, which is 102.6% of the ground-truth efficiency for CBGAN and 98.2% of the ground-truth efficiency for CEBGAN.

If we compare the area enveloped by the blue solid curve, y-axis, and the blue dash-dotted line to the area enveloped by the orange/green solid curve, y-axis, and the orange/green dash-dotted line (as described in Sec. 3.4.3), we can see a significant relative reduction in cumulative optimality gap (RiCOG) by 41.6% for CBGAN and 91.3% for CEBGAN.

It is also worth noticing that both the CEBGAN and CBGAN can yield some predictions that either instantaneously goes beyond the 100% ground-truth optimal efficiency (as in Fig. 7), or can be further optimized to surpass this threshold

Copyright © by ASME

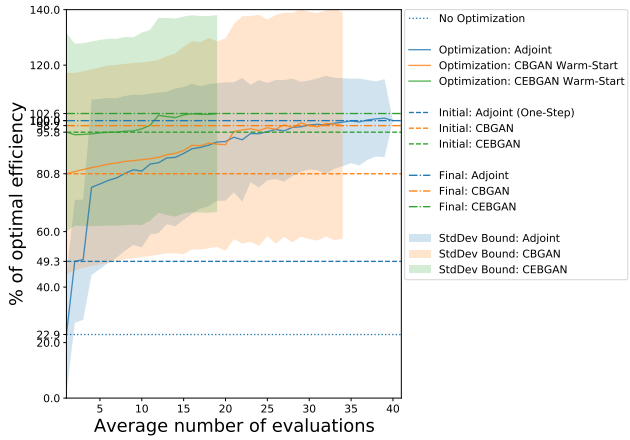


Fig. 8: Illustration of reduction in cumulative optimality gap using the averaged optimization histories of the 250 testing samples.

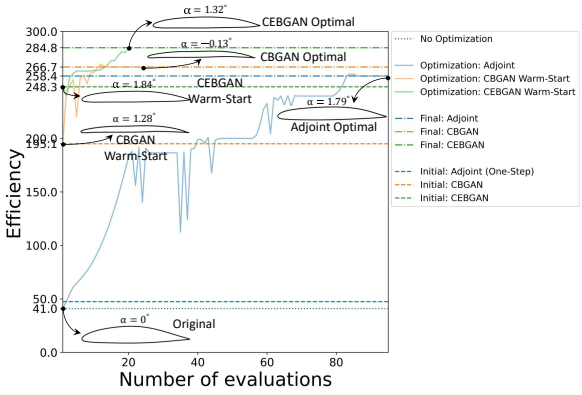


Fig. 9: Illustration of reduction in cumulative optimality gap using a single concrete case ($Ma = 0.72$, $Re = 60,799,053$, and $C_l = 0.88$).

(as in Fig. 8). This indicates that the dataset contains many locally optimal airfoil designs, even though we attempt to select the best one among the eight candidates, as mentioned in Sec. 3.1. Therefore the conditional GANs are still learning the distribution of local minima. More discussion on this will be presented in Sec. 5.

Reduction in cumulative optimality gap of an example airfoil: We demonstrate a single concrete case of one example airfoil (predicted using both CBGAN and CEBGAN) to show how the results in Fig. 8 translate to a single data point. Fig. 9 demonstrates the example case:

1. The blue, orange, and green dashed lines indicate the efficiencies of the one-step adjoint, CBGAN predicted, and CEBGAN predicted airfoils, respectively. The blue dotted line indicate the efficiency of the original airfoil.
2. The blue, orange, and green dash-dotted lines indicate the efficiencies of the ground-truth optimal, CBGAN restart optimized, and CEBGAN restart optimized airfoils, respectively.
3. The blue, orange, and green solid curves indicate the original, CBGAN, and CEBGAN restart adjoint optimization histories of this specific example airfoil.

Instead of using the percentage of the ground-truth optimal efficiency, in Fig. 9, the y-axis indicates the actual efficiency values (these are normalized to percentages to make multiple airfoils comparable on the same axis in Fig. 7 and Fig. 8). In Fig. 9, other than demonstrating the reduction in cumulative optimality gap of a specific case, we can also show the specific status of the airfoil in different stages. For example, we provide the airfoil shapes in four different stages (original airfoil, optimal airfoil after original adjoint optimization, GAN predicted airfoils, and the optimized airfoils after restart adjoint optimization). As a warm start, the GAN predicted airfoils can achieve the optima (even higher efficiency) faster than using the original adjoint optimization. With the GAN predicted airfoil, even if we halt the restart optimization halfway, we can still achieve an airfoil of similar quality to a full adjoint run. Fig. 10 uses histograms and kernel density estimation (KDE) to show the distribution of the relative reduction in cumulative optimality gap (RiCOG) for the 250 testing samples.

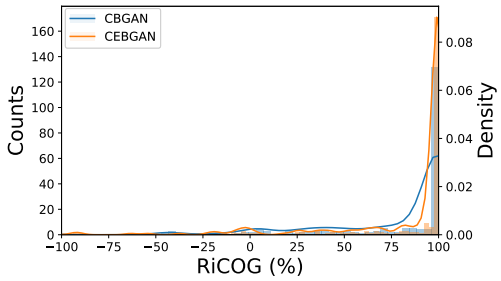


Fig. 10: Distribution of relative reduction in cumulative optimality gap (RiCOG) of the 250 testing samples using histograms (left y-axis) and KDE (right y-axis).

4.3.3 Practicability of the Surrogate Log-Likelihood

In the above example, we fixed the noise vector of the generator to demonstrate an example case. However, fixing this noise vector can only let us produce a single sample for each input condition. This becomes inappropriate when the diversity of predictions is critical or when inversion ambiguity exists. Multiple modes can correspond to the same input condition (*e.g.*, the conditional distribution in Fig. 4 or multiple, approximately equally good airfoils for a given set of boundary conditions). In that case, we need to randomly sample a large batch of noise vectors from the noise distribution $p(\mathbf{z})$ and use them to produce many valid samples from across all modes.

Assuming we can sample the generator, how do we distinguish samples with a higher likelihood of being high quality? Conceptually, these samples should lie within the vicinity of the conditional distribution modes, while low-performing samples should lie in the long tails of the distri-

Copyright © by ASME

bution. How can we assess this without running any further costly CFD simulations?

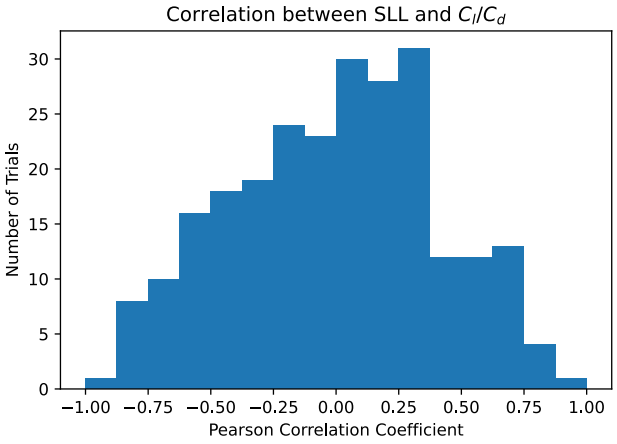


Fig. 11: The distribution of Pearson correlation coefficient between SLL of airfoil predictions and their actual C_l/C_d efficiencies.

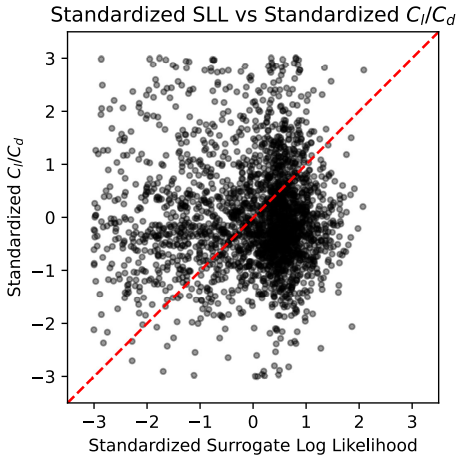


Fig. 12: The scatter plot of the normalized SLL of airfoil predictions and the corresponding normalized actual C_l/C_d efficiencies. No positive correlation can be observed.

As elucidated before, unlike traditional GANs, EGAN provides us with a surrogate log-likelihood which is essentially a lower bound of the sample's likelihood. It is then promising to use this surrogate value as an indicator of a sample's quality. That is, the higher the surrogate value, the better the quality, provided the EGAN converges well to the target conditional distribution, as demonstrated by Balaji *et al.* [54]. To verify this experimentally for CEBGAN, we randomly sample 10 airfoils for each of the 250 test input conditions and calculate the Pearson correlation coefficient between the surrogate values of the airfoils and their ground-truth C_l/C_d efficiencies, as introduced in Sec. 3.4.4. If the surrogate likelihood is a meaningfully useful quality measure, it should have a high correlation with the expensive-to-compute ground truth CFD simulation values.

For this task, we draw 4,096 samples from the distribution (Eqn. (17)) when evaluating the expectation in Eqn. (16). The distribution of these coefficients for the 250 test samples is illustrated in Fig. 11. Unfortunately, the correlations are almost symmetrically distributed around 0. This implies that the SLL does not usefully discriminate between high- and low-quality predictions for inverse airfoil design. We may also use a scatter plot to illustrate the correlation between SLL and C_l/C_d ratio more concretely. Before doing this, for each of the 250 input conditions, we need to normalize the SLL and the C_l/C_d ratio over all the 10 trials with mean and standard deviation to unify their range across input conditions and mitigate the distortion of outliers. Fig. 12 demonstrates the distribution of these normalized points, where we can neither notice any apparent positive correlation. More discussions on the possible causes are included in Sec. 5.

5 LIMITATIONS AND FUTURE WORK

Our inverse design methodology shows its potential in accelerating design optimization. However, before extrapolating it to the other tasks, one should be aware of several limitations of our data-driven inverse design paradigm.

First, because of the data-driven training process, our GANs' performance depends on the quality of the training dataset. Thus, if our optimal airfoil dataset obtained through the CFD adjoint method consists of many local optima, we should not expect our GANs to always yield global optima on either seen or unseen input conditions. We observe that our GANs can generate airfoils of higher performance than our targets in the dataset, so this local optima issue indeed exists in our current dataset. Likewise, the diversity of training samples directly affects the diversity of conditional GAN's generation. While generating the dataset, we used eight airfoils generated by the BézierGAN as our initialization points, but for at least some input conditions as shown in Fig. 2, the final optimized airfoils more or less have shapes similar to their original counterparts. It is impossible that these eight typical shapes can represent all the airfoils. Therefore, just like in many other design tasks, we made global optimality and diversity optional. Despite this limitation, our conditional GANs still demonstrate their ability to accurately approximate the posterior imperfectly represented by the dataset. This data-driven paradigm is still valuable in many tasks where global optimality and diversity of results are not mandatory.

One would likely then propose to increase the dataset size to overcome these issues. However, the optimal airfoil dataset generation is very time-consuming, and it is impractical to retrieve all optima. One possible solution is to use a low-dimensional design representation, as presented in [44], to accelerate the optimization process. Another more radical way is to discard this data generating process and use variational inference to directly learn the posterior, which could be an interesting research topic in the future.

Copyright © by ASME

In addition, the surrogate log likelihood—the lower bound to the exact sample log likelihood—cannot effectively differentiate between samples of good and bad quality. There are several hypotheses on this that might be worth probing in the future: 1. Sample likelihood is just not very related to sample's quality, let alone its lower bound approximation; 2. The lower bound is not tight enough to reveal the sample likelihood's positive correlation with sample quality; 3. This is cause by the incompleteness of the training dataset such that many designs of better performance are either downplayed or neglected in this empirical distribution, thus assigned lower likelihood by the CEGAN after training, but they somehow get recovered when generating the predictions.

Furthermore, this paper did not address two important factors that affect real-world use cases. First, we did not address scalability—the relation between the design problem's dimensionality and the number of data samples required to maintain a good approximation. Higher dimensions could occur from the design parameterization (*e.g.*, 3D vs 2D), the input conditions (*e.g.*, including heat transfer inputs), or the design objectives (*e.g.*, optimizing not only drag, but also manufacturability or vibration). Increasing any of these would complicate Inverse Design and require a larger number of data samples. Second, we did not address how to handle design constraints (aside from those not implicitly encoded in the training dataset). Future work could address how inverse design can more explicitly adapt and capture new design constraints without requiring retraining.

6 CONCLUSIONS

In this work, we employed two conditional GANs—CBGAN and CEBGAN—to approximate the posterior of the optimal airfoils conditioned on the freestream conditions and the target properties. Then, given unseen input conditions, we could generate warm start initialization points near the optima and accelerate the subsequent airfoil shape optimization. Our results show that both generative models can accomplish this task, but the CEBGAN—CEGAN based on regularized optimal transport and equipped with Bézier layer—performs uniformly better than that of vanilla CBGAN, either in terms of training speed or prediction accuracy among other metrics. CEGAN's advantage in approximating the multimodal distribution also manifests in simple toy examples. In addition, unlike traditional GANs, CEGAN also provides us with the unique ability to approximate the sample likelihood via a lower bound dubbed the surrogate log-likelihood, though its potential in selecting good samples remains to be investigated and uncovered.

Our GAN-based probabilistic inverse design paradigm is applicable to inverse airfoil design and the other inverse problems where high-dimensional posteriors need to be approximated. The Bézier layer we employed is suitable for curve-related design problems, but in other problems we can replace the Bézier layer with different architectures. For example, we can use a free-form deformation (FFD) layer for 3D airfoil design problems [63], or use standard 2D/3D convolutional layers for pixelated/voxelized designs [38]. The advantages brought by the conditional GAN framework and the entropic regularization should be invariant to such architecture adjustment. In future work, we will test our method on more applications to demonstrate this point. The performance and surrogate log-likelihood of CEGAN suggest it is a GAN more suitable for these posterior-retrieving tasks.

Though we solely focused on GANs' application in this inverse design work, it does not exclude the potential of using other generative models for similar tasks. Our preference for GANs is based on the widely alleged good quality of its generated samples in other works and the possibility of extending its power to more complicated inverse designs in the future. However, for many inverse design problems, other generative models like VAEs and flow-based models may also generate predictions of comparable quality, and given their more straightforward and accurate sample likelihood evaluation process, they may even be better candidates than GANs in certain tasks. This is worth investigating in the future.

Acknowledgements

This research was supported in part by funding from the U.S. Department of Energy's Advanced Research Projects Agency-Energy (ARPA-E) DIFFERENTIATE funding opportunity through award DE-AR0001216. The authors acknowledge the University of Maryland supercomputing resources (`http://hpcc.umd.edu`) made available for conducting the research reported in this paper.

References

[1] Tarantola, A., 2005. *Inverse problem theory and methods for model parameter estimation*. SIAM.

[2] Arridge, S., Maass, P., Öktem, O., and Schönlieb, C.-B., 2019. "Solving inverse problems using data-driven models". *Acta Numerica,* ***28***, pp. 1–174.

[3] Engl, H., Hanke, M., and Neubauer, A., 1996. *Regularization of Inverse Problems*. Mathematics and Its Applications. Springer Netherlands.

[4] Hornik, K., Stinchcombe, M., and White, H., 1989. "Multilayer feedforward networks are universal approximators". *Neural Networks,* ***2***(5), pp. 359–366.

[5] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2014. "Generative adversarial networks". *arXiv preprint arXiv:1406.2661*.

[6] Nowozin, S., Cseke, B., and Tomioka, R., 2016. "f-GAN: Training generative neural samplers using variational divergence minimization". *arXiv preprint arXiv:1606.00709*.

[7] 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 (ICCV).

[8] Arjovsky, M., Chintala, S., and Bottou, L., 2017. "Wasserstein generative adversarial networks". In In-

Copyright © by ASME

ternational conference on machine learning, PMLR, pp. 214–223.

[9] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A., 2017. "Improved training of wasserstein gans". *arXiv preprint arXiv:1704.00028*.

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

[11] Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M., 2016. "Improving variational inference with inverse autoregressive flow". *arXiv preprint arXiv:1606.04934*.

[12] Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P., 2016. "Variational lossy autoencoder". *arXiv preprint arXiv:1611.02731*.

[13] Mescheder, L., Nowozin, S., and Geiger, A., 2017. "Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks". In International Conference on Machine Learning, PMLR, pp. 2391–2400.

[14] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A., 2017. "beta-VAE: Learning basic visual concepts with a constrained variational framework". In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.

[15] Kim, H., and Mnih, A., 2018. "Disentangling by factorising". In International Conference on Machine Learning, PMLR, pp. 2649–2658.

[16] Dinh, L., Krueger, D., and Bengio, Y., 2014. "NICE: Non-linear independent components estimation". *arXiv preprint arXiv:1410.8516*.

[17] Dinh, L., Sohl-Dickstein, J., and Bengio, S., 2016. "Density estimation using Real NVP". *arXiv preprint arXiv:1605.08803*.

[18] Kingma, D. P., and Dhariwal, P., 2018. "Glow: Generative flow with invertible 1x1 convolutions". *arXiv preprint arXiv:1807.03039*.

[19] Wiecha, P., Arbouet, A., Girard, C., and Muskens, O., 2021. "Deep learning in nano-photonics: inverse design and beyond". *Photonics Research*, Jan.

[20] Liu, Z., Zhu, D., Rodrigues, S. P., Lee, K.-T., and Cai, W., 2018. "Generative model for the inverse design of metasurfaces". *Nano Letters,* ***18***(10), pp. 6570–6576. PMID: 30207735.

[21] So, S., and Rho, J., 2019. "Designing nanophotonic structures using conditional deep convolutional generative adversarial networks". *Nanophotonics,* ***8***(7), pp. 1255–1261.

[22] Jiang, J., Sell, D., Hoyer, S., Hickey, J., Yang, J., and Fan, J. A., 2019. "Free-form diffractive metagrating design based on generative adversarial networks". *ACS Nano,* ***13***(8), pp. 8872–8878. PMID: 31314492.

[23] Jiang, J., and Fan, J. A., 2020. "Simulator-based training of generative neural networks for the inverse design of metasurfaces". *Nanophotonics,* ***9***(5), pp. 1059–1069.

[24] Dong, Y., Li, D., Zhang, C., Wu, C., Wang, H., Xin, M., Cheng, J., and Lin, J., 2020. "Inverse design of two-dimensional graphene/h-bn hybrids by a regressional and conditional gan". *Carbon,* ***169***, pp. 9–16.

[25] Wang, J., Chen, W., Fuge, M., and Rai, R., 2021. "Ih-gan: A conditional generative model for implicit surface-based inverse design of cellular structures". *arXiv preprint arXiv:2103.02588*.

[26] Kim, B., Lee, S., and Kim, J., 2020. "Inverse design of porous materials using artificial neural networks". *Science Advances,* ***6***(1).

[27] Kim, S., Noh, J., Gu, G. H., Aspuru-Guzik, A., and Jung, Y., 2020. "Generative adversarial networks for crystal structure prediction". *ACS Central Science,* ***6***(8), pp. 1412–1420.

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

[29] Sanchez-Lengeling, B., and Aspuru-Guzik, A., 2018. "Inverse molecular design using machine learning: Generative models for matter engineering". *Science,* ***361***(6400), pp. 360–365.

[30] Adler, J., and Öktem, O., 2018. "Deep bayesian inversion". *arXiv preprint arXiv:1811.05910*.

[31] Ongie, G., Jalal, A., Metzler, C. A., Baraniuk, R. G., Dimakis, A. G., and Willett, R., 2020. "Deep learning techniques for inverse problems in imaging". *IEEE Journal on Selected Areas in Information Theory,* ***1***(1), pp. 39–56.

[32] Wang, X., Ghasedi Dizaji, K., and Huang, H., 2018. "Conditional generative adversarial network for gene expression inference". *Bioinformatics,* ***34***(17), 09, pp. i603–i611.

[33] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A., 2017. "Image-to-image translation with conditional adversarial networks". In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1125–1134.

[34] Smyl, D., 2018. "An inverse method for optimizing elastic properties considering multiple loading conditions and displacement criteria". *Journal of Mechanical Design,* ***140***(11).

[35] Bostanabad, R., Chan, Y.-C., Wang, L., Zhu, P., and Chen, W., 2019. "Globally approximate gaussian processes for big data with application to data-driven metamaterials design". *Journal of Mechanical Design,* ***141***(11).

[36] Lee, X. Y., Balu, A., Stoecklein, D., Ganapathysubramanian, B., and Sarkar, S., 2019. "A case study of deep reinforcement learning for engineering design: Application to microfluidic devices for flow sculpting". *Journal of Mechanical Design,* ***141***(11).

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

[38] Oh, S., Jung, Y., Kim, S., Lee, I., and Kang, N., 2019.

 Copyright © by ASME

"Deep generative design: Integration of topology optimization and generative models". *Journal of Mechanical Design,* ***141***(11).

[39] 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". *Journal of Mechanical Design,* ***142***(7).

[40] Yilmaz, E., and German, B., 2020. "Conditional generative adversarial network framework for airfoil inverse design". In AIAA AVIATION 2020 FORUM, p. 3185.

[41] 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, p. 2261.

[42] Yang, G., and Da Ronch, A., 2018. "Aerodynamic shape optimisation of benchmark problems using SU2". In 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, p. 0412.

[43] Menčík, J., 2016. "Latin hypercube sampling". *Concise Reliability for Engineers*, p. 117.

[44] Chen, W., Chiu, K., and Fuge, M., 2020. "Airfoil design parameterization and optimization using bézier generative adversarial networks". *AIAA Journal*.

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

[46] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P., 2016. "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets". *arXiv preprint arXiv:1606.03657*.

[47] Arjovsky, M., and Bottou, L., 2017. "Towards principled methods for training generative adversarial networks". *arXiv preprint arXiv:1701.04862*.

[48] Goodfellow, I., 2016. "Nips 2016 tutorial: Generative adversarial networks". *arXiv preprint arXiv:1701.00160*.

[49] Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J., 2016. "Unrolled generative adversarial networks". *arXiv preprint arXiv:1611.02163*.

[50] Mirza, M., and Osindero, S., 2014. "Conditional generative adversarial nets". *arXiv preprint arXiv:1411.1784*.

[51] Genevay, A., Peyré, G., and Cuturi, M., 2018. "Learning generative models with sinkhorn divergences". In International Conference on Artificial Intelligence and Statistics, PMLR, pp. 1608–1617.

[52] Sanjabi, M., Ba, J., Razaviyayn, M., and Lee, J. D., 2018. "On the convergence and robustness of training gans with regularized optimal transport". *arXiv preprint arXiv:1802.08249*.

[53] Salimans, T., Zhang, H., Radford, A., and Metaxas, D., 2018. "Improving GANs using optimal transport". *arXiv preprint arXiv:1803.05573*.

[54] Balaji, Y., Hassani, H., Chellappa, R., and Feizi, S., 2019. "Entropic GANs meet VAEs: A statistical approach to compute sample likelihoods in GANs". In International Conference on Machine Learning, PMLR, pp. 414–423.

[55] Pope, P., Zhu, C., Abdelkader, A., Goldblum, M., and Goldstein, T., 2021. "The intrinsic dimension of images and its impact on learning". In International Conference on Learning Representations.

[56] Cuturi, M., 2013. "Sinkhorn distances: lightspeed computation of optimal transport". In NIPS, Vol. 2, p. 4.

[57] Peyré, G., Cuturi, M., et al., 2019. "Computational optimal transport: With applications to data science". *Foundations and Trends® in Machine Learning,* ***11***(5-6), pp. 355–607.

[58] Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouvé, A., and Peyré, G., 2019. "Interpolating between optimal transport and mmd using sinkhorn divergences". In The 22nd International Conference on Artificial Intelligence and Statistics, PMLR, pp. 2681–2690.

[59] Theis, L., Oord, A. v. d., and Bethge, M., 2015. "A note on the evaluation of generative models". *arXiv preprint arXiv:1511.01844*.

[60] Smola, A., Gretton, A., Song, L., and Schölkopf, B., 2007. "A hilbert space embedding for distributions". In International Conference on Algorithmic Learning Theory, Springer, pp. 13–31.

[61] Bishop, C. M., 2006. *Pattern Recognition and Machine Learning*. Springer.

[62] Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., and Alonso, J. J., 2016. "Su2: An open-source suite for multiphysics simulation and design". *AIAA Journal,* ***54***(3), pp. 828–846.

[63] Chen, W., and Ramamurthy, A., 2021. "Deep generative model for efficient 3d airfoil parameterization and generation". In AIAA Scitech 2021 Forum, p. 1690.

Copyright © by ASME

Supplemental Material: Inverse Design of 2D Airfoils using Conditional Generative Models

Qiuyi Chen[1]*, **Jun Wang**[1], **Phillip Pope**[2], **Wei Chen**[3], **Mark Fuge**[1]
[1]Department of Mechanical Engineering, University of Maryland, College Park, Maryland 14260
[2]Department of Computer Science, University of Maryland, College Park, Maryland 14260
[3]Siemens Technology, Princeton, NJ 08540

1 Proof of Explicit Density Model and the Derivation of Conditional Surrogate Log-Likelihood

The proof follows the same logic as the unconditional one [1], with some variances.

1.1 Proposition

Let $\mathbf{y}$, $\hat{\mathbf{y}}$ be the input condition sampled from the marginal distribution $p_r(\mathbf{y})$ of the real data distribution $p_r(\mathbf{x},\mathbf{y})$, $\mathbf{z}$ be the noise sampled from $p_Z(\mathbf{z})$, $\hat{\mathbf{x}} = G(\mathbf{z},\hat{\mathbf{y}})$ be the generator G's prediction on $\hat{\mathbf{y}}$, and $\mathbf{x}$ be the output corresponding to $\mathbf{y}$. Training CEGAN is equivalent to maximizing a lower bound of the sample likelihood of this latent model below:

$$p(\mathbf{x},\mathbf{y}) = \int_{\hat{\mathbf{y}},\mathbf{z}} p(\mathbf{x},\mathbf{y} \mid \hat{\mathbf{y}},\mathbf{z})\, p_r(\hat{\mathbf{y}})\, p_Z(\mathbf{z})\, d\hat{\mathbf{y}}\, d\mathbf{z} \tag{S1}$$

$$p(\mathbf{x},\mathbf{y} \mid \hat{\mathbf{y}},\mathbf{z}) = C \cdot \exp - \frac{c([\mathbf{x},\mathbf{y}],[G(\mathbf{z},\hat{\mathbf{y}}),\hat{\mathbf{y}}])}{\lambda} \tag{S2}$$

where C is a constant for normalization ensured by the shift invariant cost function c, and λ is the weight controlling the degree of entropic regularization.

1.2 Proof

From Bayes' rule, we have

$$p(\mathbf{x},\mathbf{y}) = \frac{p(\mathbf{x},\mathbf{y} \mid \hat{\mathbf{y}},\mathbf{z})\, p_r(\hat{\mathbf{y}})\, p_Z(\mathbf{z})}{p(\hat{\mathbf{y}},\mathbf{z} \mid \mathbf{x},\mathbf{y})} \tag{S3}$$

Take the logarithm, then

$$\log p(\mathbf{x},\mathbf{y}) = \log p(\mathbf{x},\mathbf{y} \mid \hat{\mathbf{y}},\mathbf{z}) + \log p_r(\hat{\mathbf{y}}) + \log p_Z(\mathbf{z}) - \log p(\hat{\mathbf{y}},\mathbf{z} \mid \mathbf{x},\mathbf{y}) \tag{S4}$$

Consider a joint distribution $\mathbb{P}_{X,Y,\hat{Y},Z}$ of marginal distributions $\mathbb{P}_{X,Y}$ and $\mathbb{P}_{\hat{Y},Z}$, which are empirical distributions formed

*corresponding author. e-mail: qchen88@umd.edu

by the batch sampled from $p_r(\mathbf{x},\mathbf{y})$ and $p_Z(\mathbf{z})$. Taking expectation of both sides w.r.t. $\mathbb{P}_{\hat{Y},Z|X,Y}$, we get

$$\begin{aligned}\log p(\mathbf{x},\mathbf{y}) &= \mathbb{E}_{\mathbb{P}_{\hat{Y},Z|X,Y}}\left[\log p(\mathbf{x},\mathbf{y}\mid\hat{\mathbf{y}},\mathbf{z})+\log p_r(\hat{\mathbf{y}})+\log p_Z(\mathbf{z})-\log p(\hat{\mathbf{y}},\mathbf{z}\mid\mathbf{x},\mathbf{y})\right]\\ &= -\frac{1}{\lambda}\mathbb{E}_{\mathbb{P}_{\hat{Y},Z|X,Y}}\left[c([\mathbf{x},\mathbf{y}],[G(\mathbf{z},\hat{\mathbf{y}}),\hat{\mathbf{y}}])\right]+\log C+\mathbb{E}_{\mathbb{P}_{\hat{Y}|X,Y}}\left[\log p_r(\hat{\mathbf{y}})\right]+\mathbb{E}_{\mathbb{P}_{Z|X,Y}}\left[\log p_Z(\mathbf{z})\right]\\ &\qquad +\mathrm{KL}\left(\mathbb{P}_{\hat{Y},Z|X,Y}\,\|\,p_{\hat{Y},Z|X,Y}\right)+H\left(\mathbb{P}_{\hat{Y},Z|X,Y}\right) \quad \text{(S5)}\end{aligned}$$

Next take expectation of both sides w.r.t. $\mathbb{P}_{X,Y}$, then

$$\begin{aligned}\mathbb{E}_{\mathbb{P}_{X,Y}}\left[\log p(\mathbf{x},\mathbf{y})\right] &= -\frac{1}{\lambda}\mathbb{E}_{\mathbb{P}_{X,Y,\hat{Y},Z}}\left[c([\mathbf{x},\mathbf{y}],[G(\mathbf{z},\hat{\mathbf{y}}),\hat{\mathbf{y}}])\right]+\log C+\mathbb{E}_{\mathbb{P}_{\hat{Y}}}\left[\log p_r(\hat{\mathbf{y}})\right]+\mathbb{E}_{\mathbb{P}_Z}\left[\log p_z(\mathbf{z})\right]\\ &\qquad +\mathbb{E}_{\mathbb{P}_{X,Y}}\left[\mathrm{KL}\left(\mathbb{P}_{\hat{Y},Z|X,Y}\,\|\,p_{\hat{Y},Z|X,Y}\right)\right]+H\left(\mathbb{P}_{X,Y,\hat{Y},Z}\right)-H\left(\mathbb{P}_{X,Y}\right) \quad \text{(S6)}\end{aligned}$$

Then after using the data processing inequality $H\left(\mathbb{P}_{X,Y,\hat{Y},Z}\right)\geq H\left(\mathbb{P}_{X,Y,\hat{X},\hat{Y}}\right)$, we have

$$\begin{aligned}\mathbb{E}_{\mathbb{P}_{X,Y}}\left[\log p(\mathbf{x},\mathbf{y})\right] &\geq -\frac{1}{\lambda}\left\{\mathbb{E}_{\mathbb{P}_{X,Y,\hat{Y},Z}}\left[c([\mathbf{x},\mathbf{y}],[G(\mathbf{z},\hat{\mathbf{y}}),\hat{\mathbf{y}}])\right]-\lambda H\left(\mathbb{P}_{X,Y,\hat{X},\hat{Y}}\right)\right\}\\ &\qquad +\log C+\mathbb{E}_{\mathbb{P}_{\hat{Y}}}\left[\log p_r(\hat{\mathbf{y}})\right]+\mathbb{E}_{\mathbb{P}_Z}\left[\log p_z(\mathbf{z})\right]-H\left(\mathbb{P}_{X,Y}\right) \quad \text{(S7)}\end{aligned}$$

where for each batch of fixed $\mathbb{P}_{X,Y}$ and $\mathbb{P}_{\hat{Y},Z}$ the last four terms are constant and the term within the curly bracket is exactly the objective of CEGAN with the KL divergence regularization replaced by the equivalent entropy regularization. □

1.3 Conditional Surrogate Log-Likelihood

The RHS of (S5) excluding the KL divergence is a lower bound that can serve as a surrogate. However, the term $\mathbb{E}_{\mathbb{P}_{\hat{Y}|X,Y}}\left[\log p_r(\hat{\mathbf{y}})\right]$ in (S5) is infeasible since the expression of p_r is unknown. For a $\log(\mathbf{x},\mathbf{y})$ to be estimated, if we take $\mathbf{y}$ as a *constant*, then as a work-around for deriving the surrogate log-likelihood, since there is no restriction on the choice of $\mathbb{P}_{\hat{Y},Z|X,Y}$, we can design it as follows to circumvent this issue:

$$\mathbb{P}_{\hat{Y},Z|X,Y}=\mathbb{P}_{\hat{Y}|X,Y}\,\mathbb{P}_{Z|X,Y,\hat{Y}}=\mathbb{P}_{\hat{Y}|Y}\,\mathbb{P}_{Z|X}=\mathbf{1}_{\hat{Y}|Y}\,\mathbb{P}_{Z|X} \quad \text{(S8)}$$

where $\mathbf{1}_{\hat{Y}|Y}$ is the single point mass distribution centered at the realization of Y.

Now take expectation of both sides of (S5) w.r.t. $\mathbb{P}_{\hat{Y},Z|X,Y}=\mathbf{1}_{\hat{Y}|Y}\,\mathbb{P}_{Z|X}$, we get

$$\begin{aligned}\log p(\mathbf{x},\mathbf{y}) &= -\frac{1}{\lambda}\mathbb{E}_{\mathbb{P}_{Z|X}}\left[c([\mathbf{x},\mathbf{y}],[G^\star(\mathbf{z},\mathbf{y}),\mathbf{y}])\right]+\log C+\log p_r(\mathbf{y})+\mathbb{E}_{\mathbb{P}_{Z|X}}\left[\log p_Z(\mathbf{z})\right]\\ &\qquad +\mathrm{KL}\left(\mathbf{1}_{\hat{Y}|Y}\,\mathbb{P}_{Z|X}\,\|\,p_{\hat{Y},Z|X,Y}\right)+H\left(\mathbb{P}_{Z|X}\right) \quad \text{(S9)}\end{aligned}$$

Take expectation of both sides w.r.t. $\mathbb{P}_X=\mathbf{1}_{X=\mathbf{x}}$, then

$$\begin{aligned}\log p(\mathbf{x},\mathbf{y}) &= -\frac{1}{\lambda}\mathbb{E}_{\mathbb{P}_{X,Z}}\left[c([\mathbf{x},\mathbf{y}],[G^\star(\mathbf{z},\mathbf{y}),\mathbf{y}])\right]+\log C+\log p_r(\mathbf{y})+\mathbb{E}_{\mathbb{P}_Z}\left[\log p_Z(\mathbf{z})\right]\\ &\qquad +\mathrm{KL}\left(\mathbf{1}_{\hat{Y}|Y}\,\mathbb{P}_{Z|X}\,\|\,p_{\hat{Y},Z|X,Y}\right)+H\left(\mathbb{P}_{X,Z}\right)\\ &\geq -\frac{1}{\lambda}\left\{\mathbb{E}_{\mathbb{P}_{X,Z}}\left[c([\mathbf{x},\mathbf{y}],[G^\star(\mathbf{z},\mathbf{y}),\mathbf{y}])\right]-\lambda H\left(\mathbb{P}_{X,Z}\right)\right\}+\log C+\log p_r(\mathbf{y})+\mathbb{E}_{\mathbb{P}_Z}\left[\log p_Z(\mathbf{z})\right] \quad \text{(S10)}\end{aligned}$$

While the last three terms are constant, the term inside the bracket is the objective of an optimal transport problem regularized

with entropy that is minimized when $\mathbb{P}_{X,Z} = \mathbb{P}^{\star}_{X,Z}$, which has the closed form as per [1]:

$$\mathbb{P}^{\star}_{X,Z} = \mathbb{P}_X \mathbb{P}_Z \exp \frac{v^{\star}([\mathbf{x},\mathbf{y}],[G^{\star}(\mathbf{z},\mathbf{y}),\mathbf{y}])}{\lambda} = \mathbf{1}_{X=\mathbf{x}} \mathbb{P}_Z \exp \frac{v^{\star}([\mathbf{x},\mathbf{y}],[G^{\star}(\mathbf{z},\mathbf{y}),\mathbf{y}])}{\lambda} \tag{S11}$$

Therefore we can take (S10) as our surrogate log-likelihood when we set $\mathbb{P}_{X,Z} = \mathbb{P}^{\star}_{X,Z}$. □

2 Hyperparameters of CEBGAN

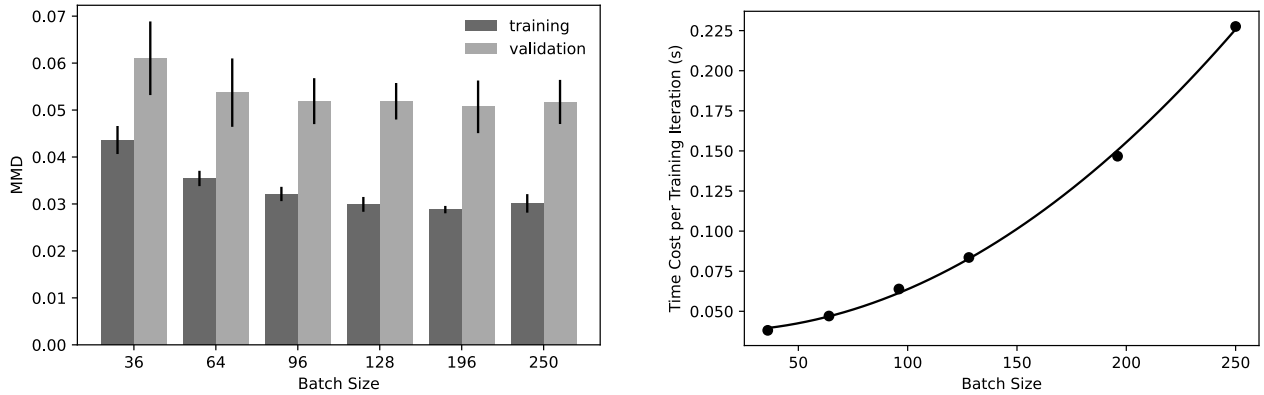
Since there is abundant literature on the hyperparameter tuning of vanilla GANs, and we inherited the neural network architectures of CBGAN and CEBGAN from our previous work on unconditional BezierGAN [2], in this section we focus only on the hyperparameter tuning of CEBGAN (or EGAN with Sinkhorn divergence, on a larger scope), which is not covered by many existing materials. Specifically, we concentrate on the effect of batch size, regularization weight λ, cost function c and the number of epochs, which are either unique to EGAN or have distinctive effect. The other part of the hyperparameter configuration can be found in our code at https://github.com/IDEALLab/CEBGAN_JMD_2021.

Throughout these tuning experiments, we use the MMD function introduced in §3.4.1 as our metric, which is a standard choice and fast for evaluating the discrepancy between complicated distributions. We implement 4-fold cross validation on the complete training set of 995 airfoil samples for each hyperparameter configuration to take into account any disturbances from stochasticity, such as initialization, mini-batch gradient descent, *etc.* In other words, for each fold of the cross validation, 747 samples out of the 995 are for training the CEBGAN while the rest 248 are reserved for validation.

Another thing we need to clarify is the difference between the *number of epochs* and the *number of iterations*. Since our code is written in PyTorch, in each epoch we leverage the DataLoader constructor in PyTorch to generate mini-batches for gradient descent. For a training set of m samples, if we set the batch size to n, the DataLoader will divide it into $\lceil m/n \rceil$ batches for inner loop iteration. Therefore, the number of training iterations for N epochs equals $N \times \lceil m/n \rceil$.

2.1 Batch Size

Six batch size candidates—**36, 64, 96, 128, 196, 250**—are selected to illustrate the effect of batch size on the final prediction, with their number of iterations set equally to 42,000. The value of λ is set to 5, while the cost function c is constructed using L_1 distance, as introduced in §3.3.4. The MMD result is plotted in Fig. S1a, where the height of each bar indicates the mean of MMD across the 4 folds and the error line on top indicates its standard deviation.



(a) MMD

(b) Training Time Cost

Fig. S1: MMD and training iteration time cost against batch size.

We can see that in general, the larger the batch size, the smaller the MMD, and this improvement saturates as the batch size increases. Intuitively speaking, in each training iteration, we use the mini-batch to approximately represent the data distribution p_r, hoping that by minimizing the $S_\lambda(p_r, p_g)$ evaluated with this mini-batch, p_g converges to p_r eventually. The larger the batch size, the more accurately the mini-batches can represent the true data distribution, therefore the more precisely p_g can be driven to p_r. And just like a regular machine learning practice, because of the discrepancy between the training set and the validation set, inevitably the CEBGAN performs not as good on the validation set as on the training set.

Yet this precision improvement entails an increase in the computational cost. The time complexity of the Sinkhorn algorithm is $\tilde{O}(n^2/\varepsilon^3)$ [3], so the computational cost scales quadratically with the batch size. Hence the final choice of the batch size is a trade-off between precision and time cost. This trend is demonstrated in Fig. S1b, where we use quadratic regression to fit the average time cost per CEBGAN training iteration and it seems accurate. We select 128 as our final batch size, which has an MMD of 0.030 ± 0.0016 on the training set and an MMD of 0.052 ± 0.0039 on the validation set, while the time cost is acceptable to us.

2.2 Regularization Weight λ

This weight controls the strength of entropic regularization, as introduced in §3.3.1. Theoretically, it manipulates Sinkhorn divergence S_λ's interpolation between optimal transport distance (OT) and MMD [4, 5]—$S_\lambda \to$ OT as $\lambda \to 0$ and $S_\lambda \to$ MMD as $\lambda \to \infty$. In consequence, λ leads to the similar trade-off between precision and time cost [4, 6, 7]—as λ increases, the Sinkhorn algorithm evaluating S_λ converges faster, but minimizing such S_λ leads to lower distribution learning precision, and *vice versa*. Setting this λ properly can achieve a good balance between both worlds.

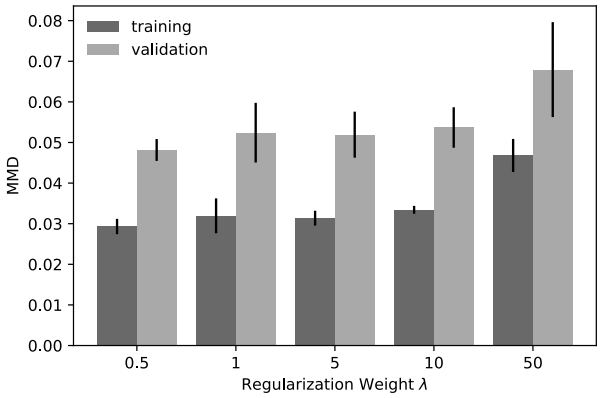


Fig. S2: MMD against λ.

We select the final λ out of five candidates—**0.5, 1, 5, 10, 50**. The experiment setting is similar to the previous one, except we now fix the batch size at 128. The result is shown in Fig. S2 in the same manner as before. We can see that in general the aforementioned trend holds, a smaller λ leads to better performance, but the improvement seems marginal as λ goes under 10. It is also interesting to notice that although a S_λ with larger λ is closer to MMD, minimizing it results in an MMD larger than that of minimizing one with smaller λ, probably due to the vanishing gradient issue associated with MMD [6]. Therefore, minimizing the metric directly as the loss function is not always a wise choice. In terms of time cost, under our configuration, training the CEBGAN with $S_{\lambda=5}$ takes nearly half as much time as with $S_{\lambda=0.5}$, while it takes the similar amount of time as with $\lambda = 50$. We hence choose $\lambda = 5$ as our final setting.

2.3 Cost Function c

This cost function c is the one involved in the definition of regularized optimal transport distance (Eqn. 7). We only investigate the ones constructed with L_1 and L_2 distance here, as they are quite common choices among the existing materials and very easy to realize. Although it is alleged that an adaptive cost function [4]—which should be trained throughout EGAN's training process to adapt to the dataset—could further improve the performance of EGANs, we do not implement it in this work, given its greater complexity, larger time cost, the already satisfactory performance of the ordinary cost functions, and that it may not be compatible with the surrogate log likelihood.

Specifically, in the experiment we construct two candidate cost functions by:

$$c_1([\mathbf{x},\mathbf{y},\mathbf{b}],[\hat{\mathbf{x}},\hat{\mathbf{y}},\hat{\mathbf{b}}]) = \|\mathbf{x}-\hat{\mathbf{x}}\|_1 + \|\mathbf{y}-\hat{\mathbf{y}}\|_1 + \|\mathbf{b}-\hat{\mathbf{b}}\|_1$$
$$c_2([\mathbf{x},\mathbf{y},\mathbf{b}],[\hat{\mathbf{x}},\hat{\mathbf{y}},\hat{\mathbf{b}}]) = \|\mathbf{x}-\hat{\mathbf{x}}\|_2 + \|\mathbf{y}-\hat{\mathbf{y}}\|_2 + \|\mathbf{b}-\hat{\mathbf{b}}\|_2$$

and set the batch size to 128, λ to 5 and the number of iterations to 42000, the same as before. The result is plotted in Fig. S3. We can see the cost function c_1 performs better, so we make it our final choice.

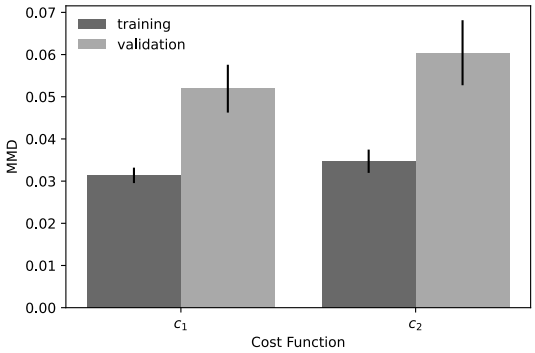
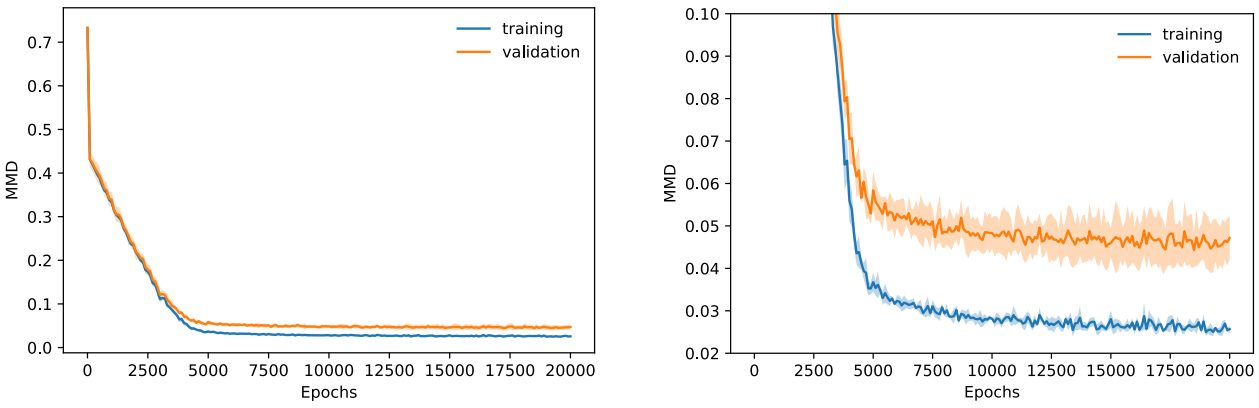


Fig. S3: MMD against cost function.

2.4 Number of Epochs

The number of training epochs can significantly affect performance: too few may lead to underfitting whereas too many may lead to overfitting. To check this on our inverse airfoil design application, for the final choice of hyperparameters—batch size of 128, $\lambda = 5$, cost function set to c_1—we increase the number of iterations from 42000 to 120000 and plot the MMD history throughout the training process. To achieve this, for each of the four cross validation folds, we correspondingly increase the number of epochs from 7000 to 20000, and record the MMD every 100 epochs.

The result is shown in Fig. S4, with the full scale history on the left and the magnified plot on the right to reveal its asymptotic behavior. The shades cover the $\pm$StdDev region. We can see that the MMDs rapidly drop before epoch #5000 on both the training set and the validation set, then gradually converge to their limits, with a smaller variance on the training set and a larger one on the validation set. There is generally no uptick in the MMD on the validation set, which suggests the absence of overfitting. Since the improvement after epoch #15000 is marginal, and given that once we use all the 995 samples to training the final model, there will be more training iterations for the same number of epochs, we train our final model with 15000 epochs.



(a) Full Scale

(b) Magnified

Fig. S4: MMD history throughout the training process.

References

[1] Y. Balaji, H. Hassani, R. Chellappa, S. Feizi, Entropic gans meet vaes: A statistical approach to compute sample likelihoods in gans (2019). arXiv:1810.04147.

[2] W. Chen, K. Chiu, M. D. Fuge, Airfoil design parameterization and optimization using bézier generative adversarial networks, AIAA Journal 58 (11) (2020) 4723–4735.

[3] J. Altschuler, J. Weed, P. Rigollet, Near-linear time approximation algorithms for optimal transport via sinkhorn iteration, arXiv preprint arXiv:1705.09634 (2017).

[4] A. Genevay, G. Peyré, M. Cuturi, Learning generative models with sinkhorn divergences, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2018, pp. 1608–1617.

[5] A. Ramdas, N. G. Trillos, M. Cuturi, On wasserstein two-sample testing and related families of nonparametric tests, Entropy 19 (2) (2017) 47.
[6] J. Feydy, T. Séjourné, F.-X. Vialard, S.-i. Amari, A. Trouvé, G. Peyré, Interpolating between optimal transport and mmd using sinkhorn divergences, in: The 22nd International Conference on Artificial Intelligence and Statistics, PMLR, 2019, pp. 2681–2690.
[7] M. Cuturi, Sinkhorn distances: Lightspeed computation of optimal transport, Advances in neural information processing systems 26 (2013) 2292–2300.