Proceedings of the ASME 2018 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2018
August 26-29, 2018, Quebec City, Quebec, Canada

DETC2018-85339

SYNTHESIZING DESIGNS WITH INTER-PART DEPENDENCIES USING HIERARCHICAL GENERATIVE ADVERSARIAL NETWORKS

Wei Chen[*]
Dept. of Mechanical Engineering
University of Maryland
College Park, Maryland 20742
Email: wchen459@umd.edu

Ashwin Jeyaseelan
Dept. of Computer Science
University of Maryland
College Park, Maryland 20742
Email: ashwinjey@gmail.com

Mark Fuge
Dept. of Mechanical Engineering
University of Maryland
College Park, Maryland 20742
Email: fuge@umd.edu

ABSTRACT

Real-world designs usually consist of parts with hierarchical dependencies, i.e.*, the geometry of one component (a child shape) is dependent on another (a parent shape). We propose a method for synthesizing this type of design. It decomposes the problem of synthesizing the whole design into synthesizing each component separately but keeping the inter-component dependencies satisfied. This method constructs a two-level generative adversarial network to train two generative models for parent and child shapes, respectively. We then use the trained generative models to synthesize or explore parent and child shapes separately via a parent latent representation and infinite child latent representations, each conditioned on a parent shape. We evaluate and discuss the disentanglement and consistency of latent representations obtained by this method. We show that shapes change consistently along any direction in the latent space. This property is desirable for design exploration over the latent space.*

INTRODUCTION

Representing a high-dimensional design space with a lower-dimensional *latent space* makes it easier to explore, visualize, or optimize complex designs. This often means finding a *latent representation*, or a *manifold*, along which valid design geometries morph [1,2].

While this works well for single parts, designs usually have hierarchical structures. For example, the size and position of a (conduit, lightening, or alignment) hole in an airfoil depend on the shape of the airfoil, thus there is a hierarchical relationship between the airfoil and the hole in it. Here the airfoil is called the *parent shape*, and the hole is called the *child shape*. In this case, one may want to identify first the *parent manifold* that captures major variation of parent shapes, and then the *child manifold* that captures major variation of feasible child shapes conditioned on any parent shape (Fig. 1). Because, for example, we may first optimize the airfoil shape on the airfoil manifold (parent) to obtain the optimal lift and drag; and then given the optimal airfoil, we may optimize the hole's size and position on the hole manifold (child) for other consideration.

However, finding individual part manifolds that both represent the design space well, while also satisfying part configuration, is non-trivial. To learn the inter-part dependency, one can define explicit constraints [3,4] or learn implicit constraints via adaptive sampling [2,5]. The former uses hard-coded constraints and hence lacks flexibility; whereas the latter queries external sources by human annotation, experiment, or simulation, thus is expensive. In this paper, we solve the problem by identifying different levels of manifolds, where the higher-level manifold imposes implicit constraints on the lower-level manifolds. Our main contribution is a deep generative model that synthesizes designs in a hierarchical manner: it first synthesizes the parent shape, and then the child shape conditioned on its parent. The model simultaneously captures a parent manifold and infinite child manifolds which are conditioned on parent shapes.

[*]Address all correspondence to this author.

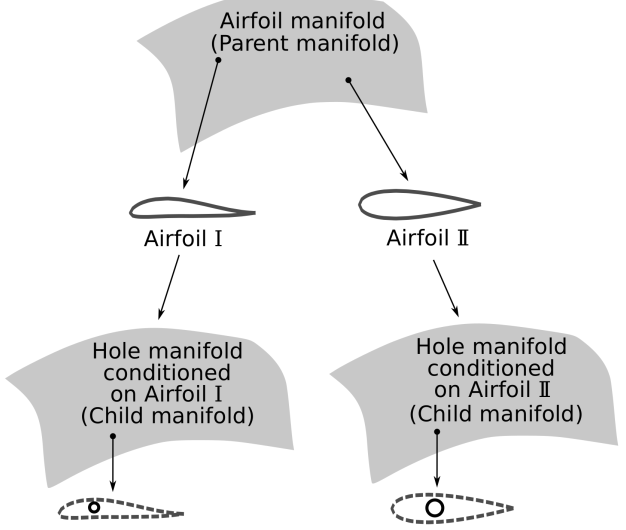

Figure 1: MANIFOLDS OF PARENT AND CHILD SHAPES.

This results in two latent spaces allowing synthesis and searching of parent and child shapes. Our method is fully data-driven and requires no hard-coded rules, querying external sources, or complex preprocessing [6], except for specifying parent and child shapes in the dataset.

RELATED WORK

Our work produces generative models that synthesize designs from latent representations. There are primarily two streams of related research—design space dimensionality reduction and design synthesis—from the fields of engineering design and computer graphics. We also review generative adversarial nets (GAN) [7], which we use to build our model.

Design Space Dimensionality Reduction

While designs can be parametrized by various techniques [8], the number of design variables (*i.e.*, the dimensionality of a design space) increases with the geometric variability of designs. In tasks like design optimization, to find better designs we usually need a design space with higher variability, *i.e.*, higher dimensionality. This demand brings up the problem of exploring a high-dimensional design space. Based on the curse of dimensionality [9], the cost of exploring the design space grows exponentially with its dimensionality. Thus researchers have studied approaches of reducing the design space dimensionality. Normally, dimensionality reduction methods identify a lower-dimensional latent space that captures most of the design space's variability. This can be grouped into linear and non-linear methods.

Linear dimensionality reduction methods select a set of optimal directions or basis functions where the variance of shape geometry or certain simulation output is maximized. Such methods includes the Karhunen-Loève expansion (KLE) [10, 11], principal component analysis (PCA) [12], and the active subspaces approach [13].

In practice, it is more reasonable to assume that design variables lie on a non-linear manifold, rather than a hyperplane. Thus researchers also apply non-linear methods to reduce the dimensionality of design spaces. This non-linearity can be achieved by 1) applying linear reduction techniques locally to construct a non-linear global manifold [14, 15, 16, 17, 12]; 2) using kernel methods with linear reduction techniques [1, 12]; 3) latent variable models like Gaussian process latent variable model (GPLVM) and generative topographic mapping (GTM) [18]; and 4) neural networks based approaches such as self-organizing maps [19] and autoencoders [20, 1, 12, 21].

This work differs from these approaches in that we aim at identifying two-level latent spaces with the lower-level encodes inter-part dependencies, rather than learning only one latent space for the complete design.

Data-Driven Design Synthesis

Design synthesis methods can be divided into two categories: rule-based and data-driven design synthesis. The former (*e.g.*, grammars-based design synthesis [22, 4, 3]) requires labeling of the reference points or surfaces and defining rule sets, so that new designs are synthesized according to this hard-coded prior knowledge; while the latter learns rules/constraints from a database and generates plausible new designs with similar structure/function to exemplars in the database.

Usually dimensionality reduction techniques allow inverse transformations from the latent space back to the design space, thus can synthesize new designs from latent variables [11, 1, 12, 21]. For example, under the PCA model, the latent variables define a linear combination of principal components to synthesize a new design [11]; for local manifold based approaches, a new design can be synthesized via interpolation between neighboring points on the local manifold [17]; and under the autoencoder model, the trained decoder maps any given point in the latent space to a new design [20, 21].

Methods for synthesizing 3D models are frequently studied by the field of computer graphics. Generally, researchers have employed generative models such as kernel density estimation [23], Boltzmann machines [24], variational autoencoders (VAEs) [25], and generative adversarial nets (GANs) [26, 27] to learn the distribution of samples in the design space, and synthesize new designs by drawing samples from the learned distribution. Discriminative models like deep residual networks [28] are also used to generate 3D shapes.

There are also methods that synthesize new shapes by assembling or reorganizing parts from an existing shape database, while preserving the desired structures [29, 6, 30, 31, 32]. The shapes are usually parameterized by high-level abstract representations, such as hand-crafted feature vectors [6] or shape grammars [30]. While these methods edit shapes at a high-level, they do not control the local geometry of each synthesized component.

Previously the inter-part dependencies of shapes have been modeled by grammar induction [30], kernel density estimation [33], probabilistic graph models [29,6,24], and recursive autoencoders [27]. Those methods handle part relations and design synthesis separately. In contrast, our method encodes part relations through the model architecture, so that it simultaneously learns the inter-part dependencies and single part geometry variation. The model can also be used for inferring the generative distribution of child shapes conditioned on any parent shape.

Generative Adversarial Networks

Generative adversarial nets [7] model a game between a generative model (generator) and a discriminative model (discriminator). The generative model maps an arbitrary noise distribution to the data distribution (*i.e.*, the distribution of designs in our scenario), thus can generate new data; while the discriminative model tries to perform classification, *i.e.*, to distinguish between real and generated data (Fig. 2). The generator G and the discriminator D are usually built with deep neural networks. As D improves its classification ability, G also improves its ability to generate data that fools D. Thus the objective of GAN is

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

where $\boldsymbol{x}$ is sampled from the data distribution P_{data}, $\boldsymbol{z}$ is sampled from the noise distribution $P_{\boldsymbol{z}}$, and $G(\boldsymbol{z})$ is the generator distribution. A trained generator thus can map from the predefined noise distribution to the distribution of designs. The noise input $\boldsymbol{z}$ is considered as the latent representation in the dimensionality reduction scenario, since $\boldsymbol{z}$ captures the variability of the data.

One advantage GANs hold over Variational Auto-encoders (VAEs) [34] is that GANs tend to generate more realistic data [35]. But a disadvantage of vanilla GANs is that it cannot learn an interpretable latent representation. Built upon the vanilla GANs, the InfoGAN [36] aims at regularizing the latent representation of the data space by maximizing a lower bound of the mutual information between a set of *latent codes* $\boldsymbol{c}$ and the generated data. The generator is provided with both $\boldsymbol{z}$ and $\boldsymbol{c}$. Thus the generator distribution $P_G = G(\boldsymbol{z},\boldsymbol{c})$ is conditioned on $\boldsymbol{c}$. The mutual information lower bound L_I is

$$L_I(G,Q) = \mathbb{E}_{\boldsymbol{x}\sim P_G}[\mathbb{E}_{\boldsymbol{c}'\sim P(\boldsymbol{c}|\boldsymbol{x})}[\log Q(\boldsymbol{c}'|\boldsymbol{x})]] + H(\boldsymbol{c}) \tag{2}$$

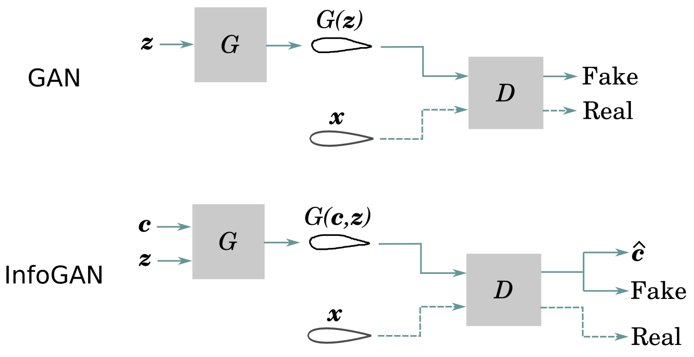

Figure 2: ARCHITECTURES OF GAN AND INFOGAN.

where $H(\boldsymbol{c})$ is the entropy of the latent codes, and Q is the auxiliary distribution for approximating $P(\boldsymbol{c}|\boldsymbol{x})$. We direct interested readers to [36] for the derivation of L_I. The InfoGAN objective combines L_I with the standard GAN objective:

$$\min_{G,Q} \max_D V_{InfoGAN}(D,G,Q) = V(D,G) - \lambda L_I(G,Q) \tag{3}$$

where λ is a weight parameter.

In practice, $H(\boldsymbol{c})$ is treated as constant if the distribution of $\boldsymbol{c}$ is fixed. The auxiliary distribution Q is simply approximated by sharing all the convolutional layers with D and adding an extra fully connected layer to D to predict the conditional distribution $Q(\boldsymbol{c}|\boldsymbol{x})$. Thus as shown in Fig. 2, InfoGAN's generator is conditioned on $\boldsymbol{c}$, and the discriminator tries to predict both the source of the data and the latent codes $\boldsymbol{c}$ [1].

In our design synthesis scenario, the latent codes $\boldsymbol{c}$ can represent any continuous or discrete factor that controls the geometry of the design, *e.g.*, the upper/lower surface protrusion of the airfoil.

METHOD

In this section we introduce our proposed deep neural network architecture and its training details.

Problem Formulation

This paper deals with a design synthesis problem, where we have a database of geometric designs with hierarchical structures. A parent shape is denoted as $\boldsymbol{x}_p \in \mathcal{X}_p$, where $\mathcal{X}_p$ is the design space for the parent shapes. Similarly, a child shape is denoted as $\boldsymbol{x}_c \in \mathcal{X}_c$, where $\mathcal{X}_c$ is the design space for the child shapes. We want to 1) learn the distributions of the parent shapes

[1]Here we use the discriminator D to denote both Q and D, since they share neural network parameters.

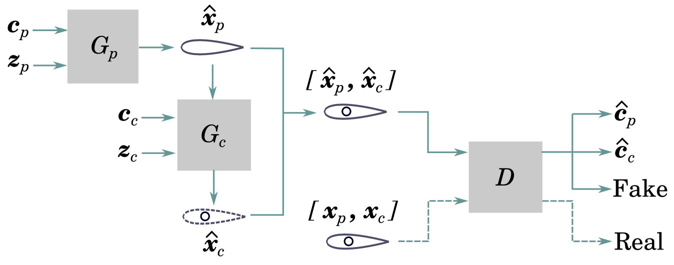

Figure 3: ARCHITECTURE OF THE HIERARCHICAL GAN.

and the child shapes (given any parent shape) separately from the database; 2) use two low-dimensional *latent spaces* $\mathcal{C}_p$ and $\mathcal{C}_c$ to represent the design spaces $\mathcal{X}_p$ and $\mathcal{X}_c$, respectively. The ability to learn separate distributions or representations of parent and child shapes is useful for decomposing a design space exploration problem (*e.g.*, design optimization).

Desired latent spaces satisfy the following requirements:

1. Any child latent space is conditioned on a parent shape.
2. Major variability of designs in the database is captured by those latent spaces.
3. Each dimension of a latent space corresponds to a semantic feature of designs.
4. The design changes consistently as it moves along any direction of the latent space.

The last two requirements benefit design space exploration as it is easier to search for desired designs in a latent space with these properties.

To meet the first requirement we construct a two-level generative model—the first level generates parent shapes from $\mathcal{C}_p$, and the second level generates child shapes from $\mathcal{C}_c$, conditioned on the output of the first level. We ensure the rest of the requirements by adapting InfoGAN's architecture and objective, *i.e.*, conditioning the generator on latent codes and maximizing a lower bound of the mutual information between the synthesized designs and their latent representations. This is known to 1) make latent codes target salient features of the designs, and 2) disentangle the latent representation of the data space [36]. We introduce the details of our model's architecture in the following section.

Model Architecture

To learn separate distributions/representations of parent and child shapes, we use a generative adversarial net with two generators—a *parent generator* G_p and a *child generator* G_c. We call this network the Hierarchical Generative Adversarial Networks (HGAN). Its architecture is shown in Fig. 3. Instead of using only noise as input to the generators, we add latent codes $\boldsymbol{c}_p$ and $\boldsymbol{c}_c$ as extra inputs. These latent codes define latent representations of designs. With this architecture, G_p learns the conditional parent shape distribution $P_p = P(\boldsymbol{x}_p|\boldsymbol{c}_p)$, and G_c learns the conditional child shape distribution $P_c = P(\boldsymbol{x}_c|\boldsymbol{x}_p, \boldsymbol{c}_c)$, where $\boldsymbol{z}_p$ and $\boldsymbol{z}_c$ are random noise input of G_p and G_c, respectively. The output distributions $P_{G_p} = G_p(\boldsymbol{c}_p, \boldsymbol{z}_p)$ and $P_{G_c} = G_c(\boldsymbol{c}_c, \boldsymbol{z}_c, \hat{\boldsymbol{x}}_p)$ represent synthesized parent and child shape distributions.

The *discriminator* D performs two tasks: 1) to classify whether the input complete design $[\boldsymbol{x}_p, \boldsymbol{x}_c]$ is from the database or generated by generators; and 2) to predict the latent codes of the parent and child shapes, *i.e.*, to estimate the conditional distributions $P(\boldsymbol{c}_p|\boldsymbol{x}_p)$ and $P(\boldsymbol{c}_c|\boldsymbol{x}_p, \boldsymbol{x}_c)$. A properly trained D should distinguish designs with unrealistic or mismatched parent/child shapes, and accurately predict the latent codes used to synthesize designs.

The objective of HGAN is expressed as

$$\begin{aligned} \min_{G_p,G_c,Q_p,Q_c} \max_{D} & V_{HGAN}(D, G_p, G_c, Q_p, Q_c) \\ &= V(D, G_p, G_c) - \lambda L_I(G_p, G_c, Q_p, Q_c) \end{aligned} \tag{4}$$

where the first term denotes the standard GAN objective:

$$\begin{aligned} V(D, G_p, G_c) = \mathbb{E}_{\boldsymbol{x}_p,\boldsymbol{x}_c \sim P_{data}}[\log D(\boldsymbol{x}_p, \boldsymbol{x}_c)] + \\ \mathbb{E}_{\boldsymbol{z}\sim P_{\boldsymbol{z}}}[\log(1 - D(G_p(\boldsymbol{z}_p, \boldsymbol{c}_p), G_c(\boldsymbol{z}_c, \boldsymbol{c}_c, G_p(\boldsymbol{z}_p, \boldsymbol{c}_p))))] \end{aligned} \tag{5}$$

and the second term is the lower bound of the mutual information between the latent codes and the synthesized designs:

$$\begin{aligned} L_I(G_p, G_c, Q_p, Q_c) = \mathbb{E}_{\boldsymbol{x}_p \sim P_{G_p}}[\mathbb{E}_{\boldsymbol{c}'_p \sim P(\boldsymbol{c}_p|\boldsymbol{x}_p)}[\log Q_p(\boldsymbol{c}'_p|\boldsymbol{x}_p)]] \\ + \mathbb{E}_{\boldsymbol{x}_p \sim P_{G_p}, \boldsymbol{x}_c \sim P_{G_c}}[\mathbb{E}_{\boldsymbol{c}'_c \sim P(\boldsymbol{c}_c|\boldsymbol{x}_p,\boldsymbol{x}_c)}[\log Q_c(\boldsymbol{c}'_c|\boldsymbol{x}_p, \boldsymbol{x}_c)]] \\ + H(\boldsymbol{c}_p) + H(\boldsymbol{c}_c) \end{aligned} \tag{6}$$

In practice, similar to the InfoGAN architecture [36], Q_p and Q_c share all the convolutional layers with D, and output parameters for $Q_p(\boldsymbol{c}_p|\boldsymbol{x}_p)$ and $Q_c(\boldsymbol{c}_c|\boldsymbol{x}_p, \boldsymbol{x}_c)$ through a fully connected layer.

In the field of image processing, GANs with similar structure have been proposed. To the best of our knowledge, the most similar model is the Style and Structure Generative Adversarial Network (S^2-GAN) [37]. It also uses a two-level architecture—the Structure-GAN learns the surface normal map of an image, and the Style-GAN is conditioned on this surface normal map and generates a natural image. The S^2-GAN has two discriminators. The first is trained to evaluate surface normal maps, and the second is to evaluate natural images. In comparison, our HGAN uses just one discriminator to evaluate the complete design. We compared HGAN with this S^2-GAN based architecture (we call it Naïve GAN in the rest of the paper because the S^2-GAN's cost function is built for image synthesis and is not what we need) in the next section.

EXPERIMENTS

We train HGAN on two datasets, and evaluate the trained generative models through quantitative measures.

Experimental Setup

Dataset. We denote the two datasets as A+H (airfoils with holes) and S+E (superformulas [38] with ellipses). Both datasets have over 90,000 samples. Examples of the two datasets are shown in Fig. 4.

The airfoil shapes are from the UIUC airfoil coordinates database[2], which provides the Cartesian coordinates for nearly 1,600 airfoils. Each airfoil is represented with 100 Cartesian coordinates, resulting in a 100×2 matrix. We use these airfoils as parent shapes. For each airfoil we add a hole (*e.g.*, a conduit hole) as its child shape. We add the holes by generating circles with random centers and sizes, and ruling out the ones not inside the airfoils. Each hole is also represented with 100 Cartesian coordinates. Hence each sample in this A+H dataset is a 200×2 matrix.

Though targeted for real-world applications, the airfoils may not be a perfect experimental dataset to visualize the latent space, because the ground truth intrinsic dimension[3] of the airfoil dataset is unknown. Thus we create another dataset using superformulas and ellipses, the intrinsic dimensions of which are controllable. The superformula is a generalization of the ellipse [38]. We generate superformulas as parent shapes using the following equations:

$$\begin{aligned} n_1 &= 10 s_1 \\ n_2 = n_3 &= 10(s_1 + s_2) \\ r(\theta) &= \left(|\cos\theta|^{n_2} + |\sin\theta|^{n_3}\right)^{-\frac{1}{n_1}} \\ (x, y) &= (r(\theta)\cos\theta, r(\theta)\sin\theta) \end{aligned} \tag{7}$$

where $s_1, s_2 \in [0, 1]$, and (x, y) is a Cartesian coordinate. For each superformula, we sample 100 evenly spaced θ from 0 to 2π, and get 100 grid-point Cartesian coordinates. Equations (7) show that we can control the deformation of the superformula shape with s_1 and s_2. Thus the ground truth intrinsic dimension of our superformula dataset is two. We use ellipses as child shapes. We generate ellipses with random semi-major axis and semi-minor axis lengths and fixed centers (at the centers of the ellipses). Hence the intrinsic dimension for the ellipses is also two. We rule out the ellipses that are not inside the superformulas. We set the intrinsic dimensions of both superformulas and ellipses to two because we want a two-dimensional latent space which is easy to visualize, evaluate, and verify. As with the A+H dataset, every superformula/ellipse is represented with 100 Cartesian coordinates, thus each sample in the S+E dataset is also a 200×2 matrix.

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

[3]The intrinsic dimension is the minimum number of variables required to represent the data.

Network Details. Figure 5 shows the details of the discriminator. The input is a 200×2 matrix, containing the parent shape and the child shape, each of which is expressed with 100 2D Cartesian coordinates. The input goes through four downsampling blocks, each performs a 1D convolution, a batch normalization, and a LeakyReLU activation. The length of each 1D convolution window is 5. The last downsampling block is followed by fully connected layers, predicting the mean and log standard deviation of latent codes, and the source of the input.

The generators' architectures, as shown in Fig. 6, basically mirror that of the discriminator. The parent generator gets two inputs: the parent latent codes $\boldsymbol{c}_p$ of size 2 and the input noise $\boldsymbol{z}_p$ of size 100. The inputs are then concatenated, fed into a fully connected layer, and reshaped to a 3D matrix. Then it goes through four upsampling blocks. The difference between the downsampling block and the upsampling block is that we use deconvolution (also called transposed convolution) [39] in the upsampling block, rather than convolution in the downsampling block. The final layer is a deconvolutional layer with hyperbolic tangent activation, and outputs the parent shape $\hat{\boldsymbol{x}}_p$, a $2 \times 100 \times 1$ matrix. It is then flattened and fed into the child generator. The child generator has the same architecture as the parent generator, except that there are three inputs—the flattened parent shape $\boldsymbol{x}_p$ of size 200, the child latent codes $\boldsymbol{c}_c$ of size 2 and the input noise $\boldsymbol{z}_c$ of size 100. The final output is the child shape $\hat{\boldsymbol{x}}_c$.

We compared HGAN with a naïve solution where we built two independent GANs—one for generating parent shapes, and another for generating child shapes conditioned on parent shapes. So there are two generators and two discriminators in this naïve solution. One discriminator for predicting the source of parent shapes and parent latent codes, and the other for predicting the source of child shapes and child latent codes. We used the same architecture for both generators as in HGAN.

Training Details. At training, we sample $\boldsymbol{c}_p$ and $\boldsymbol{c}_c$ from uniform distribution $\mathcal{U}(0,1)$, and $\boldsymbol{z}_p$ and $\boldsymbol{z}_c$ from normal distribution $\mathcal{N}(0, 0.25)$. The hyperparameter λ in Eqn. (4) was set to 1 in all experiments. The network was optimized using Adam [40] with the momentum terms $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rates of the discriminator and two generators were set to 0.00005 and 0.0002, respectively. The total number of training steps was 100,000. The batch size was 100. The training procedure is summarized in Algorithm 1.

To avoid the problem that the generative distribution and the true data distribution does not have overlapping support, we added normally distributed instance noise to discriminator in-

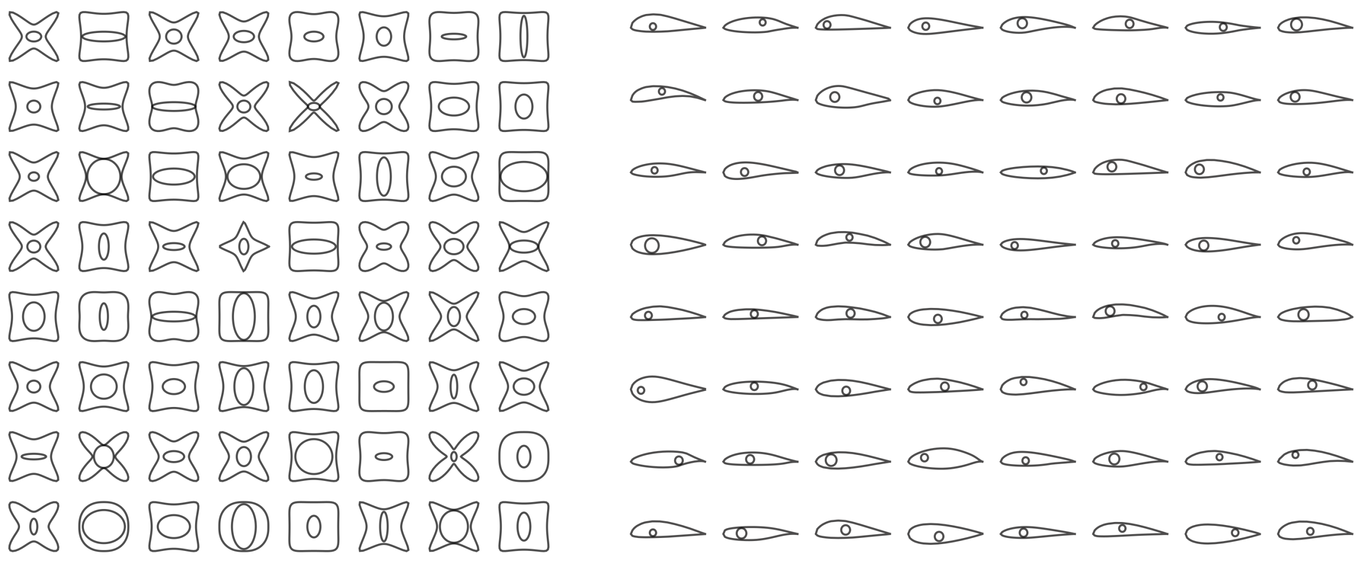
Figure 4: RANDOMLY SELECTED TRAINING DATA. **LEFT**: SUPERFORMULAS WITH ELLIPSES; **RIGHT**: AIRFOILS WITH HOLES.

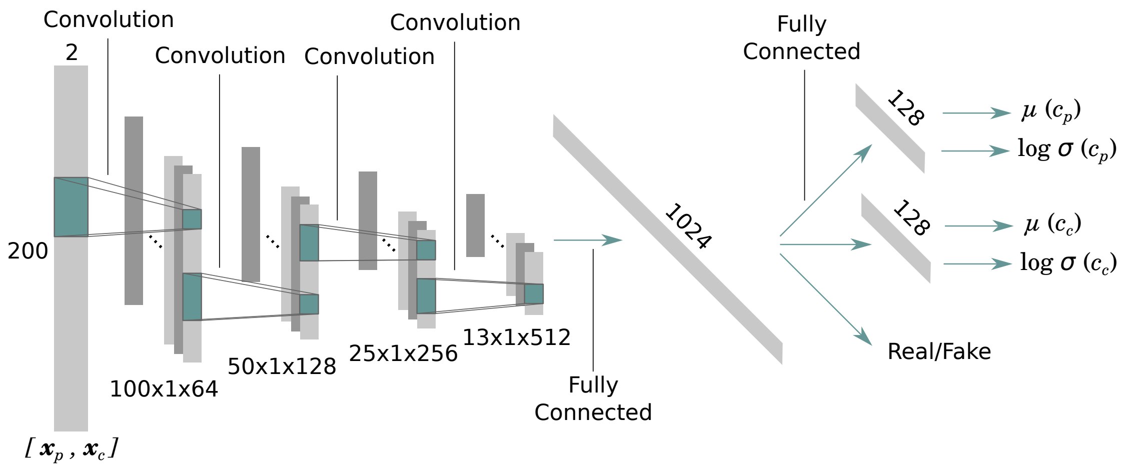

Figure 5: ARCHITECTURE OF THE DISCRIMINATOR.

puts [41]. Its standard deviation decreases as the number of training steps increases:

$$\sigma = e^{-t/10000}$$

We used Keras [42] with TensorFlow [43] backend to build the networks. We trained our networks on a Nvidia Titan X GPU. For each experiment, the training process took around 6.5 hours, and the testing took less than 10 seconds.

Results and Discussion

We evaluated the performance of our trained generative models using both visual inspection and quantitative measures. The quantitative results are shown in Table 1.

Visual Inspection. The captured latent spaces for both examples are shown in Fig. 7 and Fig. 8. To test the generalization ability of the child generator, we use parent shapes from the test set as inputs to the child generator. Thus each child la-

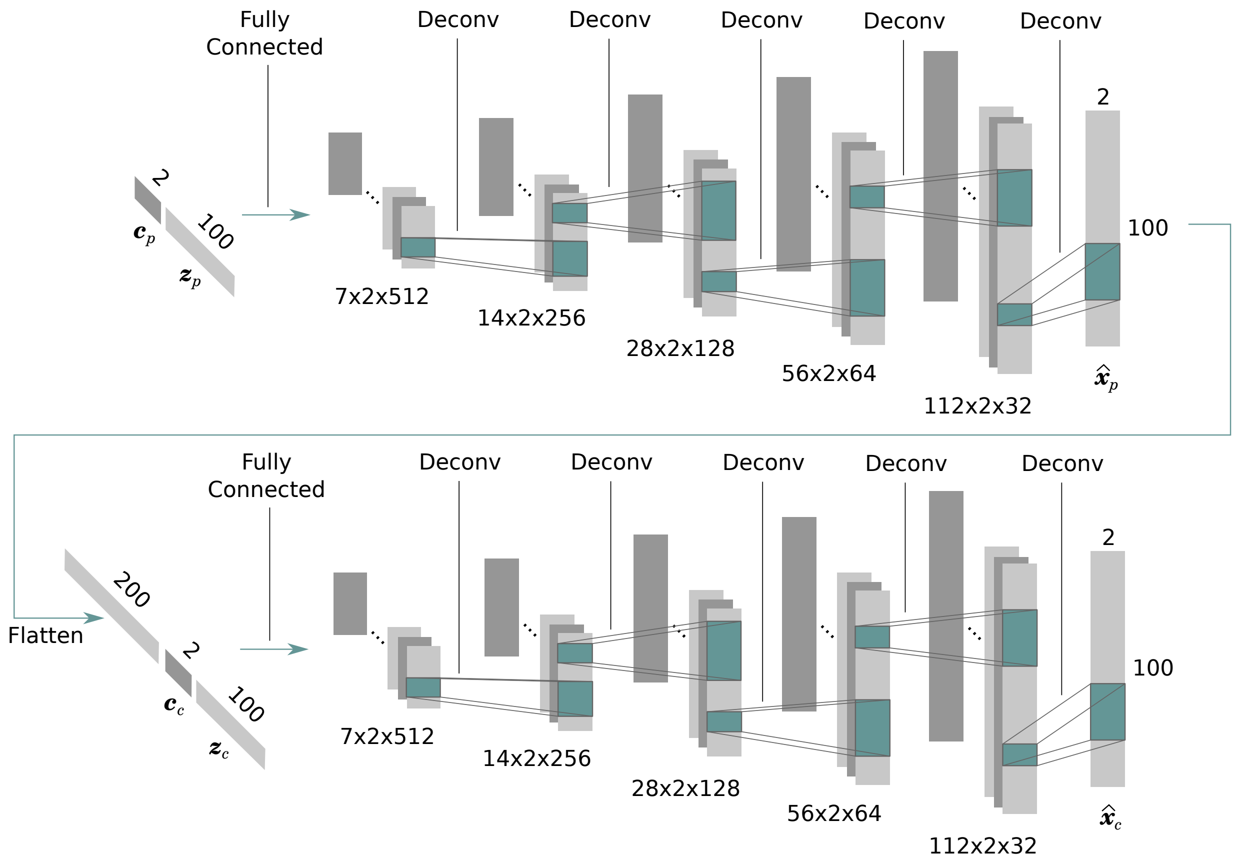

Figure 6: ARCHITECTURE OF THE GENERATORS. **TOP**: PARENT GENERATOR; **BOTTOM**: CHILD GENERATOR.

tent space shown in the plots is conditioned on a random test sample (plotted with dashed lines). The results show that the child latent spaces adjust themselves according to their parent shapes, so that the sizes/positions of child shapes fit in their parent shapes. This indicates that the child generator figures out the implicit constraints that holes/ellipses should be inside airfoils/superformulas. The shapes show a consistent change along any direction of the latent spaces. In contrast, latent spaces became entangled and inconsistent (Fig. 9) when we removed the mutual information lower bound L_I in Eqn. (4).

Likelihood under Generator Distribution. We measure how good our generator approximates the real data distribution by estimating the likelihood of test samples under our trained generator distribution. Specifically, since the exact likelihood of our generative models is not tractable, we compute the mean log-likelihood (MLL) by fitting a Gaussian kernel density estimator (KDE) to 2,000 synthesized designs, and computing the mean probability density of test data under the estimated distribution [44]. The bandwidth of the Gaussian kernel was obtained by cross-validated grid search.

Results show that the MLL of the naïve solution is much lower than HGAN. This means that HGAN, which contains only one discriminator, synthesizes designs that better resemble those in the database than the naïve solution which uses two discriminators. The maximization of L_I did not affect the MLL much.

Diversity of Generated Designs. A common problem in GANs' training is mode collapse, during which the generator only generates a few types of designs to fool the discriminator instead of properly learning the complete data distribution. Therefore it is important to measure the diversity of the synthesized designs. We use Structural Similarity Index (SSIM) [45] to measure this diversity. The SSIM is expressed as

$$\mathrm{SSIM}(\boldsymbol{x},\boldsymbol{y}) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{8}$$

where x and y are two designs, μ_x, σ_x, and σ_{xy} are the mean of x, the variance of x, and the covariance of x and y, respectively. The

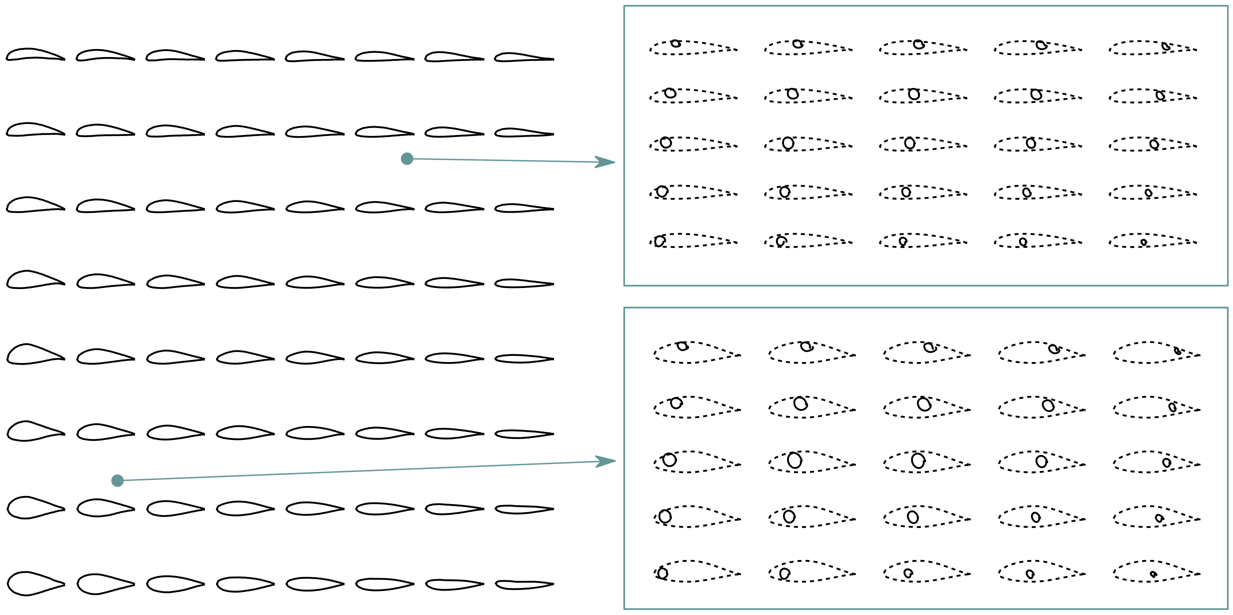

Figure 7: **LEFT**: AIRFOILS IN THE PARENT LATENT SPACE; **RIGHT**: HOLES IN THE LATENT SPACES CONDITIONED ON TWO RANDOM TEST SAMPLES OF AIRFOIL.

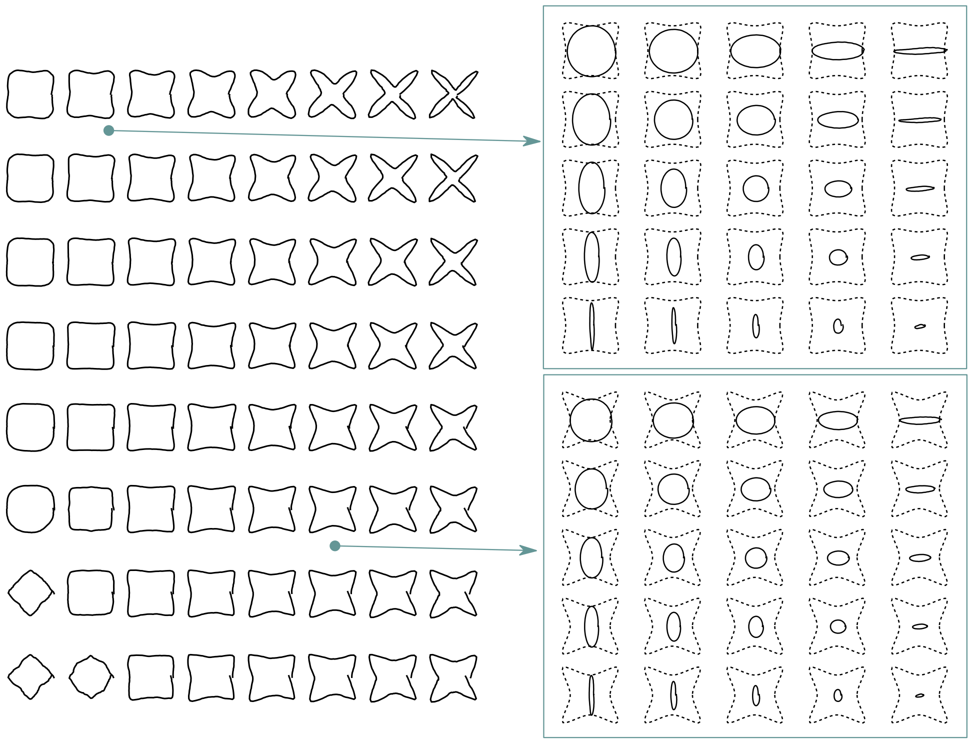

Figure 8: **LEFT**: SUPERFORMULA SHAPES IN THE PARENT LATENT SPACE; **RIGHT**: ELLIPSES IN THE LATENT SPACES CONDITIONED ON TWO RANDOM TEST SAMPLES OF SUPERFORMULA.

Table 1: QUANTITATIVE EVALUATION OF SYNTHESIZED DESIGNS.

Method	Design	MLL	R-SSIM	LSC (Parent)	LSC (Child)
A+H	Naïve	438.1 ± 272.9	$\mathbf{1.019 \pm 0.005}$	0.880 ± 0.011	0.995 ± 0.000
	HGAN w/o max L_I	$\mathbf{1256.4 \pm 1.6}$	0.993 ± 0.007	0.098 ± 0.008	0.071 ± 0.005
	HGAN	1240.1 ± 1.6	1.006 ± 0.005	$\mathbf{0.976 \pm 0.001}$	$\mathbf{0.998 \pm 0.000}$
S+E	Naïve	-1.7 ± 62.2	1.046 ± 0.008	$\mathbf{0.951 \pm 0.006}$	0.993 ± 0.000
	HGAN w/o max L_I	811.3 ± 27.9	$\mathbf{1.058 \pm 0.005}$	0.061 ± 0.007	0.065 ± 0.004
	HGAN	$\mathbf{948.0 \pm 26.9}$	1.006 ± 0.006	0.935 ± 0.006	$\mathbf{0.997 \pm 0.000}$

Algorithm 1 Train HGAN.

```
1:  ▷ $T$: number of training steps
2:  ▷ $m$: batch size
3:  ▷ $X$: training set
4:  procedure TRAIN($T$, $m$)
5:      for $t = 1 : T$ do
6:          Sample $\{\boldsymbol{x}^1, ..., \boldsymbol{x}^m\}$ from $X$, where $\boldsymbol{x}^i = [\boldsymbol{x}_p^i, \boldsymbol{x}_c^i], i = 1, ..., m$
7:          Sample $\{\boldsymbol{c}_p^1, ..., \boldsymbol{c}_p^m\}$ from a uniform distribution
8:          Sample $\{\boldsymbol{z}_p^1, ..., \boldsymbol{z}_p^m\}$ from a normal distribution
9:          Sample $\{\boldsymbol{c}_c^1, ..., \boldsymbol{c}_c^m\}$ from a uniform distribution
10:         Sample $\{\boldsymbol{z}_c^1, ..., \boldsymbol{z}_c^m\}$ from a normal distribution
11:         ▷ Based on Eqn. (4-6):
12:         Update $D$, fixing $G_p$ and $G_c$
13:         Update $G_p$, fixing $D$ and $G_c$
14:         Update $G_c$, fixing $D$ and $G_p$
15:     end for
16: end procedure
```

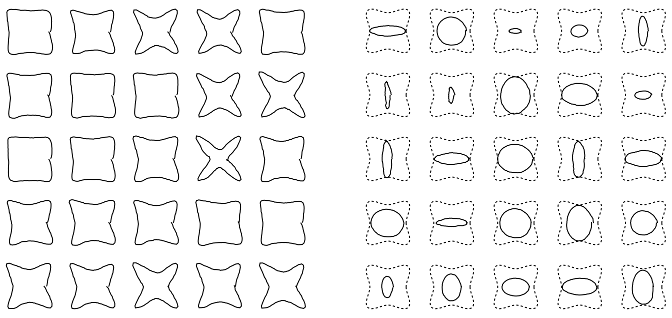
Figure 9: LATENT SPACES LEARNED WITHOUT MAXIMIZING THE MUTUAL INFORMATION LOWER BOUND.

constants $C_1 = (k_1L)^2$, and $C_2 = (k_2L)^2$ are used to prevent the error of division by a small integer close to zero from happening. By default, $k_1 = 0.01$, $k_2 = 0.03$, and L is the range of coordinates. The SSIM score ranges from 0.0 (no similarity) to 1.0 (high similarity). Different from the Euclidean distance which focuses on local details, SSIM compares the overall structure of designs, and is more invariant under translation and rotation.

Here we use relative SSIM (R-SSIM) to measure how well the generated designs cover the diversity of the training samples:

$$\text{R-SSIM} = \frac{\mathbb{E}_{\boldsymbol{x}_D, \boldsymbol{x}'_D \sim P_{data}}[\text{SSIM}(\boldsymbol{x}_D, \boldsymbol{x}'_D)]}{\mathbb{E}_{\boldsymbol{x}_G, \boldsymbol{x}'_G \sim P_G}[\text{SSIM}(\boldsymbol{x}_G, \boldsymbol{x}'_G)]} \tag{9}$$

Results show that the R-SSIM scores under all methods are around 1, which means the diversity of generated designs is at the same level as the database and mode collapse did not happen.

Latent Space Consistency. A desirable latent space has two properties: 1) *disentanglement*: each latent variable is related to only one factor; and 2) *consistency*: shapes change consistently along any direction in the latent space. Note that this consistency is evaluated along one direction at a time, since scales along different directions may vary. To the best of our knowledge, existing quantitative measurements for the first property—latent space disentanglement—are supervised, *i.e.*, the ground-truth independent factors causing shape deformation have to be provided [46, 47, 48]. The second property is important for design space exploration. When searching for designs along a direction in the latent space, the algorithm or human usually prefer the shape to change consistently, such that the objective function over the latent space is less complicated and has less local optima.

We propose Latent Space Consistency (LSC) as a quantitative measure of how consistently shapes change along any direction in the latent space. Since change from one shape to another can be measured by their dissimilarity, distances between samples along a certain direction in the latent space should be consistent with the dissimilarity between those samples. We use Pearson correlation coefficient to measure this consistency. Details for computing LSC is shown in Algorithm 2. The choice of the dissimilarity function d is not central to the overall method. In our experiments, we simply use the Euclidean distance to measure the dissimilarity of designs. The LSC obtained without maximizing the mutual information lower bound L_I in Eqn. (4) is also shown in Table 1.

Algorithm 2 Evaluate Latent Space Consistency.

```
1: procedure LATENTCONSISTENCY($G, m, n, d$)
2:     ▷ $G$: the mapping from a latent space to a design space
3:     ▷ $m$: the number of lines to be evaluated
4:     ▷ $n$: the number of points sampled on each line
5:     ▷ $d$: a dissimilarity function
6:     ▷ $\mathcal{C}$: the latent space
7:     ▷ $\mathcal{X}$: the design space
8:     $sum = 0$
9:     for $i = 1 : m$ do
10:        Sample a line $\mathcal{L}$ parallel to any basis of $\mathcal{C}$
11:        Sample $n$ equally-spaced and ordered points $\{\boldsymbol{c}^1, \boldsymbol{c}^2, ..., \boldsymbol{c}^n\}$ along $\mathcal{L}$
12:        $\{\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^n\} = \{G(\boldsymbol{c}^1), G(\boldsymbol{c}^2), ..., G(\boldsymbol{c}^n)\}$
13:        $D_\mathcal{C} = \{\|\boldsymbol{c}^2 - \boldsymbol{c}^1\|, \|\boldsymbol{c}^3 - \boldsymbol{c}^2\|, ..., \|\boldsymbol{c}^n - \boldsymbol{c}^{n-1}\|\}$
14:        $D_\mathcal{X} = \{d(\boldsymbol{x}^2, \boldsymbol{x}^1), d(\boldsymbol{x}^3, \boldsymbol{x}^2), ..., d(\boldsymbol{x}^n, \boldsymbol{x}^{n-1})\}$
15:        Compute Pearson correlation coefficient:
               $\rho = \frac{\mathrm{cov}(D_\mathcal{C}, D_\mathcal{X})}{\sigma(D_\mathcal{C})\sigma(D_\mathcal{X})}$
16:        $sum = sum + \rho$
17:    end for
18:    Return $sum/m$
19: end procedure
```

Results show that the LSC is significantly lower when we do not maximize L_I. This is also verified by comparing Fig. 8 and Fig. 9.

CONCLUSION

We introduced a GAN-based generative model for synthesizing designs with hierarchical structures. It decomposes the synthesis into two stages: first synthesizes the parent shape from a learned latent representation; and then synthesizes the child shape from another learned latent representation conditioned on the parent shape. This model is build for problems where design space exploration over the latent space has to be staged since the optimal solution of one component depends on the geometry of another. We will use this model to solve design optimization problems in our future work.

An advantage of neural networks based generative models (*e.g.*, GANs and VAEs), compared to other dimensionality reduction models (*e.g.*, PCA and GPLVM), is that one can define or regularize latent distributions. Our model adapts InfoGAN's mutual information objective to derive a consistent latent space, where the change of shapes is consistent along any direction of the latent space. This property is desirable in design space exploration, as the objective function over the latent space is less complicated and has less local optima.

One problem of our method is that some synthesized shapes have low smoothness. This can be solved by either adding a post-processing step to fit Bezier or B-spline curves to the synthesized grid-point coordinates; or a pre-processing step to convert the grid-point coordinates in the training data to spline representations (*i.e.*, control point coordinates of spline curves), so that the generators directly synthesize spline representations.

Further testing on synthesizing designs with higher dimensions or more complex inter-part dependencies is also required. For example, the parent-child relationship can possibly be one-to-many (increased complexity in breadth), and the hierarchy may have three or more levels (increased complexity in depth). Techniques may need to be developed to reduce the high computational cost and avoid possible convergence issues.

ACKNOWLEDGMENT

This work was supported by The Defense Advanced Research Projects Agency (DARPA-16-63-YFA-FP-059) via the Young Faculty Award (YFA) Program. The views, opinions, and/or findings contained in this article are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

REFERENCES

[1] Chen, W., Fuge, M., and Chazan, N., 2017. "Design manifolds capture the intrinsic complexity and dimension of design spaces". *Journal of Mechanical Design,* ***139***(5), pp. 051102–051102–10.

[2] Chen, W., and Fuge, M., 2017. "Beyond the known: Detecting novel feasible domains over an unbounded design space". *Journal of Mechanical Design,* ***139***(11), pp. 111405–111405–10.

[3] Königseder, C., and Shea, K., 2016. "Visualizing relations between grammar rules, objectives, and search space exploration in grammar-based computational design synthesis". *Journal of Mechanical Design,* ***138***(10), p. 101101.

[4] Königseder, C., Stanković, T., and Shea, K., 2016. "Improving design grammar development and application through network-based analysis of transition graphs". *Design Science,* ***2***.

[5] Chen, W., and Fuge, M., 2017. "Active expansion sampling for learning feasible domains in an unbounded input space". *Structural and Multidisciplinary Optimization*, pp. 1–21.

[6] Kalogerakis, E., Chaudhuri, S., Koller, D., and Koltun, V., 2012. "A probabilistic model for component-based shape synthesis". *ACM Transactions on Graphics (TOG),* ***31***(4), p. 55.

[7] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2014. "Generative adversarial nets". In Advances in neural information processing systems, pp. 2672–2680.

[8] Samareh, J. A., 2001. "Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization". *AIAA journal,* ***39***(5), pp. 877–884.

[9] Bellman, R., 1957. *Dynamic programming*. Princeton University Press, Princeton, NY.

[10] Diez, M., Campana, E. F., and Stern, F., 2015. "Design-space dimensionality reduction in shape optimization by karhunen–loève expansion". *Computer Methods in Applied Mechanics and Engineering,* ***283***, pp. 1525–1544.

[11] Chen, X., Diez, M., Kandasamy, M., Zhang, Z., Campana, E. F., and Stern, F., 2015. "High-fidelity global optimization of shape design by dimensionality reduction, metamodels and deterministic particle swarm". *Engineering Optimization,* ***47***(4), pp. 473–494.

[12] DAgostino, D., Serani, A., Campana, E. F., and Diez, M., 2017. "Nonlinear methods for design-space dimensionality reduction in shape optimization". In International Workshop on Machine Learning, Optimization, and Big Data, Springer, pp. 121–132.

[13] Tezzele, M., Salmoiraghi, F., Mola, A., and Rozza, G., 2017. "Dimension reduction in heterogeneous parametric spaces with application to naval engineering shape design problems". *arXiv preprint arXiv:1709.03298*.

[14] Raghavan, B., Breitkopf, P., Tourbier, Y., and Villon, P., 2013. "Towards a space reduction approach for efficient structural shape optimization". *Structural and Multidisciplinary Optimization,* ***48***(5), pp. 987–1000.

[15] Raghavan, B., Xia, L., Breitkopf, P., Rassineux, A., and Villon, P., 2013. "Towards simultaneous reduction of both input and output spaces for interactive simulation-based structural design". *Computer Methods in Applied Mechanics and Engineering,* ***265***, pp. 174–185.

[16] Raghavan, B., Le Quilliec, G., Breitkopf, P., Rassineux, A., Roelandt, J.-M., and Villon, P., 2014. "Numerical assessment of springback for the deep drawing process by level set interpolation using shape manifolds". *International Journal of Material Forming,* ***7***(4), pp. 487–501.

[17] Le Quilliec, G., Raghavan, B., and Breitkopf, P., 2015. "A manifold learning-based reduced order model for springback shape characterization and optimization in sheet metal forming". *Computer Methods in Applied Mechanics and Engineering,* ***285***, pp. 621–638.

[18] Viswanath, A., J. Forrester, A., and Keane, A., 2011. "Dimension reduction for aerodynamic design optimization". *AIAA journal,* ***49***(6), pp. 1256–1266.

[19] Qiu, H., Xu, Y., Gao, L., Li, X., and Chi, L., 2016. "Multistage design space reduction and metamodeling optimization method based on self-organizing maps and fuzzy clustering". *Expert Systems with Applications,* ***46***, pp. 180–195.

[20] Burnap, A., Liu, Y., Pan, Y., Lee, H., Gonzalez, R., and Papalambros, P. Y., 2016. "Estimating and exploring the product form design space using deep generative models". In ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V02AT03A013–V02AT03A013.

[21] D'Agostino, D., Serani, A., Campana, E. F., and Diez, M., 2018. "Deep autoencoder for off-line design-space dimensionality reduction in shape optimization". In 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, p. 1648.

[22] Gmeiner, T., and Shea, K., 2013. "A spatial grammar for the computational design synthesis of vise jaws". In ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V03AT03A006–V03AT03A006.

[23] Talton, J. O., Gibson, D., Yang, L., Hanrahan, P., and Koltun, V., 2009. "Exploratory modeling with collaborative design spaces". *ACM Transactions on Graphics-TOG,* ***28***(5), p. 167.

[24] Huang, H., Kalogerakis, E., and Marlin, B., 2015. "Analysis and synthesis of 3d shape families via deep-learned generative models of surfaces". In Computer Graphics Forum, Vol. 34, Wiley Online Library, pp. 25–38.

[25] Nash, C., and Williams, C. K., 2017. "The shape variational autoencoder: A deep generative model of part-segmented 3d objects". In Computer Graphics Forum, Vol. 36, Wiley Online Library, pp. 1–12.

[26] Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J., 2016. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling". In Advances in Neural Information Processing Systems, pp. 82–90.

[27] Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., and Guibas, L., 2017. "Grass: Generative recursive autoencoders for shape structures". *ACM Transactions on Graph-*

ics (TOG), ***36***(4), p. 52.

[28] Sinha, A., Unmesh, A., Huang, Q., and Ramani, K., 2017. "Surfnet: Generating 3d shape surfaces using deep residual networks". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6040–6049.

[29] Chaudhuri, S., Kalogerakis, E., Guibas, L., and Koltun, V., 2011. "Probabilistic reasoning for assembly-based 3d modeling". In ACM Transactions on Graphics (TOG), Vol. 30, ACM, p. 35.

[30] Talton, J., Yang, L., Kumar, R., Lim, M., Goodman, N., and Měch, R., 2012. "Learning design patterns with bayesian grammar induction". In Proceedings of the 25th annual ACM symposium on User interface software and technology, ACM, pp. 63–74.

[31] Xu, K., Zhang, H., Cohen-Or, D., and Chen, B., 2012. "Fit and diverse: set evolution for inspiring 3d shape galleries". *ACM Transactions on Graphics (TOG),* ***31***(4), p. 57.

[32] Zheng, Y., Cohen-Or, D., and Mitra, N. J., 2013. "Smart variations: Functional substructures for part compatibility". *Computer Graphics Forum (Eurographics),* ***32***(2pt2), pp. 195–204.

[33] Fish, N., Averkiou, M., Van Kaick, O., Sorkine-Hornung, O., Cohen-Or, D., and Mitra, N. J., 2014. "Meta-representation of shape families". *ACM Transactions on Graphics (TOG),* ***33***(4), p. 34.

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

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

[36] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P., 2016. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets". In Advances in Neural Information Processing Systems, pp. 2172–2180.

[37] Wang, X., and Gupta, A., 2016. "Generative image modeling using style and structure adversarial networks". In European Conference on Computer Vision, Springer, pp. 318–335.

[38] Gielis, J., 2003. "A generic geometric transformation that unifies a wide range of natural and abstract shapes". *American journal of botany,* ***90***(3), pp. 333–338.

[39] Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R., 2010. "Deconvolutional networks". In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, pp. 2528–2535.

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

[41] Sønderby, C. K., Caballero, J., Theis, L., Shi, W., and Huszár, F., 2016. "Amortised map inference for image super-resolution". *arXiv preprint arXiv:1610.04490*.

[42] Chollet, F., et al., 2015. Keras. `https://github.com/keras-team/keras`.

[43] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[44] Breuleux, O., Bengio, Y., and Vincent, P., 2011. "Quickly generating representative samples from an rbm-derived process". *Neural computation,* ***23***(8), pp. 2058–2073.

[45] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P., 2004. "Image quality assessment: from error visibility to structural similarity". *IEEE transactions on image processing,* ***13***(4), pp. 600–612.

[46] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A., 2016. "beta-vae: Learning basic visual concepts with a constrained variational framework".

[47] Kim, H., and Mnih, A., 2018. "Disentangling by factorising". *arXiv preprint arXiv:1802.05983*.

[48] Chen, T. Q., Li, X., Grosse, R., and Duvenaud, D., 2018. "Isolating sources of disentanglement in variational autoencoders". *arXiv preprint arXiv:1802.04942*.