Author version

Published as:

Ruh, M. L., Liu, X., Yu, R., & Hwang, J. T. (2023). Airfoil shape parameterization using reconstruction-error-minimizing generative adversarial networks. In AIAA AVIATION 2023 Forum (Paper No. AIAA 2023-3722). https://doi.org/10.2514/6.2023-3722

BibTeX for this reference:

https://lsdolab.github.io/lsdo-publications/publication/ruh2023airfoil/

```
@inproceedings{ruh2023airfoil,
  author = {Marius L. Ruh and Xiangbei Liu and Rose Yu and John T. Hwang},
  title = {Airfoil shape parameterization using reconstruction-error-minimizing
    generative adversarial networks},
  booktitle = {AIAA AVIATION 2023 Forum},
  year = {2023},
  doi = {10.2514/6.2023-3722},
  url = {https://doi.org/10.2514/6.2023-3722}
}
```

Airfoil shape parameterization using reconstruction-error-minimizing generative adversarial networks

Marius L. Ruh*
University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093

Xiangbei Liu†
Dartmouth College, Hanover, NH 03755

Rose Yu‡, John T. Hwang§
University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093

Many physics-based aerodynamic models discretize lifting surfaces spanwise and evaluate an embedded airfoil model at each section. When used in a design optimization context, it is critical that the embedded airfoil model is fast, is robust, and provides derivatives in the case of gradient-based optimization. Well-trained data-driven airfoil models can fit this need and can provide reasonable accuracy if a sufficiently large data set is available. An important component of such an airfoil model is an effective parameterization of the airfoil shape. In this work, we develop a method based on generative adversarial networks (GANs) for parameterizing airfoil shapes using mappings called generators. This method modifies the standard GAN architecture by adding a term to the training objective that captures the generator's ability to reconstruct a given set of known airfoils. In addition, we propose three measures to quantitatively assess the effectiveness of a particular generator (i.e., a particular airfoil shape parameterization) and find that our method improves upon existing methods according to all three measures. In particular, our method is over 20% better in capturing the space of real airfoils (the first measure), which is quantified as the agreement in distributions of airfoil properties between randomly generated airfoils and a set of known airfoils. Our method is 41% better in terms of the reconstruction error for the known airfoils (the second measure). We also show preliminary results for a supervised learning model to predict aerodynamic coefficients, based on XFOIL data for randomly generated airfoils.

I. Nomenclature

M	=	Mach number
Re	=	Reynolds number
AoA	=	angle of attack
C_l	=	lift coefficient
C_d	=	drag coefficient
SVD	=	singular value decomposition
GAN	=	generative adversarial network
$\mathbf{c}$	=	latent variables
$\mathbf{z}$	=	noise variables
$\mathbf{x}$	=	noise and latent input variables
$\widetilde{\mathbf{x}}$	=	noise and latent design variables
$\widetilde{\mathbf{c}}$	=	randomly generated airfoil
$\overline{\mathbf{c}}$	=	reconstructed airfoil
$\mathbf{c}$	=	airfoil in database

*Ph.D. Student, Department of Mechanical and Aerospace Engineering, AIAA student member
†Ph.D. Student, Department of Engineering
‡Assistant Professor, Department of Computer Science and Engineering
§Assistant Professor, Department of Mechanical and Aerospace Engineering, AIAA senior member

II. Introduction

URBAN air mobility (UAM) is an exciting vision that aims to provide on-demand transportation through the use of new, electric rotary-wing aircraft with vertical take-off and landing capabilities (eVTOL). In the conceptual design and sizing of such aircraft, low-fidelity aerodynamic models such as the vortex lattice method (VLM) or blade element momentum (BEM) theory are typically used. These models require airfoil polar information, and an accurate airfoil model can improve prediction accuracy. However, accurate airfoil models are generally based on physics-based solvers such as panel codes (e.g., XFOIL [1]) or Reynolds-averaged Navier-Stokes (RANS) simulations. Embedding these solvers into VLM or BEM methods directly introduces a computational bottleneck as these solvers can be expensive to evaluate and may not be suited for rapid design optimization due to a lack of robustness.

A potential solution is to train a surrogate model with data from physics-based solvers. If trained well, a surrogate model can predict aerodynamic quantities of interest such as lift and drag coefficients or pressure distributions almost instantaneously with reasonable accuracy and without failure. In recent years, advances in data science as well as an abundance of available aerodynamic data have enabled the application of surrogate modeling techniques such as machine learning (ML) to airfoil simulation.

The first step in applying ML to airfoil modeling is to effectively parameterize the airfoil shape, preferably in a way that reduces the dimensionality of the high-dimensional design space. Li et al. [2] used singular value decomposition (SVD) on a subset of the UIUC [3] airfoil data base. They were able to regenerate the data set with only the first 20 modes (i.e., the left singular vectors). While the use of SVD allows for accurate reconstruction of existing airfoils from a low-dimensional subspace, it may not be able to generate new airfoils by combining different modes. For airfoil shape optimization, it is advantageous to be able to generate new, realistic airfoils that are not contained within existing data bases. An unsupervised machine learning technique that is able to generate realistic airfoils is generative adversarial networks (GAN), which we leverage in this paper. Variations of the GAN approach have been applied [4–10] to generate airfoils from a low-dimensional input space (e.g., noise and latent variables). Another unsupervised learning technique that has been applied to the airfoil parameterization problem is variational autoencoders [11–14].

The second step in airfoil modeling using ML is the prediction of aerodynamic quantities. This requires a large data set to ensure reasonable accuracy. Li et al. [2] used data from a large set of RANS simulations to train a surrogate model by combining gradient-enhanced kriging, partial least squares, and a mixture of experts. In a similar study, Bouhlel et al. [15] used gradient-enhanced neural networks and demonstrated that this approach outperforms more traditional surrogate modeling techniques such as kriging. One particular focus area of applying ML to airfoil simulation is that of turbulence modeling [16–23], which is seldom considered at the conceptual design stage. Other notable studies [24–26] use deep convolutional neural networks for aerodynamic prediction.

While many studies focus on applying machine learning to enable fast aerodynamic prediction, it is equally important to consider the parameterization of the airfoil shape. A well-trained machine learning model that can effectively represent any existing airfoil in a low-dimensional subspace and also generate new realistic airfoils can be beneficial in the optimization of airfoil shapes to find the best design. In this work, we propose a modified, GAN-based training technique that aims to find a nonlinear, low-dimensional manifold for the infinite-dimensional space of all real airfoils. A generator neural network maps from a low-dimensional input space to this manifold. Our contribution is the modified training procedure. We also propose the following three performance measures for a generator:

1) *Capturing the space of realistic airfoils*: measures the "closeness" of distributions of key airfoil properties between the training and generated data
2) *Reconstruction of real airfoils*: measures how well a generator can approximate the shape of existing airfoils
3) *Generation of only realistic airfoils*: measures the number of realistic airfoils in a method-independent way

These measures are used to assess and compare our method to existing methods for constructing airfoil shape parameterizations. Based on the findings of this paper, we anticipate that an airfoil generator trained with the proposed modified training method can improve the exploration of the design space of realistic airfoils and aid in the optimization of airfoil shapes.

In the remainder of the paper, we give a brief overview of the theory of generative adversarial networks (GANs) in Sec. III and describe the modified training approach and the three criteria for assessing the performance of a generator in Sec. IV. Next, we show results based on these criteria in Sec.V, along with preliminary results of applying supervised learning to aerodynamic prediction based on XFOIL training data.

III. Background

A. Generative adversarial networks

Generative adversarial networks (GAN) are an unsupervised, generative learning technique developed by Goodfellow [27] to generate data that is close to indistinguishable from the real data. Heuristically, GAN models can be viewed as a min-max game between two players, a generator G and discriminator D, which are both neural networks. The input to the generator are noise variables $\mathbf{z}$ that are typically drawn from a normal distribution. During training, the goal of the generator is to produce data that is as close to the real data as possible such that the discriminator classifies $G(\mathbf{z})$ as real data. On the other hand, the discriminator D is a classifier that classifies the output of the generator as real (1) or fake (0). The objective of the discriminator is to always classify the output of the generator as fake. If either player is successful at their task, the other player is penalized via a loss function, whose derivatives are back-propagated to train the neural network. We illustrate the training process of GAN in Fig. 1, in terms of the airfoil parameterization problem. The training process can be formulated mathematically as

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

One of the challenges of the standard GAN approach is that of mode collapse [28–30], which means that the generator is only able to learn few modes (i.e., characteristics) of the training data distribution. This can occur due to local minima when training the discriminator, which causes the generator to produce the same data repeatedly. Using different loss functions can alleviate this problem, leading to variations of GAN such as CGAN [31], WGAN [32], of InfoGAN [33], the last of which we describe next.

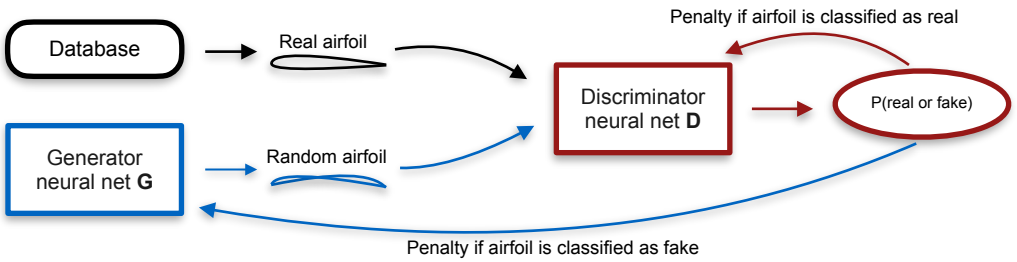


Fig. 1 Illustration of the GAN training process

B. Additional mutual information through InfoGAN

The standard GAN formulation parameterizes an airfoil in terms of noise variables only. A challenge related to mode collapse is that information regarding the airfoil geometry is densely packed, meaning that a particular noise variable is unlikely to correspond to a certain geometric property like thickness. In some cases, the noise variables are not able to capture the design space at all. From an application standpoint, we would like the inputs to the generator to exert unique control over the airfoil shape.

Chen et al. [33] introduced an information-theoretic extension to GAN called information maximizing generative adversarial networks (InfoGAN). In addition to noise variables $\mathbf{z}$, the generator also takes latent variables $\mathbf{c}$ as inputs, such that the form of the generator becomes $G(\mathbf{z}, \mathbf{c})$. The latent variables $\mathbf{c}$ encode hidden features of the data, which in our case refers to different properties of the airfoil shape. This is achieved via a regularization term that maximizes the mutual information $I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$ between latent variables $\mathbf{c}$ and the generator distribution $G(\mathbf{z}, \mathbf{c})$. In practice, this formulation adds a variational lower bound of mutual information, $L_I(G, Q)$, to the original GAN objective function with some hyper-parameter λ. That is

$$\min_{G,Q} \max_D V_{\text{InfoGAN}}(D, G, Q) = V(D, G) - \lambda L_I(G, Q). \tag{2}$$

For parameterization of airfoils, Du et al. [6] proposed the addition of two regularization terms to keep adjacent B-spline control points close in the y-direction and avoid intersection between upper and lower curve control points. The resulting InfoGAN formulation then becomes

$$\min_{G,Q} \max_D V_{\text{InfoGAN}}(D, G, Q) = V(D, G) - \lambda L_I(G, Q) + \sum_{i=1}^{2} \lambda_i R_i(G). \tag{3}$$

In this paper, we parameterize the airfoil in terms of the camber and thickness line, which only requires the regularization term that keeps control points close in the y-direction.

IV. Methodology

A. Data set

For our dataset, we use a subset of the UIUC airfoil database [3], considering cambered airfoils only. This is because the GAN model has difficulty learning symmetric airfoils. We apply a filter based on the standard deviation of geometric parameters such as max camber, max thickness, leading edge angle, and others, which we compute for the entire data set. This ensures that outliers are removed from the final training set, which contains 1320 airfoils. To generate aerodynamic training data, we use a well-trained generator to produce 500 airfoils. We use XFOIL [1] to perform aerodynamic analysis on these airfoils for different combinations of angle of attack (-8 to 17°), Reynolds number (100,000 to 8,000,000), and Mach number (0 to 0.6). In total, roughly 330,000 XFOIL evaluations are used as aerodynamic training data. For this work, we will only show results for the lift and drag coefficients.

B. Parameterization of airfoil coordinates

We represent the airfoils in terms of their camber and thickness distributions because several geometric parameters like maximum camber or maximum thickness are computed from these distributions. To reduce the dimensionality of the airfoil design space, we use basis-splines (B-splines) to represent the camber and thickness lines with 18 control points each. B-splines are piece-wise polynomials that can be expressed as

$$B(u) = \sum_{i=0}^{n} N_{i,k}(u) p_i, \tag{4}$$

where k is the order of the B-spline curve, $N_{i,k}$ are the basis functions that take in a knot vector u with entries ranging between 0 and 1, and p_i is the ith control point. The B-spline basis functions $N_{i,k}$ are defined as

$$N_{i,1} = \begin{cases} 1 & u_i \leq u \leq u_{i+1} \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

$$N_{i,k} = \frac{u - u_i}{u_{i+k-1} - u_i} N_{i,k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1,k-1}(u), \tag{6}$$

where the length of the knot vector increases by $n + 1 + k$, $u_0 = 0$, and $u_{k+n} = 1$. To ensure consistency between airfoils, we keep the x-coordinates of all control points equal and only vary the y-coordinates. We set the control points at the leading and trailing edge zero and chose a half-cosine distribution for the remaining control points. This ensures a high resolution of points near the leading edge, which is favorable for aerodynamic predictive tools like XFOIL.

C. Modified training of generator

The application of (Info)GAN to the airfoil parameterization problem ensures that for a well-trained generator any randomly generated airfoil is realistic, based on the discriminator neural network. However, we also want to ensure that the generator is able to reconstruct any airfoil contained in the training set to within a certain build tolerance. This is done by optimizing the noise and latent variables such that the L2 norm between the reconstructed airfoil coordinates and original coordinates is minimized. To achieve this goal, we propose a modified training procedure for the generator network, shown in Fig. 2.

The blue and red branch show the training process that has been applied previously to the airfoil shape parameterization problem by Du et al. [6]. In addition, the green branch indicates a new training objective that is defined by adding another loss term to the generator. We call this the reconstruction loss L_r, which expresses how good the generator is at reconstructing an airfoil contained in our database. In terms of the training process, we first follow the regular GAN procedure for N epochs (i.e., training iterations). We then use a copy of the generator (with fixed weights and biases) to reconstruct a batch of UIUC airfoils by optimizing the noise and latent variables for M iterations and add the resulting reconstruction error to the generator loss term. Here, N and M are not to be confused with previous notation. Optimization of the noise and latent variables is achieved by retaining the graph of the generator at the current

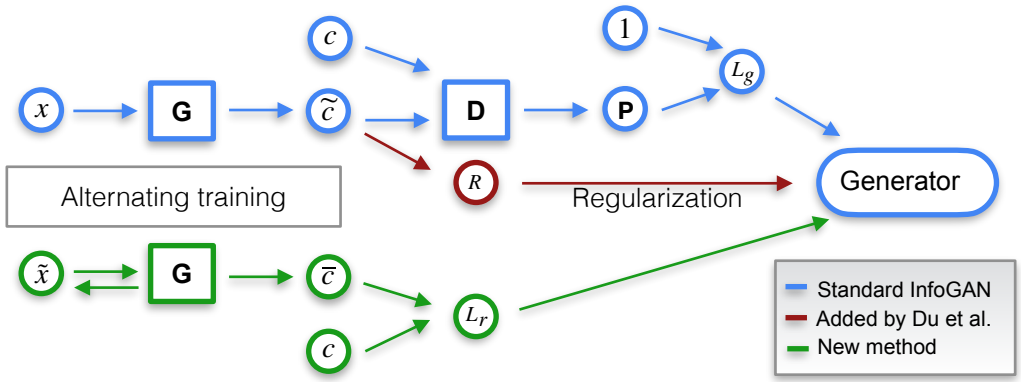


Fig. 2 Modified training process of generator network

training iteration and using it to back-propagate the derivatives of the loss function to the noise and latent variables. The reconstruction loss is added as an additional loss terms to the generator loss function. We then continue the regular GAN training procedure and alternate in this manner until the loss function has decreased sufficiently.

D. Performance measures for a generator neural network

In the introduction, we proposed three methods to measure the performance of an airfoil generator. We elaborate on these methods here.

Capturing the space of real airfoils: The goal of GAN is to learn the distribution of the real data. A well-trained generator should therefore capture the space of the real data as closely as possible. In terms of the space of real airfoils, we identify six geometric parameters that are important in airfoil design: maximum camber, maximum thickness, the x-coordinate of maximum thickness, leading and trailing edge angles, and leading edge radius. In addition, the camber and thickness lines of real airfoils can be viewed as low-frequency cosine waves. As such, we compute the first four Fourier coefficients of the camber and thickness curves to measure the frequency content. For this paper, we assume that space of real airfoils is spanned by these 14 parameters. However, we acknowledge that this is not an exhaustive list and that a real airfoils may not be uniquely be defined in terms of these parameters. We start by computing the 14 mentioned parameters for the 1320 airfoils contained in our data set. Next, we sample a trained generator to produce a set of airfoils, for which we compute the same parameters. Here, 'sampling' means drawing noise ($\mathbf{z}$) and latent ($\mathbf{c}$) variables from a normal distribution as inputs to the generator to produce airfoils. For a well-trained generator, the distributions of these parameters should closely match those of airfoils contained in the training set. To measure how close the distributions of airfoil parameters obtained with a generator are to those of the training airfoils, we use two frequently used methods for computing the statistical distance between two (probability) distributions. The first is the Kullback-Leibler (KL) divergence, which for discrete data is defined as

$$D_{KL}(P||Q) = \sum_{i}^{N} p(x_i) \log\left(\frac{p(x_i)}{q(x_i)}\right), \tag{7}$$

where N is the number of samples, $q(x)$ is the probability of the approximated (i.e., generated) data, and $p(x)$ is the probability of the true (i.e., training) data. Two noteworthy properties of the KL divergence are that it is non-symmetric, meaning $D_{KL}(P||Q) \neq D_{KL}(Q||P)$ and that it has no upper bound. Both properties are reversed for the second method we use for computing the statistical distance between two distributions, the Jensen-Shannon (JS) divergence, which is based on the KL divergence and is defined as

$$D_{JS}(P||Q) = \frac{1}{2} D_{KL}\left(P||\frac{P+Q}{2}\right) + \frac{1}{2} D_{KL}\left(Q||\frac{P+Q}{2}\right). \tag{8}$$

The JS divergence is symmetric and bounded between zero and one. In certain cases, this makes the JS divergence easier to interpret. For both, the KL and JS divergence, the lower the divergence, the closer the match between the distributions.

Reconstruction of real airfoils: GAN-based machine learning approaches are unsupervised by nature. While a well-trained generator network will produce data that, in theory, looks indistinguishable from the real data, there is no guarantee that the generator can also approximate an existing airfoil to within a certain tolerance. In Sec. IV.C, we proposed an additional training objective to the regular GAN method that aims at minimizing the reconstruction error during training. We define the reconstruction error ϵ_r as the maximum absolute difference between y-coordinates of reconstructed airfoils and the UIUC airfoils in out data set. For the entire set of training airfoils, the reconstruction loss can be written as

$$\epsilon_r = \frac{1}{N} \sum_{i}^{N} \max_{j} \left(\mid y_{r_i}^{j} - y_{u_i}^{j} \mid \right), \tag{9}$$

where N is the number of UIUC airfoils (1320) and y_r^j and y_u^j are the jth y-coordinates of the reconstructed and UIUC airfoils, respectively. Since UIUC airfoils are normalized by the chord length, ϵ_r can be interpreted as a percent error.

Generation of realistic airfoils only: During the GAN training procedure, the discriminator decides whether or not a generated airfoil is realistic by comparing it to airfoils in the training set. However, this judgment is dependent on the training method, meaning that no two discriminators are identical since their weights and biases are uniquely trained every training run. In addition, if mode collapse occurs, the discriminator's ability to classify whether an airfoil is realistic or not decreases significantly. As a result, we need a standard, general method to assess whether or not a generated airfoil is realistic that is independent of the training method. This method can be used to compare any two generators after training is completed. We develop this method as follows. Previously, we identified 14 parameters that span the space of all real airfoils, which we compute for each airfoil in the training set. This results in a data matrix $\mathbf{D} \in \mathbb{R}^{(1320\times14)}$. Next, we subtract the mean (column-wise) of $\mathbf{D}$ and compute the singular value decomposition as

$$\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}}, \tag{10}$$

where $\mathbf{U}$ contains the left singular vectors, $\mathbf{\Sigma}$ contains the singular values, and $\mathbf{V}$ contains the right singular vectors. Because we subtracted the mean before applying the SVD, the right singular vectors are equivalent to the eigenvectors of the covariance matrix of $\mathbf{D}$. They represent the directions of maximum variance (i.e., the principal components) of the 14 airfoil parameters across the training set. Therefore, by right-multiplying $\mathbf{D}$ by $\mathbf{V}$, we project $\mathbf{D}$ onto the principal components, represented by $\mathbf{V}$. The resulting matrix is another 1320×14 matrix. For each column we determine the minimum and maximum values, which we use as margins to decide whether an airfoil is realistic or not. For a set of N generated airfoils the procedure to decide how many airfoils are realistic then is as follows:

1) Compute the 14 airfoil parameters for each of the N airfoils
2) Subtract the mean **of the UIUC airfoils**
3) Compute the SVD
4) Right-multiply by the matrix containing the right singular vectors
5) Obtain a Boolean vector based on whether the data in each column lies within the margins determined by the airfoils in the training set
6) Sum the Boolean vector to obtain the number of realistic airfoils

We note that by determining the margins for the 14 airfoil parameters based on the SVD and not directly by the upper and lower bound of each parameter ensures that we capture the coupling between parameters in terms of the variance.

V. Results

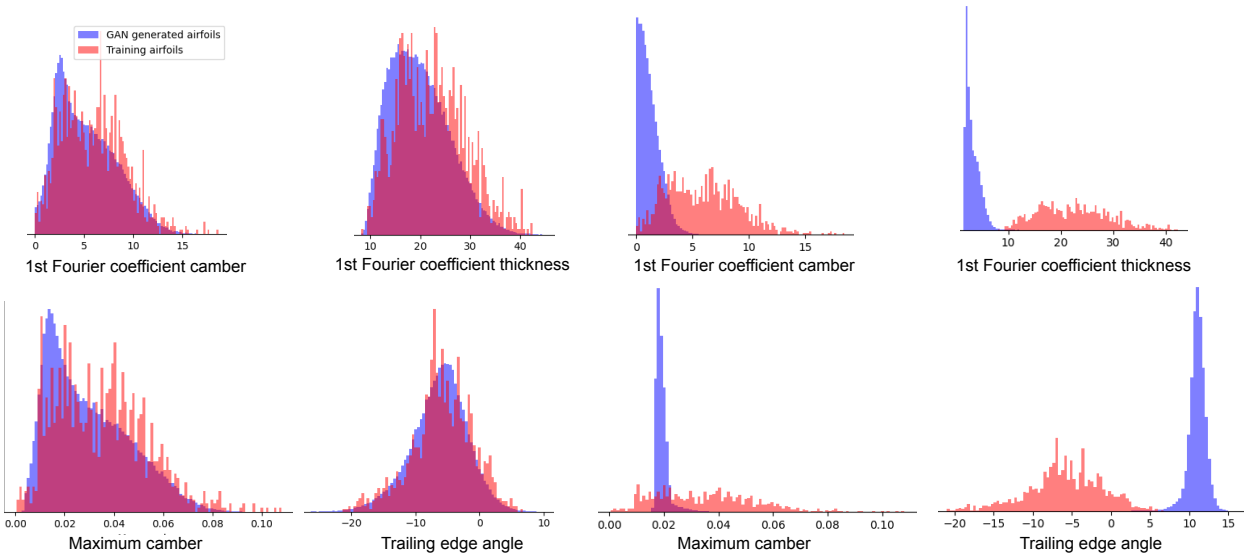
A. Verification of GAN-based airfoil parameterization

To verify the GAN-based training methods described in the previous section, we use the three measures proposed in the introduction to measure the performance of a trained Generator. We compare the results of our implementations of regular GAN and InfoGAN with the modified GAN-based training approach. In total, we train 27 generators each, using the normal GAN approach with and without the modified training, and 18 generators each, using InfoGAN, also with and without the modified training procedure. During each run we use different combinations of hyper-parameters for the regularization terms and number of training epochs. The architectures of the generator and discriminator neural networks, which we show in Tab. 5 of the appendix, were not tuned. In Fig. 10b of the appendix, we also show the training loss as a function of the training epochs for the generator, discriminator and the mutual information.

1. Capturing the space of real airfoils

After computing the 14 mentioned parameters for the 1320 airfoils contained in our data set, we sample a trained generator to produce 100,000 airfoils, for which we compute the same parameters. In Fig. 3, we show examples of distributions (in the form of histograms) that show good agreement (3a) compared to the training airfoils, and examples of distributions that do not (3b). Next, we compute the KL and JS divergence for the distributions of each of the 14 parameters of the training airfoils with respect to the airfoils generated by the generator neural net. The results are shown in Tab. 5, with all values being averaged over multiple training runs.

We see that the normal GAN approach does not approximate the true distributions of the 14 parameters well. With the modified training strategy, the agreement is worse when measured with the KL-divergence and marginally better when measured with the JS-divergence. The poor agreement of the distribution of parameters when using the normal GAN approach is an indication of mode collapse, described earlier. The InfoGAN approach produces significantly better results with the values for the KL and JS divergence indicating a closer match between the distributions of the real and generated data. Moreover, the modified training approach results in an average decrease of 20% in terms of the KL divergence and a decrease of 28% in terms of the JS divergence. This indicates that the modified training improves the approximations of the true distributions of airfoil parameters.



(a) Well-trained generator

(b) Poorly-trained generator

Fig. 3 Comparison of distributions of airfoil parameters for 100,000 randomly generated airfoils and the 1320 UIUC training airfoils

In practice, it is unrealistic to generate aerodynamic training data for 100,000 airfoils. Therefore, it may be useful to know how many generated airfoils are necessary to capture most of the space of realistic airfoils. In Fig. 4, we show the KL-divergence for the six geometric airfoil parameters as a function of the number of generated airfoils, using a generator that was trained with the modified InfoGAN approach. We see that for more than 1000 airfoils, the KL divergence does not decrease significantly, which is consistent with the number of airfoils in the training set.

2. Reconstruction of real airfoils

Based on our definition of the reconstruction error ϵ_r outline in Sec. IV.D, we compute ϵ_r for the entire set of UIUC airfoils, using generators trained with each training method. The results are shown in Tab. 2. We see that InfoGAN achieves better results than the regular GAN approach. In addition, the modified training procedure reduces ϵ_r further and achieves a minimum value across the entire data set of less than 0.5% in the InfoGAN case. This means that the maximum deviation in y-coordinates between a UIUC airfoil and the corresponding approximation by a generator is less than 0.5% of the chord length. Lastly, in Fig. 5, we show examples of a generator that can closely reproduce UIUC airfoils (5b) and one that cannot (5a).

	KL divergence			JS divergence		
Parameter/training method	GAN	Modified GAN	Δ (%)	GAN	Modified GAN	Δ (%)
Max camber	2.6516	1.9007	-28.3	0.4132	0.2926	-29.2
Max thickness	2.3384	7.2601	+210.5	0.4325	0.4566	+5.6
Max thickness x-coord.	3.8817	7.2685	+87.3	0.5609	0.5193	-7.4
Leading edge angle	6.8899	5.9737	-13.3	0.3964	0.4245	+7.1
Trailing edge angle	4.9195	4.5228	-8.1	0.4185	0.3578	-14.5
Leading edge radius	5.677	3.6239	-36.1	0.4021	0.3606	-10.3
1st camber Fourier coeff.	2.3368	1.8562	-20.7	0.3184	0.2672	-16.1
2nd camber Fourier coeff.	2.4645	2.3495	-4.7	0.4546	0.2928	-35.6
3rd camber Fourier coeff.	2.6319	3.3863	+28.7	0.3687	0.3690	+0.1
4th camber Fourier coeff.	3.1964	2.8072	-12.2	0.3555	0.3726	+4.8
1st thickness Fourier coeff.	4.5302	7.2995	+61.1	0.4246	0.4401	+3.6
2nd thickness Fourier coeff.	2.0144	7.0191	+248.4	0.3900	0.4409	+13.0
3rd thickness Fourier coeff.	4.0238	4.1356	2.8	0.4245	0.4502	+6.1
4th thickness Fourier coeff.	4.1164	4.0370	-1.9	0.4022	0.4156	+3.3
mean (GAN vs mod. GAN)	3.6910	4.5314	+36.7	0.4116	0.3896	-5.0
Parameter/training method	InfoGAN	Modified InfoGAN	Δ (%)	InfoGAN	Modified InfoGAN	Δ (%)
Max camber	0.2815	0.2954	+4.9	0.0473	0.0479	+1.3
Max thickness	0.5538	0.5244	-5.3	0.0827	0.0791	-4.4
Max thickness x-coord.	0.9642	0.9068	-6.0	0.1848	0.1194	-35.4
Leading edge angle	1.2660	1.0651	-15.9	0.1747	0.1099	-37.1
Trailing edge angle	1.6707	0.9508	-64.6	0.1838	0.0616	-66.5
Leading edge radius	1.4959	0.6911	-46.2	0.1730	0.0907	-47.6
1st camber Fourier coeff.	0.1946	0.2119	+8.9	0.0338	0.0340	+0.6
2nd camber Fourier coeff.	0.3639	0.3216	-11.6	0.0582	0.0435	-25.3
3rd camber Fourier coeff.	0.4896	0.4005	-18.2	0.1168	0.0780	-33.2
4th camber Fourier coeff.	0.5674	0.4523	-20.3	0.1270	0.0781	-38.5
1st thickness Fourier coeff.	0.4859	0.3855	-20.7	0.0510	0.0467	-8.4
2nd thickness Fourier coeff.	0.4087	0.4164	+1.9	0.0551	0.0503	-8.7
3rd thickness Fourier coeff.	0.5340	0.3346	-37.3	0.1055	0.0673	-36.2
4th thickness Fourier coeff.	0.8634	0.3938	-54.5	0.1662	0.0840	-49.5
mean (InfoGAN vs mod. InfoGAN)	0.7193	0.5733	-20.3	0.1114	0.0804	-27.8

Table 1 Summary of KL and JS divergence for different airfoil parameters and different GAN training methods

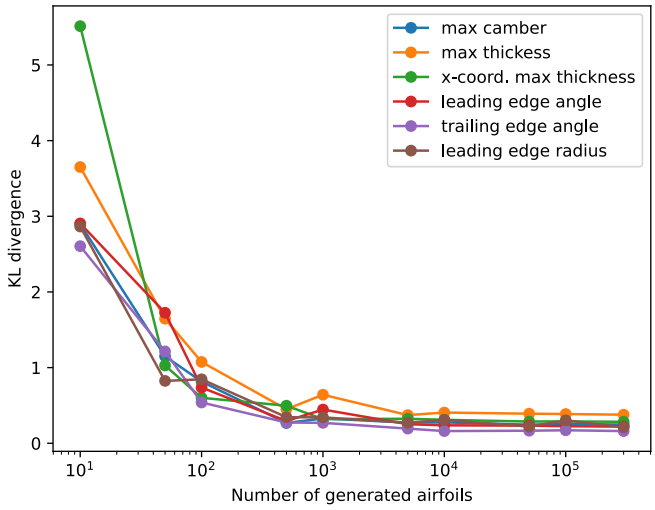
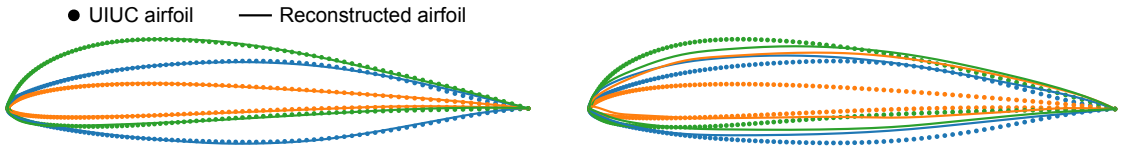


Fig. 4 KL divergence of airfoil geometric parameters vs number of generated airfoils



(a) Well-trained generator (b) Poorly-trained generator

Fig. 5 Reconstruction of UIUC airfoils with a well and poorly-trained generator

Reconstruction error ϵ_r (%)	**GAN**	**Modified GAN**	**InfoGAN**	**Modified InfoGAN**
Min (best generator)	1.38	1.23	0.81	0.48
Max (worst generator)	17.23	9.5	2.31	0.75
Mean (of all trained generators)	4.93	2.75	1.21	0.61

Table 2 Summary of reconstruction errors of GAN-based training methods

3. Generation of realistic airfoils only

Based on the procedure outlined in Sec. IV.D to determine whether or not a generated airfoil is realistic, we use generators trained with each training approach to generate 100,000 airfoils. We compute the percentage of real airfoils and show the results in Tab. 3. For the normal GAN approach (with and without modified training), we note that in some training cases the generator is either able to produce 100% or 0% realistic airfoils only. In those cases, the generator only learns a single or very similar airfoils, meaning that the noise input variables have little to no impact on the shape of the airfoil. This is a manifestation of the difficulties associated with normal GAN training, in particular mode collapse. For the InfoGAN approach, this is not a problem due to the improved ability to extract features of the airfoil through the latent variables.

(a) Well-trained generator (b) Poorly-trained generator

Fig. 6 Generation of random airfoils with a well and poorly-trained generator

Number of realistic airfoils per 100,000 (%)	**GAN**	**Modified GAN**	**InfoGAN**	**Modified InfoGAN**
Max (best generator)	100.00	100.00	99.80	99.88
Min (worst generator)	0.00	0.00	32.34	76.77
Mean (of all trained generators)	71.32	67.65	93.69	92.51

Table 3 Ability of GAN-based training methods to generate realistic airfoils only

4. Feature extraction through latent variables

As stated before, one of the goals of InfoGAN is to untangle information about the data through latent variables $\mathbf{c}$, which encode hidden features. For the application of InfoGAN to the airfoil parameterization problem, these latent variables ideally correspond to certain geometric features of an airfoil. In Fig. 7, we show the effect of changing four latent variables (c_1, c_4, c_8, and c_{10}) while keeping the rest, including the noise variables $\mathbf{z}$, constant at zero. Since GAN is an unsupervised learning technique, it is impossible to predict the information a specific latent variable encodes beforehand. In this case, it seems that latent variables c_1 and c_3 change the airfoil shape near the trailing and leading edge, respectively, while latent variables c_8 and c_{10} change the thickness and camber of the airfoil.

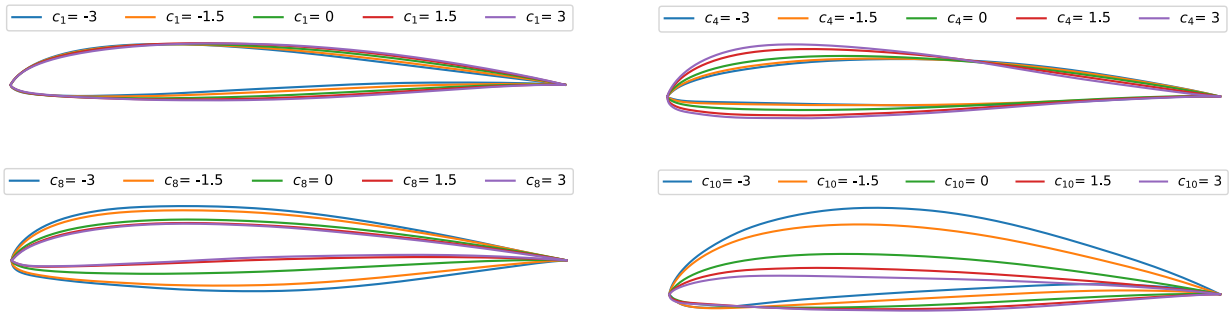


Fig. 7 Effect of changing four latent variables of a well-trained InfoGAN generator on the airfoil geometry

5. Summary of GAN results

Based on the results of the previous subsections, we qualitatively summarize the GAN-based airfoil parameterization techniques in Tab. 4 in terms of the three performance measures.

Properties/ performance measure	**GAN**	**Modified GAN**	**InfoGAN**	**Modified InfoGAN**
Capturing the space of real airfoils	poor	poor	decent	good
divergence [KL, JS]	[3.69, 0.41]	[4.53, 0.39]	[0.72, 0.11]	[0.57, 0.08]
Reconstruction of real airfoils	poor	limited	decent	good
Reconstruction error ϵ_r (%)	4.93	2.75	1.21	0.61
Generation of realistic airfoils only	poor	poor	good	good
% of real airfoils (per 100,000)	71.32	67.65	99.80	99.88
Feature extraction via noise/latent variables	poor	poor	good	good
Conclusion	not recommended	not recommended	recommended	recommended

Table 4 Summary of performance of generators trained with different methods

B. Aerodynamic prediction

As stated in Sec. IV.A, we train a generator, using the modified InfoGAN approach and use it to randomly produce 500 airfoils for which we gather XFOIL data for different combinations of angle of attack, Reynolds number, and Mach number. For the preliminary results of this paper, we train two standard artificial neural networks (for C_l and C_d) on 85% of the available data (about 280,000 XFOIL evaluations) and use the remainder for testing the model. The Adam optimizer [34] was used to minimize the mean-squared error loss function with a learning rate of 0.002, a batch size of 200, and number of training epochs of 800. These hyper-parameters were found to yield the best training and testing R2 scores. The architectures of the neural networks (see Tab. 5 in appendix VI.A) were not tuned. In Fig. 8, we show the performance of the trained neural networks (y-axis) on the test set. For both C_l and C_d, there is close correlation between the test XFOIL data (x-axis) and the prediction by the neural networks, indicated by the high R2 score.

Next, we use the trained neural networks to compute the drag polar (see Fig. 9a) and the lift-to-drag ratio as a function of the angle of attack (see Fig. 9b) for four randomly drawn airfoils from our training set. The Reynolds and Mach numbers were held constant at 1,000,000 and 0.3, respectively. The figures show decent agreement between the predictions of the neural network and the XFOIL data. However, the prediction of the neural network shows some cusps and non-smoothness, especially for higher angles of attack, which could be problematic during design optimization. A likely explanation for this is that the model is overfitted and has learned noisy features of the training data. Potential solutions for this could be to smooth the training data beforehand, remove outliers, reduce the complexity of the neural network (i.e., reduce the number of hidden layers and neurons per layer) or reduce the number of training epochs. The purpose of these measures is to find an optimal compromise for the bias-variance trade-off, such that the model does not overfit the data, while retaining good prediction accuracy. We note that these are preliminary results and more work is needed before this supervised learning model can be integrated into physics-based models for airfoil modeling.

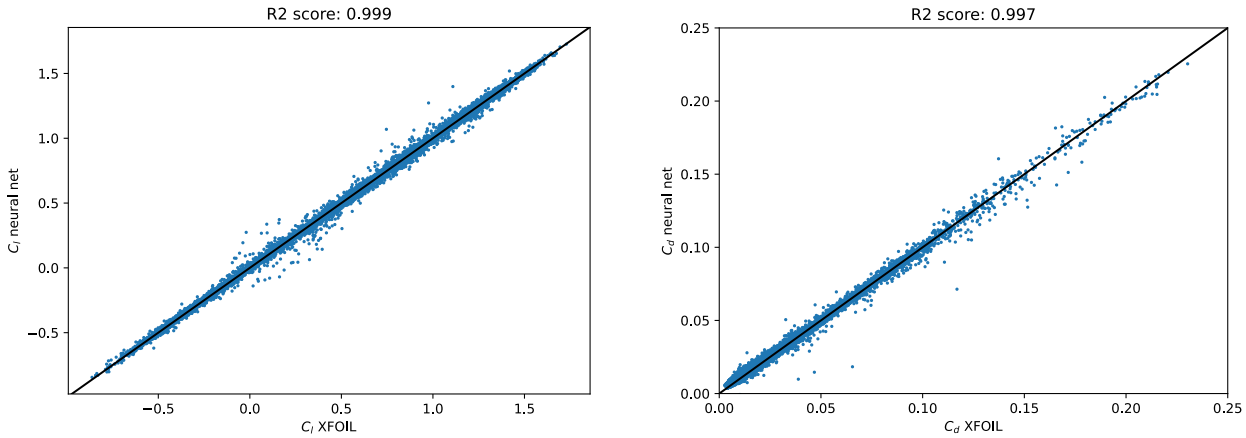
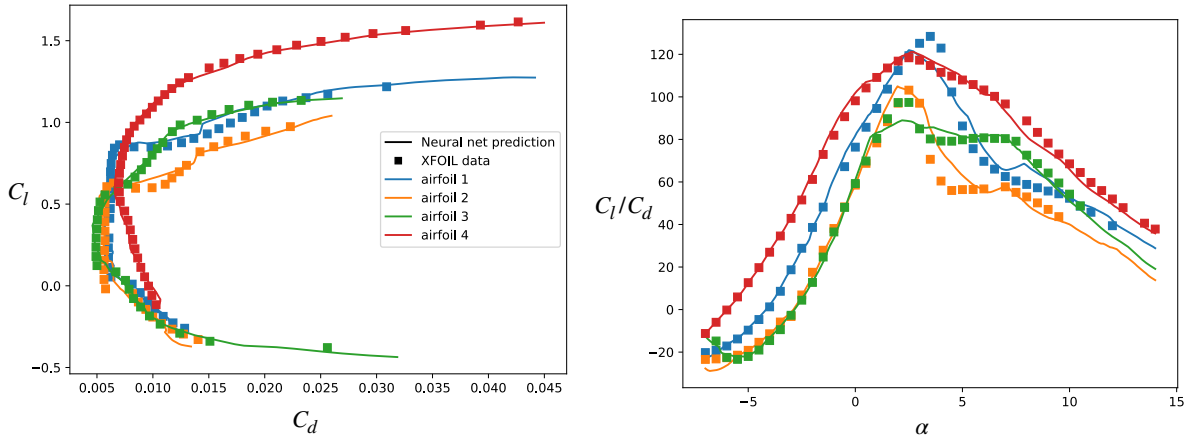


Fig. 8 Preliminary verification of artificial neural net on testing set for C_l and C_d



(a) Drag polar

(b) Lift-to-drag ratio vs. AoA

Fig. 9 Preliminary results for aerodynamic predictions of C_l and C_d for generated airfoils

VI. Conclusion

In this paper, we developed a method for constructing airfoil shape parameterizations based on generative adversarial networks (GANs), with a critical modification in the training objective. This modification is the inclusion of an additional training term that captures the reconstruction error between a set of known airfoils and a corresponding set of airfoils that represent the closest approximations yielded by the generator (i.e., the airfoil parameterization). We also proposed three measures for evaluating the effectiveness of a particular parameterization, which we used to assess and compare our modified training method to existing methods. These three measures are as follows. 1) The distributions of important airfoil properties such as maximum camber or thickness show close agreement between the training and the generated data. 2) The generator can approximate real airfoils with reasonable accuracy. 3) The generator only produces realistic airfoils.

Results show that a generator trained with the modified training approach is able to reconstruct any airfoil in our training set to within a maximum error of less than 0.5%, while over 99.8% (per 100,000) of all randomly generated airfoils are realistic, which constitutes improvements of 41% and 29%, respectively, over existing methods. In addition, the similarity of distributions of key airfoil properties between the training data and generated airfoils is improved by over 20% when measured with Kullback-Leibler divergence. Lastly, in some preliminary studies, we used a generator to produce random airfoils for which we obtained aerodynamic training data, based on which we trained a supervised learning model to predict the lift and drag coefficient. Based on the results of this work, we anticipate that an airfoil generator trained with the proposed modified training method can improve the optimization of airfoil shapes to find the most optimal design, when coupled with an aerodynamic prediction model.

This work was motivated by a renewed interest in low-fidelity aerodynamic modeling for aircraft conceptual design, due to the emergence of urban air mobility. Physics-based models like blade element momentum theory or the vortex lattice method, which are often used during conceptual design, can be benefit from a well-trained airfoil machine learning model. To enable the integration of such a machine learning model into physics-based models, more future work is needed. This includes exploring other unsupervised learning techniques for parameterizing the airfoil shape such as other versions of GAN or variational autoencoders. In addition we will expand the aerodynamic training data set and also train supervised neural networks for the prediction of the pressure distribution and boundary layer parameters.

Acknowledgements

The material presented in this paper is supported by NASA under award No. 80NSSC21M0070.

Appendix

A. Architecture of neural networks

In Tab. 5 we show the architectures of the neural networks used to train the generator, discriminator and the aerodynamic regressions. We note that in case of the discriminator, the activation function of the output layer depends on what loss function is used. If the binary cross-entropy loss function is used, the activation function would be a `sigmoid` function, whereas if the mean-squared loss function is used, a `linear` activation would work better.

Layer (neurons, activation)	**Unsupervised learning**		**Supervised learning**	
	Generator	Discriminator	Lift coefficient	drag coefficient
Input	32, `ReLU`	602, `ReLU`	35, `ReLU`	35, `ReLU`
Hidden layer 1	120, `ReLU`	400, `ReLU`	100, `ReLU`	100, `ReLU`
Hidden layer 2	120, `ReLU`	400, `ReLU`	100, `ReLU`	100, `ReLU`
Hidden layer 3	120, `ReLU`	300, `ReLU`	100, `ReLU`	100, `ReLU`
Hidden layer 4	120, `ReLU`	300, `ReLU`	75, `ReLU`	75, `ReLU`
Hidden layer 5	–	100, `ReLU`	75, `ReLU`	75, `ReLU`
Hidden layer 6	–	–	50, `ReLU`	50, `ReLU`
Output	36, `linear`	1, `linear` or `sigmoid`	1, `linear`	1, `linear`

Table 5 Summary neural network architectures

B. Training loss

We show sample plots of the training loss functions used to train the unsupervised and supervised learning models in Fig. 10 as a function of the number of training epochs. In the case of the modified InfoGAN training case (Fig. 10b), the periodic spikes for the generator and discriminator show the effects the alternating training objective of reconstructing UIUC airfoils. With the provided information regarding the architecture of the neural networks, these plots should be reproducible.

References

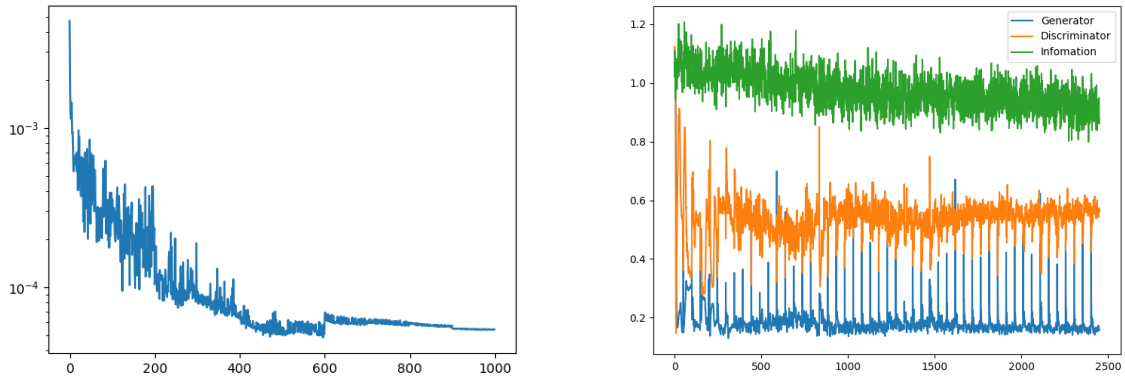
[1] Drela, M., "XFOIL: An analysis and design system for low Reynolds number airfoils," *Low Reynolds number aerodynamics*, Springer, 1989, pp. 1–12.

[2] Li, J., Bouhlel, M. A., and Martins, J. R., "Data-based approach for fast airfoil analysis and optimization," *AIAA Journal*, Vol. 57, No. 2, 2019, pp. 581–596.

[3] Selig, M. S., "UIUC airfoil data site," 1996.

[4] Chen, W., and Fuge, M., "BezierGAN: Automatic Generation of Smooth Curves from Interpretable Low-Dimensional Parameters," *arXiv preprint arXiv:1808.08871*, 2018.

[5] Chen, W., Chiu, K., and Fuge, M., "Aerodynamic design optimization and shape exploration using generative adversarial networks," *AIAA Scitech 2019 Forum*, 2019, p. 2351.



(a) C_I training loss

(b) Modified InfoGAN training loss

Fig. 10 Sample plots of loss functions for unsupervised and supervised learning methods

[6] Du, X., He, P., and Martins, J. R., "Rapid airfoil design optimization via neural networks-based parameterization and surrogate modeling," *Aerospace Science and Technology*, Vol. 113, 2021, p. 106701.

[7] Chen, W., Chiu, K., and Fuge, M. D., "Airfoil design parameterization and optimization using bézier generative adversarial networks," *AIAA journal*, Vol. 58, No. 11, 2020, pp. 4723–4735.

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

[9] Yonekura, K., Miyamoto, N., and Suzuki, K., "Inverse airfoil design method for generating varieties of smooth airfoils using conditional WGAN-gp," *arXiv preprint arXiv:2110.00212*, 2021.

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

[11] Yonekura, K., and Suzuki, K., "Data-driven design exploration method using conditional variational autoencoder for airfoil design," *Structural and Multidisciplinary Optimization*, Vol. 64, No. 2, 2021, pp. 613–624.

[12] Yonekura, K., Wada, K., and Suzuki, K., "Generating various airfoils with required lift coefficients by combining NACA and Joukowski airfoils using conditional variational autoencoders," *Engineering Applications of Artificial Intelligence*, Vol. 108, 2022, p. 104560.

[13] Kou, J., Botero-Bolívar, L., Ballano, R., Marino, O., de Santana, L., Valero, E., and Ferrer, E., "Aeroacoustic airfoil shape optimization enhanced by autoencoders," *Expert Systems with Applications*, 2023, p. 119513.

[14] Jing, W., Runze, L., Cheng, H., Haixin, C., Cheng, R., Chen, Z., and Zhang, M., "An inverse design method for supercritical airfoil based on conditional generative models," *Chinese Journal of Aeronautics*, Vol. 35, No. 3, 2022, pp. 62–74.

[15] Bouhlel, M. A., He, S., and Martins, J. R., "Scalable gradient–enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes," *Structural and Multidisciplinary Optimization*, Vol. 61, No. 4, 2020, pp. 1363–1376.

[16] Zhu, L., Zhang, W., Kou, J., and Liu, Y., "Machine learning methods for turbulence modeling in subsonic flows around airfoils," *Physics of Fluids*, Vol. 31, No. 1, 2019, p. 015105.

[17] Singh, A. P., Medida, S., and Duraisamy, K., "Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils," *AIAA journal*, Vol. 55, No. 7, 2017, pp. 2215–2227.

[18] Sun, X., Cao, W., Liu, Y., Zhu, L., and Zhang, W., "High Reynolds number airfoil turbulence modeling method based on machine learning technique," *Computers & Fluids*, Vol. 236, 2022, p. 105298.

[19] Zhang, W., Zhu, L., Liu, Y., and Kou, J., "Machine learning methods for turbulence modeling in subsonic flows over airfoils," *arXiv preprint arXiv:1806.05904*, 2018.

[20] Singh, A. P., Duraisamy, K., and Zhang, Z. J., "Augmentation of turbulence models using field inversion and machine learning," *55th AIAA Aerospace Sciences Meeting*, 2017, p. 0993.

[21] Zhu, L., Zhang, W., Sun, X., Liu, Y., and Yuan, X., "Turbulence closure for high Reynolds number airfoil flows by deep neural networks," *Aerospace Science and Technology*, Vol. 110, 2021, p. 106452.

[22] Yang, M., and Xiao, Z., "Improving the k–ω–γ–Ar transition model by the field inversion and machine learning framework," *Physics of Fluids*, Vol. 32, No. 6, 2020, p. 064101.

[23] Holland, J. R., Baeder, J. D., and Duraisamy, K., "Towards integrated field inversion and machine learning with embedded neural networks for RANS modeling," *AIAA Scitech 2019 forum*, 2019, p. 1884.

[24] Guo, X., Li, W., and Iorio, F., "Convolutional neural networks for steady flow approximation," *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 481–490.

[25] Sekar, V., Jiang, Q., Shu, C., and Khoo, B. C., "Fast flow field prediction over airfoils using deep learning approach," *Physics of Fluids*, Vol. 31, No. 5, 2019, p. 057103.

[26] Yilmaz, E., and German, B., "A convolutional neural network approach to training predictors for airfoil performance," *18th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2017, p. 3660.

[27] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative adversarial nets," *Advances in neural information processing systems*, Vol. 27, 2014.

[28] Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., and Sutton, C., "Veegan: Reducing mode collapse in gans using implicit variational learning," *Advances in neural information processing systems*, Vol. 30, 2017.

[29] Li, J., Du, X., and Martins, J. R., "Machine learning in aerodynamic shape optimization," *Progress in Aerospace Sciences*, Vol. 134, 2022, p. 100849.

[30] Bau, D., Zhu, J.-Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., and Torralba, A., "Seeing what a gan cannot generate," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4502–4511.

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

[32] Arjovsky, M., Chintala, S., and Bottou, L., "Wasserstein generative adversarial networks," *International conference on machine learning*, PMLR, 2017, pp. 214–223.

[33] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P., "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *Advances in neural information processing systems*, Vol. 29, 2016.

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