Inverse Airfoil Design Method for Generating Varieties of Smooth Airfoils Using Conditional WGAN-GP

Kazuo Yonekura (yonekura@struct.t.u-tokyo.ac.jp)
University of Tokyo
Nozomu Miyamoto
University of Tokyo
Katsuyuki Suzuki
University of Tokyo

Research Article

Keywords: Generative adversarial networks (GAN), computational fluid dynamics (CFD), variational autoencoder (VAE)

Posted Date: June 22nd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-618399/v1

Inverse airfoil design method for generating varieties of smooth airfoils using conditional WGAN-GP

Kazuo Yonekura[1,2,*], **Nozomu Miyamoto**[1], **and Katsuyuki Suzuki**[1]

[1]The University of Tokyo, Department of Systems Innovation, Tokyo, 113-8656, JAPAN
[2]IHI Corporation, Department of Mathematical and Computational Engineering, Yokohama, 235-8501, JAPAN
[*]yonekura@struct.t.u-tokyo.ac.jp

ABSTRACT

Machine learning models are recently utilized for airfoil shape generation methods. It is desired to obtain airfoil shapes that satisfies required lift coefficient. Generative adversarial networks (GAN) output reasonable airfoil shapes. However, shapes obtained from ordinal GAN models are not smooth, and they need smoothing before flow analysis. Therefore, the models need to be coupled with Bézier curves or other smoothing methods to obtain smooth shapes. Generating shapes without any smoothing methods is challenging. In this study, we employed conditional Wasserstein GAN with gradient penalty (CWGAN-GP) to generate airfoil shapes, and the obtained shapes are as smooth as those obtained using smoothing methods. With the proposed method, no additional smoothing method is needed to generate airfoils. Moreover, the proposed model outputs shapes that satisfy the lift coefficient requirements.

1 Introduction

Airfoil designs incur intensive computation costs. The airfoil design task is to obtain airfoil shapes that satisfy specific requirements. Recently, shapes have been designed iteratively using computational fluid dynamics (CFD). CFD-based design optimization methods have been studied for years[1,2]. However, CFD computation is a time-consuming process. Recently, data-driven methods, such as machine learning, have been employed for airfoil generation. Once a machine learning model is trained, it outputs immediately. As the model can be reused for other design tasks, the computation time is sufficiently smaller than that of CFD-based iterative methods. Several studies on generative adversarial networks (GAN) have been reported.[3] proposed airfoil GAN that generates airfoil shapes regardless of requirements. Airfoil generally needs smooth curves, but output shapes using GAN are not always smooth due to noise during learning and generation. To obtain smooth shapes,[4] proposed BézierGAN in which airfoil is represented by Bézier curves, which successfully produced smooth curves.[5] generated low- and high-lift-coefficient airfoils using conditional GAN (CGAN). Most shapes reported in the literature have jaggy lines, which hinder further CFD analyses. Consequently, the Savitzky–Golay filter[6] has been employed to smoothen the curves. To date, generating smooth airfoil shapes without using a smoothing method or free curves is challenging. Herein, we propose conditional Wasserstein GAN with gradient penalty (CWGAN-GP) model to overcome this issue. Furthermore, in existing CGAN models[5], the conditional label is binary, i.e., high or low, and it cannot generate airfoils that have specific lift coefficients. The proposed model uses the lift coefficient as a continuous label; therefore, shapes that have a specific lift coefficient are obtained.

Another method utilizes variational autoencoder (VAE) models for airfoil generation.[7] used conditional VAE to generate airfoils that meet specific lift coefficients.[8] compared two types of CVAE model, the CVAE with normal distribution and von Mises Fisher distribution, and investigated the generated shapes. They also proved that if different types of airfoils are combined and fed as training data, the CVAE model generates airfoils that have mixed features of the different airfoils used. Shapes obtained from these VAE models are smooth enough for CFD analysis. However, whether smooth airfoils can be generated using GAN is still an open problem. One of the differences between GAN and VAE is that the loss function in VAE is generally element-wise metrics between input and generated data, and VAE does not consider the properties of the data. GAN is useful because it considers not the element-wise metrics but the properties of the airfoils. VAEGAN[9] coupled VAE and GAN to combine the features of VAE and GAN.

Besides airfoils, data generation methods for various applications, including marine propeller[10] and turbine film-cooling holes, as well as airfoils[11] and architecture[12], have been reported.[13] used a deep generative model with topology-optimized shapes to develop various types of optimal shapes. Generative models are useful for three-dimensional (3D) objects[14,15]. employed generative models for 3D shapes and generated 3D aircraft shapes. However, flow analysis was not conducted on the generated shapes. Employing the shape generation methods in various fields is easy if smoothing methods are not required.

Various GAN models have been proposed previously. Ordinal GAN uses the Jensen–Shannon (JS) divergence to measure the similarity of probability distributions. However, instability in the training of GAN, such as mode collapse[16] and gradient dissipation[17], has been reported Mode collapse is a situation wherein a generator outputs the same data even when the input latent vector is different. Wasserstein GAN (WGAN) was proposed to overcome gradient dissipation[18]. WGAN uses the earth mover's (EM) distance to measure the distance of probability distributions. WGAN with gradient penalty (WGAN-GP)[19] was proposed to improve stability in training WGAN. To achieve our goal, we propose conditional WGAN-GP, which uses WGAN-GP conditionally. Conditional GAN usually employs discrete labels, but because we aim to obtain continuous lift coefficients, the continuous label is employed.

The rest of the paper is organized as follows: the dataset is described in section 2; section 3 introduces the CGAN and WGAN-GP models; the introduced GAN models were trained using NACA airfoil data, as described in section 4, and the results are compared; the conclusion of the study is presented in section 5.

2 Airfoil data

NACA 4-digit airfoil data[20] were used for the training. NACA 4-digit airfoil is defined by three parameters: max camber, the position of the max camber, and thickness. One thousand data of four-digit airfoils can be defined, i.e., from 0000 to 9999, but some of them are not useful for airfoils. Herein, the lift coefficient was calculated for all the four-digit airfoils, and those whose C_L could not be calculated or $C_L < 0$ or $C_L > 2.0$ were eliminated. After elimination, the total number of airfoils was 3709. The calculation condition of C_L was as follows: angle of attack was 5°, and Reynolds number was 3.0×10^6. Xfoil[21] was used for calculating C_L, which employed a panel method for computation. The histogram of C_L is shown in Figure 1. The data are almost uniform, except that the amount of data for $1.2 < C_L$ is relatively small.

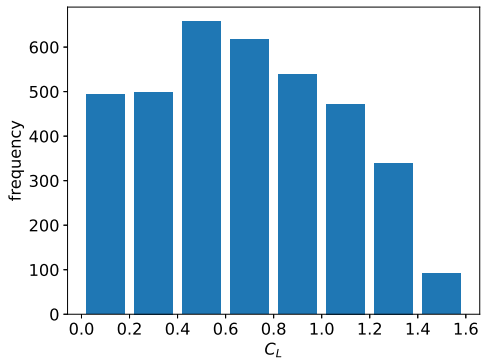


Figure 1. Histogram of C_L.

To handle airfoils, 248 points on the airfoil outline were sampled. Coordinates x_i, y_i were gathered to form a vector $x = (x_1, \ldots, x_{248}, y_1, \ldots, y_{248})$. An example of airfoil discretization is shown in Figure 2. The Xfoil computation was conducted using the 248 data points. Xfoil needed more than 120 points around the airfoil for stable computation.

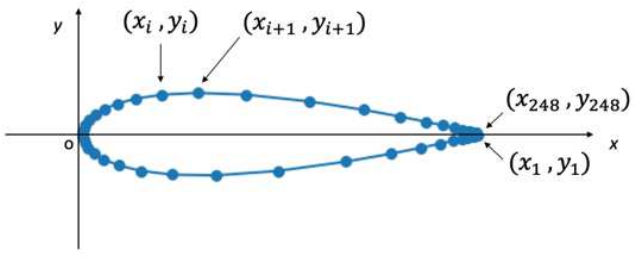


Figure 2. Example of airfoil discretization.

3 GAN models

3.1 GAN and CGAN

GAN model comprises a generator and a discriminator. The generator inputs latent vectors z and outputs shape vectors x. The discriminator inputs shape vectors x and outputs whether the input data is true or fake. The architecture of the network is illustrated in Figure 3 (a). The generator and discriminator are represented by $G : z \mapsto x$ and $D : x \mapsto [0,1]$, respectively, where $D(x) = 1$ implies that discriminator judged the input x to be true data, and $D(x) = 0$ implies that the data were fake. GAN model was trained based on $\min_G \max_D V(D,G)$. The loss function $V(D,G)$ is formulated as

$$\begin{aligned} V(D,G) &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))], \\ &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x))], \end{aligned}$$

where $E[\cdot]$ implies expectation, and $p_r(x)$ $p_z(x)$ represents the probability of occurrence of x and z. The network was optimized to minimize V with respect to the generator and maximize V with respect to the discriminator. The function V indicates a higher value if the discriminator succeeds in determining whether the input data is true or fake. Otherwise, if the generator succeeds to cheat the discriminator, V indicates a lower value because $1 - D(G(z))$ becomes smaller than 1. The generator was trained to cheat the discriminator, and the discriminator was trained not to be cheated by the generator. Because two networks were trained adversarial, the model is called a generative adversarial network.

When the optimal $D = D^*$ is given, the loss function is expressed as

$$V(D^*,G) = 2D_{JS}(p_r||p_g) - 2\log 2, \tag{1}$$

where D_{JS} represents the JS divergence. JS divergence measures the distance between two probability distributions, i.e., the distribution of real and fake data. Thus, minimizing JS divergence implies matching p_r and p_g.

Conditional GAN is developed to generate data with specific labels. The architecture of the CGAN is illustrated in Figure 3 (b). In the CGAN, both generator and discriminator has label y to be input. Therefore, the generator outputs data with a specified conditional label. The discriminator judges fake or true considering the input data and label. The label y is usually discrete, but herein, the continuous label was used to handle the continuous lift-coefficient label. The loss function is expressed as

$$V(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z|y)))].$$

The generator sometimes outputs the same data x even if the input latent vector z is different. Thus, the function G becomes an identity function. This situation is not desired, and it is called a mode collapse.

(a) GAN (b) Conditional GAN

Figure 3. Architecture of GAN and CGAN.

3.2 WGAN-GP

WGAN[18] has been proposed to overcome the gradient dissipation. The gradient dissipation in GAN is attributed to increased JS divergence in (1). In WGAN, Wasserstein distance, which is also called the EM distance, is employed. The EM distance $W(\cdot,\cdot)$ between two probability distribution p_r and p_g is defined by

$$W(p_r,p_g) = \inf_{\gamma\sim\Pi(p_r,p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|].$$

Using the Kantorovich–Rubinstein duality[22], the EM distance is equivalent to

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x\sim p_r}[f(x)] - \mathbb{E}_{y\sim p_g}[f(y)],$$

where the supremum is taken over all K-Lipshitz functions. When a parameterized family of K-Lipshitz function $\{f_w\}_{w\in\mathscr{W}}$ is given, the EM distance is calculated by

$$W(p_r, p_g) = \max_{w\in\mathscr{W}} \left[\mathbb{E}_{x\sim p_r}[f_w(x)] - \mathbb{E}_{z\sim p_z}[f_w(g_\theta(z))]\right].$$

In the neural network model, f_w and g_θ correspond to the discriminator and generator, respectively. The parameter w of the family $\{f_w\}$ is the weights of the neural network. To guarantee K-Lipshitzness, weight clipping is employed. Using the EM distance, the loss function of WGAN is expressed as $V_{\mathrm{WGAN}}(f_w, g_\theta) = W(p_r, p_g)$, and WGAN solves $\min_\theta \max_w [V_{\mathrm{WGAN}}(f_w, g_\theta)]$. It is proven[19] that the gradient of the optimal f_w has norm 1 almost everywhere. Therefore, WGAN-GP adds a penalty term to ensure that the gradient of f_w becomes 1. Consequently, the loss function of WGAN-GP is

$$V_{\text{WGAN-GP}}(f_w, g_\theta) = V_{\mathrm{WGAN}}(f_w, g_\theta) + \lambda \mathbb{E}_{x\sim p_r}[(\|\nabla_x f_w(x)\|_2 - 1)^2].$$

3.3 CWGAN-GP for airfoil generation

The architecture of the CGAN was used with Wasserstein distance and gradient penalty. The label node input the lift coefficient as a continuous label. The dimension of latent z was set as 6, and the architecture of the generator comprised five layers whose nodes were 64, 128, 256, 512, and 496. The architecture of the discriminator comprised three layers whose nodes were 512, 256, and 1. The hyperparameters in training are shown in Table 1. The code was implemented using PyTorch[23]. The computation was performed on AMD EPYC 7402P 2.80GHz CPU with NVIDIA RTX A6000 GPU.

Table 1. Hyperparameters in training GAN models.

params	value
optimization algorithm	Adam
learning rate	0.0001
number of training steps for D per iter.	5

4 Numerical example

4.1 Comparison with CGAN, CWGAN-GP, and other generative models

CGAN and CWGAN-GP models were trained individually using the NACA dataset. The latent dimension was set as $d = 3$ and $d = 6$. Once trained, shapes were generated using labels in $C_L = 0.1, 0.11, \ldots 1.5$, and 20 shapes were generated for each C_L using random latent vector. Then, C_L was recalculated using XFoil for the generated shapes. If the absolute error between the calculated and specified label was larger than 0.2, the shape generation was considered a failure. Among all the generated shapes, nonconvergence, success, and failure rates are shown in Table 2. To measure the variety of the generated shapes, μ was calculated using the following equation; $\mu = \frac{1}{N}\sum_i \left\|x_i - \sum_j x_j\right\|^2$. Comparing CGAN and CWGAN-GP, the nonconvergence rate was smaller in CWGAN-GP ($d = 3$), which implied that the generated shapes were smoother. Moreover, the success rate was significantly large in CWGAN-GP ($d = 3$).

CGAN with smoothing method[5] is also compared in Table 2. The data was adapted from the literature[5], and the training dataset was different from that of CGAN and CWGAN-GP. Further, in[5], the label is high or low lift by drag ratio, which is different from that obtained herein. Therefore, we could not directly compare the results to the labels. Therefore, the success and failure rates are shown in parentheses. The N-CVAE and S-CVAE models[7,8] were also compared. For these models, the training dataset and success threshold are the same as those obtained herein. Although CWGAN-GP showed the highest nonconvergence ratio, no significant difference existed between CWGAN-GP, CGAN with smoothing, and S-CVAE. Consequently, CWGAN-GP generated smooth shapes without any smoothing methods.

The mean squared error (MSE) was smaller in CGAN than CWGAN-GP. Moreover, μ was smaller in CGAN than CWGAN. This difference indicates that the generated shapes from CGAN were similar. In CGAN, the success rate was small, and the generated shapes were in good agreement with the labels, although the variety of shapes was limited. Conversely, in CWGAN-GP, the success rate and variation of shapes were high, and error with respect to C_L was larger than that of CGAN.

Table 2. Success rates and errors of generated shapes.

	not converge↓	failure↓	success↑	MSE↓	μ↑
CGAN ($d = 3$)	26.8%	24.4%	48.8%	0.0470	0.152
CGAN ($d = 6$)	26.6%	25.7%	47.7%	0.050	0.139
CGAN with smoothing[5]	10%	(15%)	(75%)	-	-
CWGAN-GP ($d = 3$)	9.6%	15.3%	75.1%	0.0469	0.320
CWGAN-GP ($d = 6$)	14.6%	23.8%	61.6%	0.0374	0.319
N-CVAE[7]	12.0%	13.0%	75.0%	0.027	0.226
S-CVAE[8]	10.2%	15.0%	74.8%	0.020	0.317

Among all generative models, CWGAN-GP and S-CVAE showed high μ, and MSE of CGAN and CWGAN-GP was higher than that of N- and S-CVAE. Consequently, CWGAN-GP is the best among the GAN models. VAE models are better than GAN models considering MSE and μ. VAE is trained to minimize the point-wise errors of the input and generated shapes. Conversely, GAN models do not consider point-wise errors but the properties of the shapes, i.e., the generator is trained to cheat the discriminator. Therefore, in principle, VAE models cannot generate completely new shapes.

4.2 Generated shapes

The labels and recalculated C_Ls are plotted in Figure 4 to visualize errors. In CGAN, the recalculated C_L was slightly lower in all labels. The highest calculated C_L was approximately 1.0. For the labels lower than 0.2, the points did not appear. It implies that the XFoil calculation of all generated shapes did not converge. Conversely, most generated shapes of CWGAN-GP converged. However, the error of the recalculated value and label was higher than that of CGAN. This is attributed to the varieties of generated shapes; the varieties of the shapes of CGAN were less than those of CWGAN. Among the generated shapes, those of CGAN and CWGAN at $C_L \in \{0, 1, 0.5, 1.0, 1.4\}$ are shown in Figure 5 and Figure 6. The shapes in blue indicate when the Xfoil computation converged, and those in red indicate otherwise. The recalculated C_Ls is also shown in the figures. In CGAN $C_L = 0.1$, almost all the shapes did not converge. The shapes formed airfoil, but they contained small bumps. Conversely, in CWGAN-GP $C_L = 0.1$, calculations on most shapes converged. At $C_L = 0.5, 1.0, 1.4$, the XFoil calculations on most shapes obtained from both CGAN and CWGAN-GP converged.

The rate of smooth shapes, MSE, and μ are plotted against the labels in Figure 7. The rate of smooth shapes indicates the shapes whose XFoil computation converges. The indices are calculated for each C_L. The rate of smooth shapes was higher in CWGAN-GP than in CGAN, except at $C_L > 1.3$. The MSE of both the models had a similar tendency: it increased as the label increased. At $C_L < 0.4$, the MSE of CGAN was smaller than that at $C_L > 0.4$. In this region, the rate of smooth shapes was small. It implies that there were few smooth shapes among the CGAN outputs, but the shapes satisfied the label with high accuracy. The μ of CWGAN-GP was higher than that of CGAN in all the labels. This is consistent with the fact that WGAN prevents mode collapse.

5 Conclusion

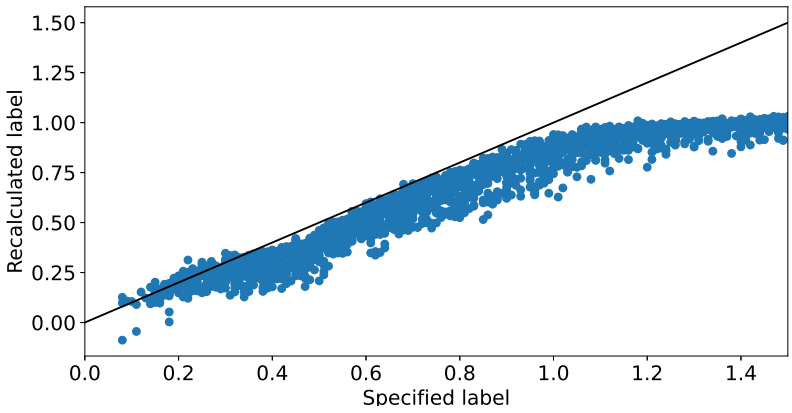
Herein, we proposed a CWGAN-GP model for generating airfoil data to overcome the issue of nonsmooth shapes in ordinal GAN models. The shapes generated using the CWGAN-GP model were as smooth as those obtained using GAN coupled with a smoothing technique. More variation existed in the shapes generated using the CWGAN-GP model than in those using the CGAN model. The difference between CGAN and CWGAN-GP was because WGAN-GP was trained stably, and it avoided mode collapse. It is beneficial that neither smoothing methods nor free curve representations are required to generate shapes, and it widens the applications of the shape generation method.

Acknowledgement

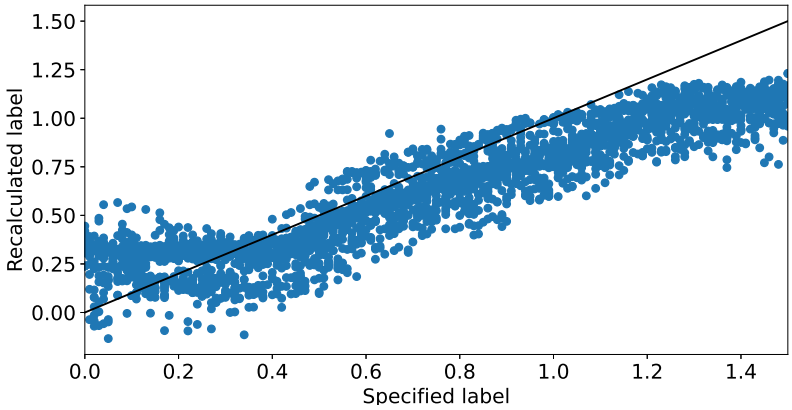
This work was partially supported by JSPS KAKENHI Grant Number 21K14064.

Replication of result

The source code and data are available on github (https://github.com/miyamotononno/generate_airfoil).

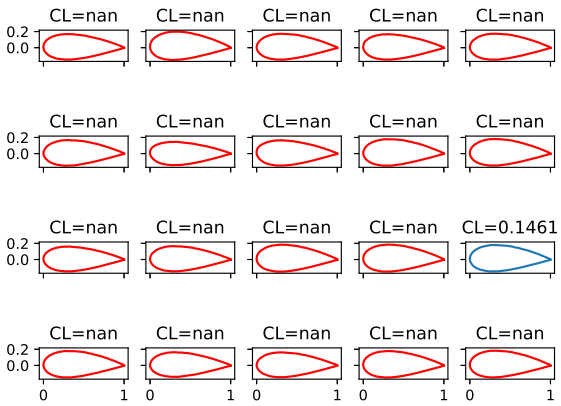


(a) Conditional GAN ($d = 3$).

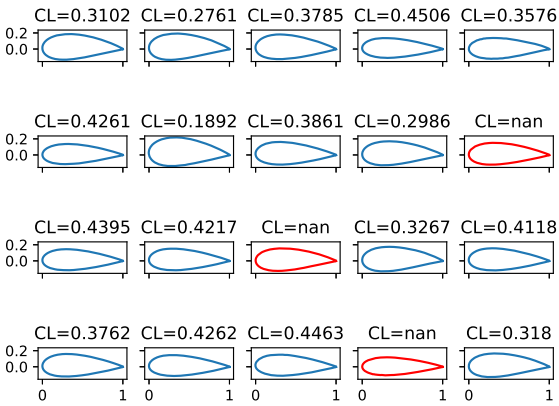


(b) Conditional WGAN-GP ($d = 3$).

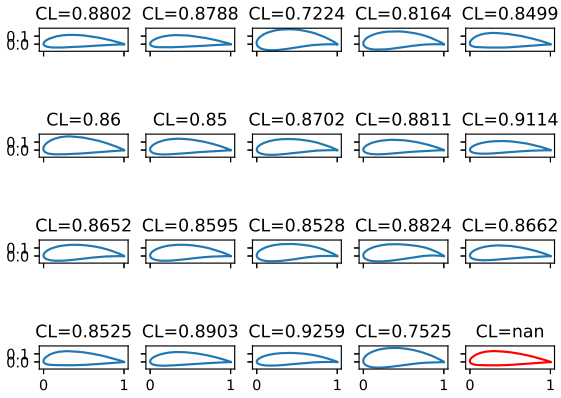
Figure 4. Error distribution.



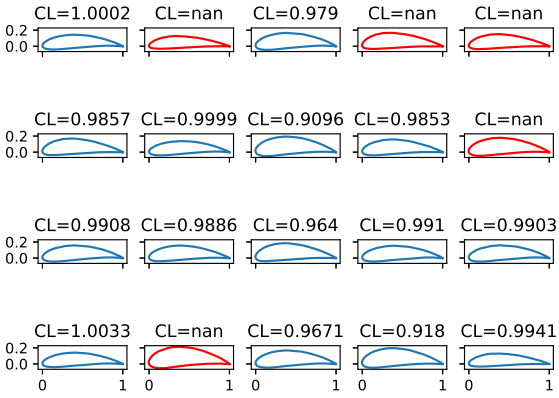
(a) CGAN, $C_L = 0.1$.



(b) CGAN, $C_L = 0.5$.

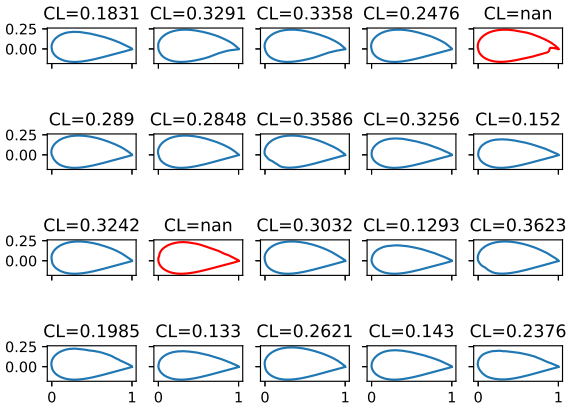


(c) CGAN, $C_L = 1.0$.

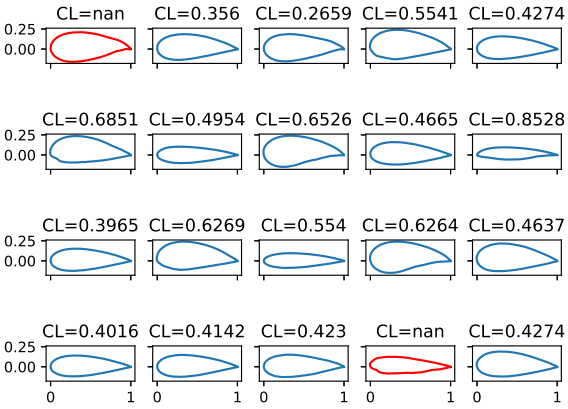


(d) CGAN, $C_L = 1.4$.

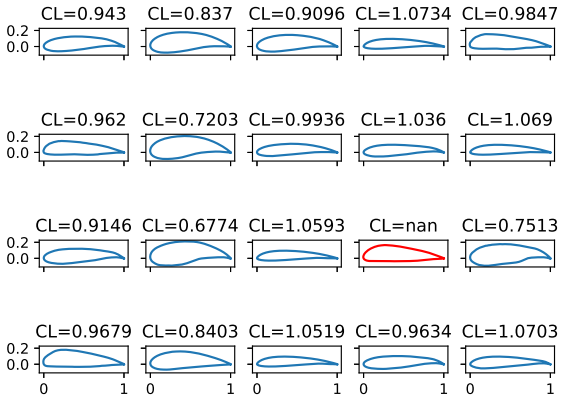
Figure 5. Generated shapes of CGAN ($d = 3$).



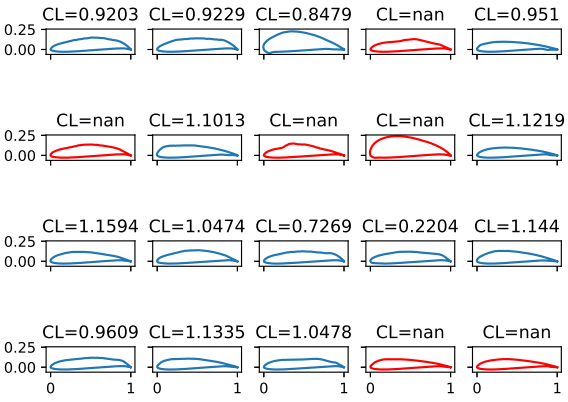
(a) CWGAN-GP, $C_L = 0.1$.



(b) CWGAN-GP, $C_L = 0.5$.

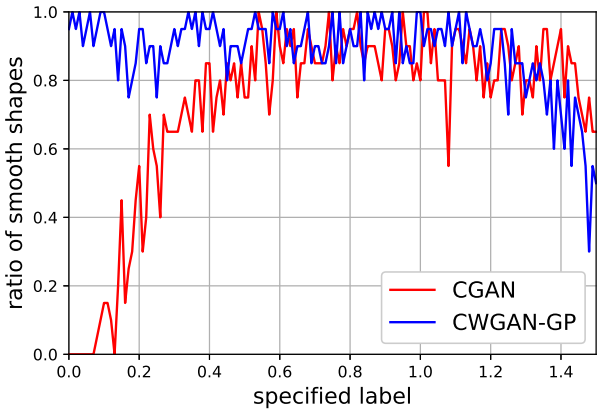


(c) CWGAN-GP, $C_L = 1.0$.

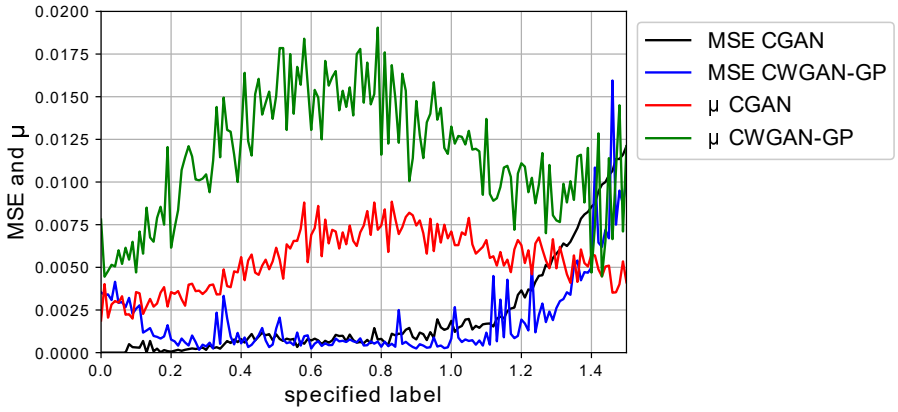


(d) CWGAN-GP, $C_L = 1.4$.

Figure 6. Generated shapes of CWGAN-GP ($d = 3$).



(a) Ratio of smooth shapes.



(b) MSE and μ.

Figure 7. Comparison between CGAN and CWGAN-GP.

References

1. Jameson, A. *Optimum aerodynamic design using CFD and control theory*.
2. Shelton, M. L. *et al.* Optimization of a transonic turbine airfoil using artificial intelligence, cfd and cascade testing. **3A: General** (1993).
3. Wang, Y., Shimada, K. & Farimani, A. B. Airfoil GAN: Encoding and synthesizing airfoils for aerodynamic-aware shape optimization. *arXiv* (2021). 2101.04757.
4. Chen, W., Chiu, K. & Fuge, M. BézierGAN: Automatic generation of smooth curves from interpretable low-dimensional parameters. *arXiv* (2021). 1808.08871v2.
5. Achour, G., Sung, W. J., Pinon-Fischer, O. J. & Mavris, D. N. *Development of a Conditional Generative Adversarial Network for Airfoil Shape Optimization*, 2261.
6. Press, W. H. & Teukolsky, S. A. Savitzky-Golay smoothing filters. *Comput. Phys.* **4**, 669–672 (1990).
7. Yonekura, K. & Suzuki, K. Data-driven design exploration method using conditional variational autoencoder for airfoil design. *Struct. Multidiscip. Optim.* in press (2021).
8. Yonekura, K., Wada, K. & Suzuki, K. Generating various airfoil shapes with required lift coefficient using conditional variational autoencoders —Novel shapes by combining NACA and Joukowski airfoils—. *Eng. Appl. Artif. Intell.* under review (2021).
9. Larsen, A. B. L., Sønderby, S. K., Larochelle, H. & Winther, O. Autoencoding beyond pixels using a learned similarity metric. In Balcan, M. F. & Weinberger, K. Q. (eds.) *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48 of *Proceedings of Machine Learning Research*, 1558–1566 (PMLR, New York, New York, USA, 2016).
10. Gaggero, S., Vernengo, G., Villa, D. & Bonfiglio, L. A reduced order approach for optimal design of efficient marine propellers. *Ships Offshore Struct.* **0**, 1–15 (2019).
11. Yonekura, K. & Watanabe, O. A shape parameterization method using principal component analysis in application to shape optimization. *J. Mech. Des.* **136**, 121401 (2014).
12. Brown, N. C. & Mueller, C. T. Design variable analysis and generation for performance-based parametric modeling in architecture. *Int. J. Archit. Comput.* **17**, 36–52 (2019).
13. Oh, S., Jung, Y., Kim, S., Lee, I. & Kang, N. Deep generative design: Integration of topology optimization and generative models. *J. Mech. Des.* **141** (2019).
14. Bidgoli, A. & Veloso, P. Deepcloud. the application of a data-driven, generative model in design. *arXiv* (2019). 1904.01083.
15. Nash, C. & Williams, C. K. I. The shape variational autoencoder: A deep generative model of part-segmented 3d objects. *Comput. Graph. Forum* **36**, 1–12 (2017).
16. Goodfellow, I. Nips 2016 tutorial: Generative adversarial networks (2017). 1701.00160.
17. Arjovsky, M. & Bottou, L. Towards principled methods for training generative adversarial networks (2017).
18. Arjovsky, M., Chintala, S. & Bottou, L. Wasserstein generative adversarial networks (2017).
19. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. Improved training of Wasserstein GANs (2017).
20. Abbot, I. H., von Doenhoff, A. E. & Stivers, L., Jr. *Summary of Airfoil Data* (United States, 1945).
21. Drela, M. Xfoil: An analysis and design system for low Reynolds number airfoils. In T.J., M. (ed.) *Low Reynolds Number Aerodynamics*, vol. 54 of *Lecture Notes in Engineering*, 1–12 (Springer, Berlin, Heidelberg, 1989).
22. Villani, C. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften (Springer, 2009).
23. Paszke, A. *et al.* Automatic differentiation in pytorch. (2017).