Scalable Parameterization of Aerodynamic Shapes using Deep-Learning-Based Geometric Compression

Tom Bamford *, Andy Keane†, and David Toal‡
University of Southampton, Southampton, SO16 7QF, UK

The design of aerodynamic parts requires a mathematical representation of the geometric shape, through which systematic modifications can be made in an iterative design procedure. Whilst numerous parameterization formulations are available for airfoil design in the 2D setting, for the three-dimensional case a constructive parameterization tool that is applicable across a wide range of problem settings is yet to be put forward. In this paper a scalable parameterization of aerodynamic designs is created by applying deep learning for compression of individual geometries and leveraging meta-learning to fit the corresponding distribution. Compression is achieved by reformulating the shapes descriptor from discrete grids to continuous neural representations of the implicit geometric field data; the corresponding variable number can be reduced by an order of magnitude or more, reducing the associated design space and parameterization complexity. In this work, the approach has been applied to both two and three-dimensional problem settings through airfoil and wing parameterization, and benchmarked against alternative deep learning parameterization approaches as well as the discrete field alternative. Results show the method is competitive - and in many cases outperforms - the benchmarks, whilst achieving strong scalability with regards to both geometric field resolution and spatial dimensionality.

Nomenclature

GAN	=	Generative Adversarial Network
VAE	=	Variational Auto-Encoder
AAE	=	Adversarial Auto-Encoder
INR	=	Implicit Neural Representation
MMD	=	Maximum Mean Discrepancy
$\mathbf{z}$	=	Latent Variables

*Post-doctoral Researcher, Rolls-Royce University Technology Centre
†Professor of Computational Engineering
‡Associate Professor

ϕ = Geometric Field Value

$\hat{\phi}$ = Predicted Field Value

f = Fitted Function

$\mathbf{x}$ = Input Variable

$\mathcal{L}$ = Loss Function

Subscripts

G = Generator/Decoder

E = Encoder

D = Discriminator

I. Introduction

The design of complex engineering parts continues to be a challenging undertaking. A given design must meet multiple competing design objectives from different engineering disciplines simultaneously, whilst satisfying a range of constraints potentially numbering in the hundreds (see e.g. [1]). For example, in the design of aerodynamic parts the aerodynamic goal aims to reduce drag and weight of a given part, whilst the structural design team aim to increase rigidity and structural integrity such that it can withstand high loads. To satisfy such a diverse set of requirements, an iterative optimization process is typically carried out over the space of designs. To enable the systematic modification of a given design based on current performance, a representation of the part is required through a mathematical parameterization.

Parameterization approaches can be categorized into two families: constructive and deformative. Constructive approaches parameterize a shape directly, essentially through a curve-fitting functionality, whereas deformative approaches aim to define (and subsequently deform) a shape relative to some pre-defined benchmark. For aerodynamic design the focus of the parameterization is often the airfoil, for which constructive approaches include NACA, B-splines and CST [2, 3], whilst deformative methods include Hicks-Henne and SVD [4, 5]. In the three-dimensional setting, geometries such as wings and blades will often be parameterized either by FFD [6] in the deformative case, or multiple airfoil curves stitched together by splines in the constructive case (e.g. [7–9]). The requirement for high quality 2D airfoil parameterizations is thus motivated significantly by their ongoing use in the three-dimensional setting.

In the last decade, the advancement in artificial intelligence and machine learning has led to the development of more modern, data-driven approaches to shape parameterization. The generative capability of GANs and VAEs in particular (introduced by [10] and [11] respectively) has been utilised by researchers as a novel approach to shape parameterization, with their potential for dimensionality reduction, compatibility with the three-dimensional case and applicability to inverse design cited as key advantages [12–14].

Despite the inherent flexibility of such a data-driven approach, the requirement for a part-agnostic method that

is scalable to three-dimensional designs still presents a challenge. In particular, to ensure sufficiently smooth output designs for aerodynamic applications, many researchers have leveraged final-layer transformation using traditional parameterization curve constraints (e.g. [12, 15, 16]). Other researchers have used surface coordinate outputs to represent the shape of interest [17, 18], which whilst flexible lacks spatial connectivity. In addition both approaches lack a surrounding domain over which to overlay flow field data for application to inverse design and low-fidelity CFD modelling. A more recent approach to three-dimensional shape parameterization was put forward by [19], in which a neural network is trained to represent mesh vertex positions for a single geometry, with the entire set of 150k+ parameters re-optimised using an adjoint solver to modify the geometric shape. Whilst the results were very promising, the requirement for a fixed vertex number in addition to the sheer number of parameters involved pose questions as to the scalability of this method.

In this work an image-based approach to aerodynamic shape parameterization is proposed, through the leveraging of a generative Adversarial Auto-Encoder (AAE, [20]) model to learn distributions of geometries represented implictly through discrete (grid-based) field data. In contrast to alternative approaches, this method allows direct learning on structured geometric data typically gathered through scans of pre-existing parts. To ensure viability, it is first shown that such a method is able to output sufficiently smooth geometries directly without any post-processing steps using two-dimensional airfoil data. Next, the challenge of scaling such an approach to higher resolutions and three-dimensional settings is tackled. Implicit Neural Representations (INRs, [21–23]) are first leveraged to compress geometric field data via a neural network surrogate, alongside which meta-learning [24] is utilised to train a generative model able to represent the space of geometric shapes. The approach is implemented both in the two-dimensional case for airfoil design, and for three-dimensional wing design.

In the next section, the methodology is presented in more detail, following which the two and three-dimensional results are given in Sections III and IV respectively. All proposed models are benchmarked, and in the 3D case a wing design optimisation is implemented for each parameterization. Finally in Section V the contributions of this work are summarized and key achievements highlighted. Note that some of the two-dimensional results have previously been published in [25] and [26]; this work builds on these studies by unifying the narrative and more significantly through extension to the three-dimensional case and application to wing optimization.

II. Methodology

A. Aerodynamic Shape Generation via Implicit Field Learning

The aim of this work is to implement model training on a dataset of geometric images so as to learn a scalable part-agnostic parameterization. In such a context, an image refers to any grid-based representation over which structured learning can take place. Defining a suitable field as one in which geometric surfaces can be represented through field

contours, two clear candidates emerge. The first is the occupancy map, in which a binary field value is used to classify points as internal or external with respect to the geometric boundary. Whilst this approach is effective, the relative lack of information leads to a cost in accuracy for a given resolution. The second approach, the Signed Distance Field (SDF), improves upon this by augmenting the binary internal/external classification with a measure of distance: each cell is assigned the value of the distance to its closest point on the geometric surface, with the sign of the value used to differentiate between internal and external points (here negative is taken to be internal). This implicit field approach, which has gained popularity in applications as varied as collision detection, medical imaging, CAD and aerodynamic design ([27–30]), is the one used here. To convert the field back into a set of surface points, the marching squares algorithm is used (marching cubes in the three-dimensional case). This algorithm defines a second grid overlaying in the first, in which the corners of the second grid are aligned with the centres of the original. Since the surface is found at the zero-contour, the cells of the second grid containing the surface can be identified as those in which the four corner values differ in sign. Linear interpolation between distance values is then carried out to approximate the precise surface location; given a higher grid resolution, the accuracy of this piecewise linear approximation will be increased and thus the corresponding reconstruction accuracy.

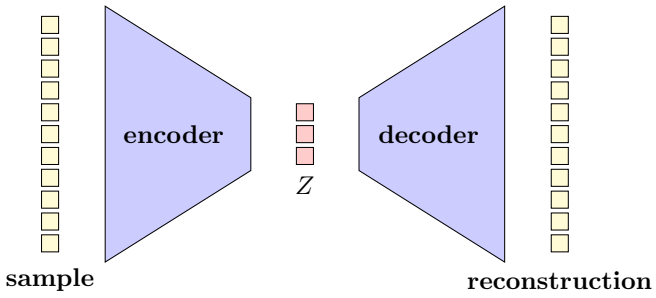


Fig. 1 Auto-Encoder, which forms the backbone as the generative VAE.

Given a suitable image representation to train on, the next step is the choice of appropriate generative deep learning model. In this work the framework selected is the Adversarial Auto-Encoder (AAE), introduced by [20]. This model is a hybrid between the two most well established generative frameworks over the last decode or so: the VAE and the GAN.

Variational Auto-Encoders (VAEs) are built upon the auto-encoder architecture shown in Figure 1. These models consist of two separate neural networks acting together to learn a compressed representation of the input training data. The first network, the encoder, takes the input data and transforms it into a low-dimensional vector (latent) representation, after which the decoder network aims to transform it back into the original format through reconstruction. Thus the model is associated with a characteristic bottleneck structure. The objective during training is simply to minimize the loss in the reconstruction. Whilst each sample in the training set can be mapped to a specific position vector in the learnt representation space, the intermediate regions in this space are typically noisy and lacking structure. [11]

proposed the VAE as a modification to the architecture through the encoder, which rather then outputting a compressed representation directly, outputs statistical values (mean and standard deviation) corresponding to an assumed Gaussian distribution of each variable. This introduces stochasticity in the positions of reconstructed samples, which helps to smooth out the space. Since novel, higher quality samples can be drawn from randomly sampling positions in the space according to the learnt distribution, this modification transforms the auto-encoding architecture into a generative framework. Mathematically, the VAE objective is to minimize the loss,

$$\mathcal{L} = \mathbb{E}_{z\sim P_E(\boldsymbol{z}|\boldsymbol{x})}[\log(P_G(\boldsymbol{x}|\boldsymbol{z}))] - \mathrm{Div}_{KL}(P_E(\boldsymbol{z}|\boldsymbol{x})||P(\boldsymbol{z}))\ , \tag{1}$$

where $\boldsymbol{z}$ is the latent representation, $P_E(\boldsymbol{z}|\boldsymbol{x})$ and $P_G(\boldsymbol{x}|\boldsymbol{z})$ the encoder and decoder networks respectively, and $P(\boldsymbol{z})$ is the prior distribution from which latent values are drawn. Note as such that the first term acts as the reconstruction loss, whilst the second is the Kullback-Liebler (KL) divergence between prior and learnt distributions. To ensure that gradients can be calculated back through the model for parameter updates, a re-parameterization trick is used; rather than treating the latent vector as a random variable (for which gradients cannot be calculated), it is re-written such that the stochasticity is provided by a newly defined, independent variable. In the context of aerodynamic design, these models have primarily been used in the inverse design case (e.g. [31–33]).

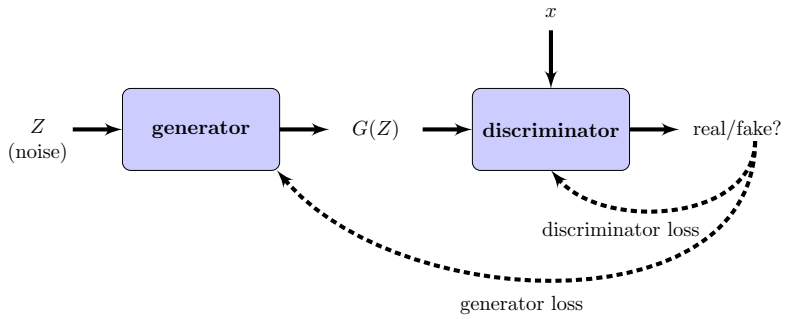


Fig. 2 The GAN consists of two networks working adversarially in a two-player minimax game.

Generative Adversarial Networks (GANs), depicted in Figure 2, were introduced at a similar time by [10] as an alternative approach to generative learning. In contrast to VAEs, these models have no reconstruction capability to guide the training process; instead a discriminative network is used to classify output samples and thus guide the model. This is implemented through an adversarial training procedure, in which the two constituent networks work against each other in a two-player game. The first, generative, network G converts an input vector representation (randomly drawn from a chosen prior distribution) into a data sample consistent with those from the training set. The second, discriminative, network D takes as input both this sample alongside those drawn from the training set itself, and aims to correctly classify the source of the sample. The generator is fed the classification during the training process and uses this to guide the model updates, aiming to fool the discriminator into incorrectly classifying generated samples as real. On the

other hand, the discriminator aims to maximize its classification accuracy. Mathematically, the objective is given by,

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim P_{\text{data}}(\boldsymbol{x})}[\log(D(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{z}\sim P_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \,, \tag{2}$$

where the notation highlights the competitive nature of the optimization through the adversarial minimization-maximization aims of the generator and discriminator respectively. GANs have been shown to be very effective at producing high quality synthetic samples across a range of applications (see [34]), and are thus often preferred to VAEs. In practice however, the adversarial nature of the training process is inherently unstable and thus difficult to implement. The lack of reconstruction capability can also make training more challenging, in addition to diminishing the interpretability of the model. Despite these considerations, GANs have already been widely used within the aerodynamic design research community, including for direct generation of airfoils through coordinate represenation [35], as well as for dimensionality reduction of traditional curve-based parameterizations using final layer mappings [12, 15]. A similar approach has also been utilised in the three-dimensional case with FFD [13].

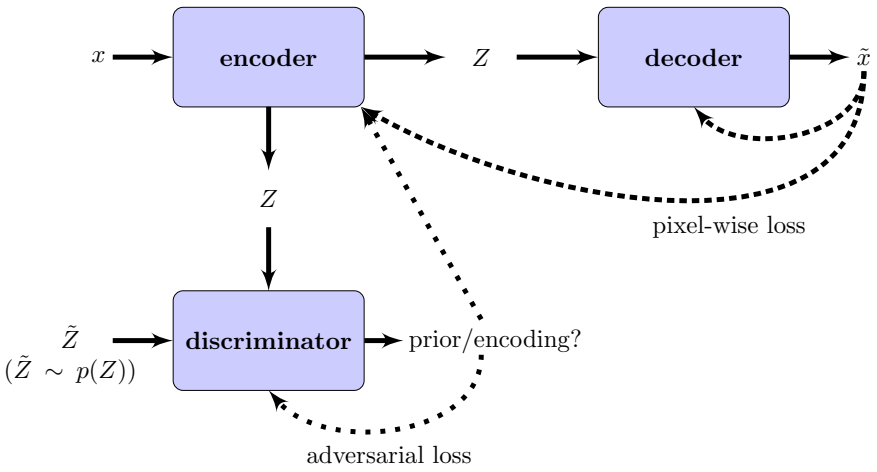


Fig. 3 Schematic of the AAE, in which three networks are used to provide a hybrid auto-encoding GAN functionality.

The difficulties mentioned above have motivated the development of numerous VAE-GAN hybrids aiming to combine the advantages of both [36–40]. The AAE used in this work is one such approach, which consists of three networks collectively allowing both adversarial training and reconstruction capability (see Figure 3). Similar to the VAE, the AAE includes auto-encoding functionality through an encoder-decoder pair working in unison to learn compact representations of the data space. The encoder outputs statistical parameters which are used to construct the latent distribution for each sample and aid smoothness in the learned space. Rather than utilisation of the KL divergence to measure the quality of the learned data distribution however, the AAE uses a third, adversarial network inspired by the GAN. In this framework the discriminator works against the encoder, attempting to distinguish between the encoded

latent codes and those drawn directly from a Gaussian prior distribution. This drives the latent space to a smooth, Gaussian structure, aiding sample quality in intermediate regions of the space.

The vanilla AAE is used as the backbone of the two-dimensional grid-based airfoil parameterization named SDF-GAN. The encoder-decoder pair are trained to input/output uniform SDF grids of fixed resolution, whilst the discriminator as usual takes as input the latent embedding vector and outputs a classification. Both architecture (e.g. number of layers, feature number, activation function) and training (e.g. optimizer, learning rate) hyper-parameters are varied systematically using one-at-a-time sampling and LP-τ DOE to determine the most effective configuration of the most impactful parameters. The final model consists exclusively of fully-connected (i.e. no convolutional) hidden layers, with 3, 5 and 3 each for the encoder, decoder and discriminator respectively. The model is trained using distributed learning for 12 hours on a 40-CPU node, with offline-evaluation taking place every two hours to assess the optimal configuration. Three metrics were used for hyper-parameter tuning: feasibility, innovativeness and reconstruction error. Feasibility measures the fraction of output geometries that converge in the CFD solver [41, 42], with infeasible geometries either lacking sufficient smoothness or structural similarity to current airfoil designs, or unable to be extracted from the SDF due to a lack of contour. Innovativeness measures the novelty of new geometries, calculated as the maximum value of enclosed area between synthetic airfoils and their nearest neighbour in the training set. Finally, reconstruction error is simply the pixelwise $L1$ value across the input/output SDF grids. All metrics are evaluated on 1000 generated airfoils. Note that a post-process smoothing hyper-parameter was also studied during the hyper-parameter tuning, in which XFOIL reverse engineers the design after applying smoothing through a Hanning filter on the resulting aerodynamic profile. This functionality was turned off after it was seen to be unnecessary, with the direct output geometries often being sufficiently smooth for XFOIL convergence. The XFOIL solver was run with a Reynolds number of 6e+06, a Mach number of 0.15 and zero angle-of-attack.

B. Hypernetwork-based Implicit Neural Field Generation of Aerodynamic Shapes

SDF-GAN acts as an image-based implementation of aerodynamic shape parameterization, learning directly on discrete field data to capture the underlying geometric distribution. The simplicity of the basic concept renders this the 'brute-force' approach to deep-learning based image parameterizations. Whilst effective as a solution, this comes with the inherent limitations in scalability associated with learning on discrete grids, which suffer from the curse-of-dimensionality. As the resolution or spatial dimensionality is increased, the costs associated with learning on these data structures become prohibitive, particularly with regards to computational memory (RAM). Even for convolutional architectures [43], which leverage kernel-based weight sharing across cells to reduce overhead, the requirement for the CNNs to take as input the entire grid structure as a collective soon renders these models computationally impractical for sufficiently high resolutions.

In this section a novel generative approach for scalable field learning is introduced. The method builds upon the

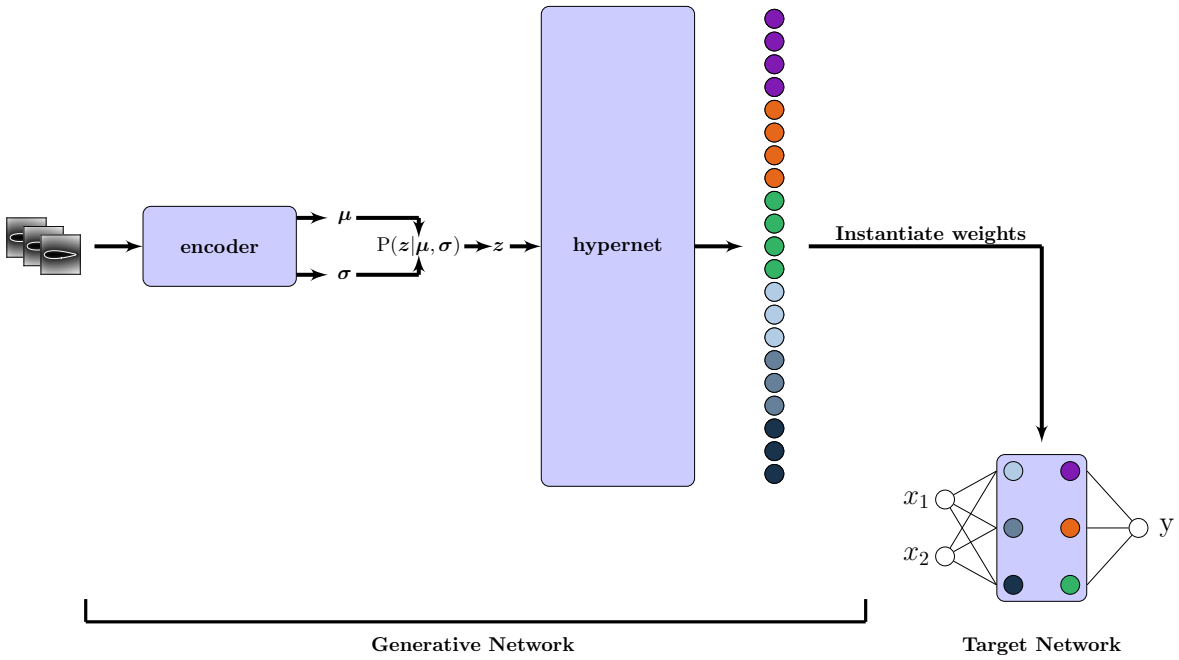


Fig. 4 AeroINR-2D. The decoder acts as a hypernetwork through which the meta-learning procedure is implemented.

development in recent years of Implicit Neural Representations (INRs) and related tools [21–24, 44–46]. INRs are trained as neural network surrogates to raw discrete field data, taking as input a coordinate position and outputting corresponding field value(s). They can be trained in aggregation to encode a dataset of field data in a single model (e.g. DeepSDF, [22]), or as distinct neural representations for each data sample (identified as functa by [24]) where each network effectively acts as an overfit model to the corresponding raw field data [21]. This transforms the representation of geometry from image to network weights, which can reduce the parameter number by an order of magnitude or more, enhancing scalability. In addition, fields are converted from discrete data to a continuous representation, allowing sampling at arbitrary non-uniform points both in training and inference. In addition, there is no constraint for sampled coordinates to have been seen within the training set. INRs are also field-agnostic, so can equally be trained on geometric fields such as occupancy fields and SDFs (e.g. [22, 23]) or physics data such as pressure fields ([47]).

In an aerodynamic context, INR-based methods have been used by Duvall et al. [48] for flow field prediction around 2D airfoils, as well as by Pan et al. [49] for three-dimensional turbulence prediction on arbitrary meshes. They have also been used for supercritial airfoil prediction using an implicit decoder network component by Li et al. [50].

This work introduces AeroINR, which leverages generative learning to learn the geometric data parameterization as a distribution of INR weights. This is implemented through meta-learning (i.e. training network weights to learn the distribution of downstream INR weights) in a fully self-contained training algorithm, such that the fit of individual surrogate weights and the learning of the overall weight distribution occur simultaneously (Figure 4). As with SDF-GAN,

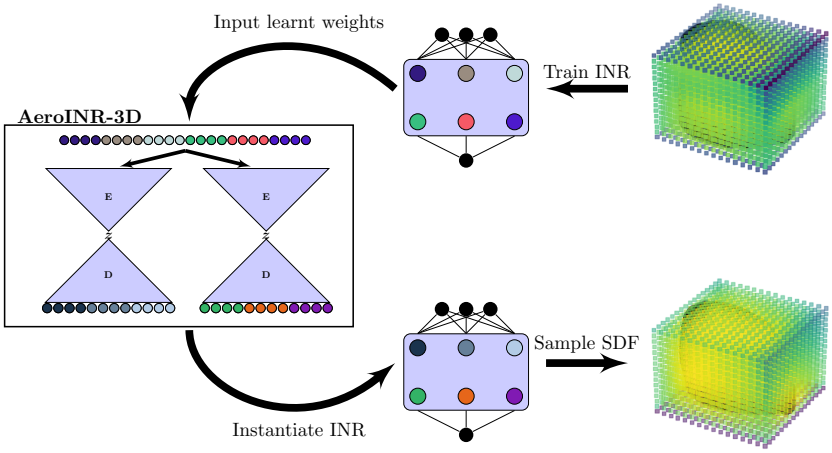


Fig. 5 Wing processing with AeroINR-3D. Each SDF blade represented is first converted to a set of INR weights through overfitting, with the (layer-wise) auto-encoding functionality of AeroINR-3D acting on the learned weights through a meta-learning procedure.

the AAE forms the backbone of the generative model, with the decoder here acting as a hypernetwork [51] which outputs weights used for downstream INR instantiation. The instantiated INR is then sampled at the coordinates corresponding to the discrete grid cell positions, and the output SDF value predictions compared to the raw grid values to compute the reconstruction loss and update (AAE) model weights. The reconstruction loss is thus given by

$$\mathcal{L}_{\text{rec}} = \frac{1}{N}\sum_{i=1}^{N} |\phi_i - \widehat{\phi}_i| \,, \tag{3}$$

where the reconstructed grid is formed from sampling of the instantiated INR through $\widehat{\phi}_j = f_{w^{k,j}}(x_j)$ and ϕ_i are the raw SDF values. Note that the INR architecture is thus fixed across the dataset, but learned internal weight parameters vary.

AeroINR was trained on the same 2D airfoil images as SDF-GAN at 64×64 resolution. Distributed training is once again carried out, but in this case due to time limitations the training was stopped based on loss convergence to within a 5% tolerance over 1000 running epochs. Most model hyper-parameters for the AAE are shared with those of SDF-GAN, with the exception of the decoder which here has only one hidden layer. As such, both encoder and discriminator networks have 3 hidden layers, with 32, 64 and 128 features in the first, second and third hidden layer respectively, whilst the decoder has only the 32 feature first hidden layer. A 3-layer fully-connected network is used for the INR, again with 32, 64 and 128 features respectively. The three metrics used for SDF-GAN were again used for model evaluation.

C. High Resolution 3D Aerodynamic Shape Learning of Independent Weight Distributions

For an effective 3D parameterization, scalable design is essential. Whilst the scalability of AeroINR relative to SDF-GAN is much improved, the dependency on the geometric grids and their cell number remains. In particular, the encoder network must still take in all field values in its first layer, limiting the compressibility of the model (although this can be mitigated by an auto-decoding functionality as shown in Section III). In addition, point-value pairs are used during training to compute the loss and drive parameter updates. Whilst they are processed independently at the INR output, for a grid size of a few 10s of thousand cells the RAM is soon overloaded and each geometry must be further split across multiple batches, increasing training overhead.

To circumvent this, the 3D implementation - AeroINR-3D - completely disentangles INR surrogate weights from the underlying geometric field values. INR surrogates are first fit to geometric field samples from the raw dataset as a distinct pre-processing step (see Section IV.A for details). The INR weight distribution is then treated directly as the generative learning objective for the 3D parameterization model. That is to say, the 3D model aims to learn the distribution of INR weights directly, without any knowledge of the underlying geometric descriptor. Output INR weights can subsequently be used to instantiate the downstream INR during post-processing, from which geometric field values are sampled and the geometry reconstructed. The model setup is shown in Figure 5. The encoder-decoder networks of the underlying AAE framework process the INR weights directly as input-output parameters, with INR sampling a distinct step. As with AeroINR, the decoder here acts as a hypernetwork, outputting downstream learned model parameters. To improve performance, a layer-wise hypernetwork model is utilized, in which each layer of the INR is processed through a separate hypernetwork (e.g. [52, 53]); note that the encoder is however shared by all layers, taking as input the full vector of weights. This simplifies the learning problem for each hypernetwork, and allows easy scalability of INR parameter number.

A 4-layer, 256 feature per layer network was used for the INRs after some manual hyper-parameter tuning, with Leaky ReLU activations except for a final layer tanh function (field values are normalized to a magnitude less than or equal to 1). The configuration is consistent with the findings of [21] for independent, overfit INRs. The loss function takes the form,

$$\mathcal{L}_{\text{AAE}} = \mathcal{L}_{\text{rec}} + \gamma \cdot \mathcal{L}_{\text{adv}} \,, \tag{4}$$

where γ sets the weighting between reconstruction term,

$$\mathcal{L}_{\text{rec}} = \frac{1}{N} \sum_{i=1}^{N} |\mathbf{w}_i - \hat{\mathbf{w}}_i| \,, \tag{5}$$

and adversarial terms,

$$\mathcal{L}_{\text{adv}} = \min_{E} \max_{D} \mathbf{E}_{z\sim P_z(z)}[\log D(z)] + \mathbf{E}_{w\sim P_{\text{weights}}(w)}[\log(1 - D(E(w)))] \ . \tag{6}$$

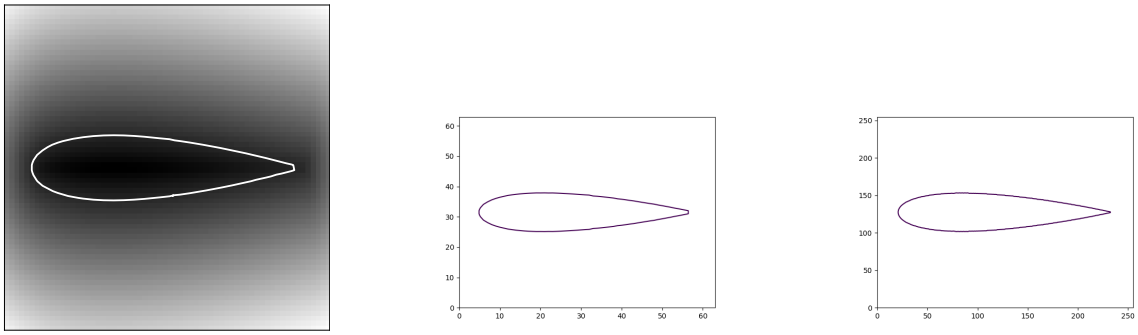
Here $\mathbf{w}$ is the vector of weights formed by concatenating the weights of each INR layer, and E the encoder network. An LP-τ DOE was implemented across 100 AeroINR-3D model configurations, treating both generation and reconstruction as independent objectives (such that the DOE is carried out twice). Distributed training is implemented across 64 CPUs, which allows parallel off-line evaluation at periodic hourly intervals, with training carried out for three hours in total for all configurations. The final generative model consists of three encoder layers, a single hidden layer decoder (hypernet) and a 4-dimensional latent embedding, whilst the reconstruction model consists of two decoder layers and a latent dimensionality of 64.

III. Airfoil Synthesis and Reconstruction

A. Dataset

SDF grids are created from UIUC airfoil data and artificially created airfoils used to supplement the dataset. The UIUC database consists of coordinate data for over 1500 real airfoils [54]. Standardization is carried out over the dataset to ensure both number and position of sampled coordinates is consistent across all airfoils. This is done by first fitting raw coordinate data to a 25-parameter CST parameterization [3], which has been shown to provide an accurate shape representation across the vast majority of UIUC airfoils [55]. Airfoils with very few coordinates in the raw form often lead to unrealistic curve fits and are subsequently removed after a visual check. This reduced the number of UIUC airfoils to around 1200 geometries, with each airfoil sampled 500 times across both upper and lower curves using a cosine distribution. The average absolute residual error across the dataset is 4.4e-05 evaluated on test points. For synthetic airfoil creation, a 25 base vector singular value decomposition is carried out over the UIUC set, and a weighted expansion of the corresponding base vectors is used to fit new airfoils. Coefficients are determined by constructing a Gaussian distribution for each coefficient value from the original UIUC set, with the standard deviation set to a third of the coefficient range. This is preferred to direct sampling of the CST parameters due to the very low likelihood of generating realistic airfoil shapes from this parameter space.

Uniform SDF grids are created using a fixed domain size chosen to cover all dataset airfoils, specifically from [-0.1,1.1] in the chordwise dimension and [-0.3, 0.3] in the thickness dimension. Grids are created at three resolutions, 64×64, 128×128 and 256×256; the former was found sufficient for a high quality representation of airfoils, and thus was used for final model results. Note however that the lower resolution acts to smooth the trailing edge (see Figure 6), so for some applications a slightly higher resolution or non-uniform sampling may be required. Training is carried out on a dataset of 1000 UIUC airfoils, with the remainder placed into the validation set, which itself consists of another 1000 airfoils (supplemented by the synthetic set).



(a) Raw SDF data (64 x 64) **(b) 64 x 64** **(c) 256 x 256**

Fig. 6 SDF grid reconstruction of the NACA 0012 airfoil. (a) shows the raw SDF grid data as a 64 x 64 image, where the data has been rescaled for display; (b) shows the airfoil boundary reconstruction from raw SDF data, using marching squares; (c) same as b, for the highest 256 x 256 resolution.

Table 2 Metric comparison between AeroINR and alternative, state-of-the-art approaches.

Model	MMD^2	Latent Space Consistency	Relative Diversity
Bezier-GAN	0.1856 ± 0.0007	0.9432 ± 0.0025	1.0455 ± 0.0144
Entropic Bezier-GAN	**0.0896 ± 0.0015**	0.9878 ± 0.0005	1.3264 ± 0.0196
SDF-GAN	0.3224 ± 0.0219	0.9640 ± 0.0030	2.5396 ± 0.2908
AeroINR	0.3494 ± 0.0151	**0.9996 ± 0.0001**	**2.6410 ± 0.1659**

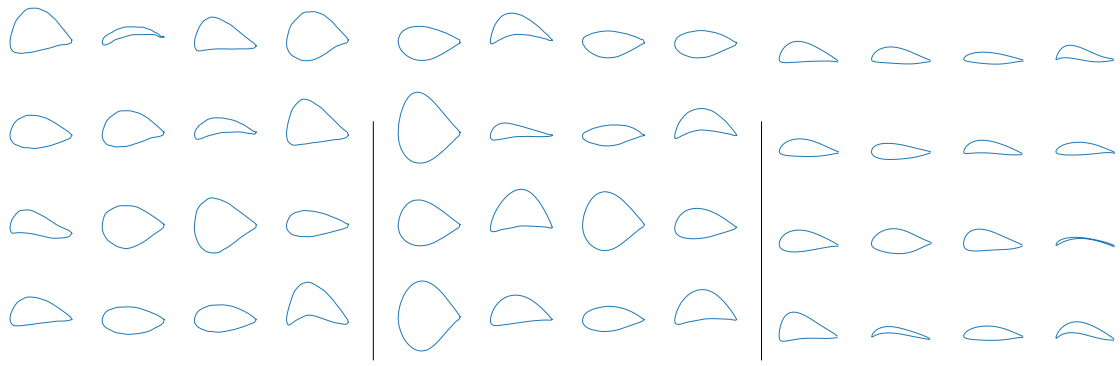
B. Benchmarks

Bezier-GAN Introduced in 2019 as one of the first applications of deep learning to airfoil shape parameterization [12]. An InfoGAN framework [56] is utilized to allow generation of shapes with some interpretability in the learnt parameter space. In particular, this framework trains two levels of generator input encoding, such that the higher level variables are interpreted as control (latent) variables of the lower frequency shape components, whilst the lower level (noise) variables act to introduce small-scale variability into the specified design. To ensure smoothness in the output shape, the final layer transforms input features into a parameterized Bezier curve, which is then sampled to output the corresponding airfoil coordinate representation.

Entropic Bezier-GAN To mitigate the instability and long training times of the GAN, an entropic variant of Bezier-GAN was introduced [57]. In this implementation, the adversarial loss is replaced with a computational optimal transport equivalent, which acts to minimise the Sinkhorn divergence. Such a formulation allows the synthetic and training data distributions to be compared without the use of a discriminative network.

C. Parameterization Results

Table 2 shows a quantitative comparison of SDF-GAN and AeroINR against the two state-of-the-art benchmark approaches in parameterization capability. In particular, following [58] the learned parameterizations have been



(a) AeroINR **(b) SDF-GAN** **(c) Bezier-GAN**

Fig. 7 Visualisation of diversity of different parameterizations through random sampling of the design space.

compared through three metrics: Maximum Mean Discrepancy (MMD), latent space consistency and relative diversity. MMD is standard generative learning measure used to evaluate the difference between the training or target distribution, and that output by the learned model. Latent space consistency aims to quantify the structure of the latent space, by evaluating the correlation between distances in this space and the distances in the geometry space. Finally, relative diversity aims to capture the ability of the parameterization to output novel designs, by measuring the variance of output geometries relative to the variance of the original training geometries. Note that MMD and relative diversity are evaluated for the aerodynamically feasible outputs only.

AeroINR can be seen to outperform the alternatives in two out of the three metrics - latent space consistency and relative diversity - whilst SDF-GAN is also very competitive on these. The performance on MMD however is less promising, with both approaches somewhat inferior to the benchmarks. In reality, this shouldn't be too surprising given the stronger constraint on the output shape in the Bezier-GAN models due the final layer transformation. It is also likely that the superior performance in diversity is directly related to the MMD score, with these metrics working against each other to some extent - more novel designs will inevitably push the output distribution away from that of the original training set. Since the aim of a parameterization is to be able to extend the original distribution, rather than simply reproduce it, this isn't too much of a concern. It is also worth noting that 70% of the SDF-GAN parameter space corresponds to feasible geometries, with the equivalent being over 50% for AeroINR. This is much higher than traditional airfoil parameterization approaches, and thus shows that the lack of designer bias in an image-based approach does not hinder the parameterization efficiency, and in fact with a well-trained model is able to enhance it. These results are particularly promising for AeroINR, which aims to simply remain competitive whilst maintaining scalability, rather than necessarily outperforming alternative approaches.

Fig. 7 and 8 show visualisations of feasible aerofoils at randomly chosen positions of the parameter space for AeroINR, SDF-GAN and Bezier-GAN, and auto-encoded reconstructions of validation airfoils for SDF-GAN and AeroINR respectively. In Table 3, a comparison of the higher frequency airfoil variations is shown. Here, output airfoils

(a) SDF-GAN. Errors are {1.4, 1.7, 3.4, 1.4}e-03 respectively.

(b) AeroINR. Errors are {2.0, 5.1, 5.5, 3.4}e-03.

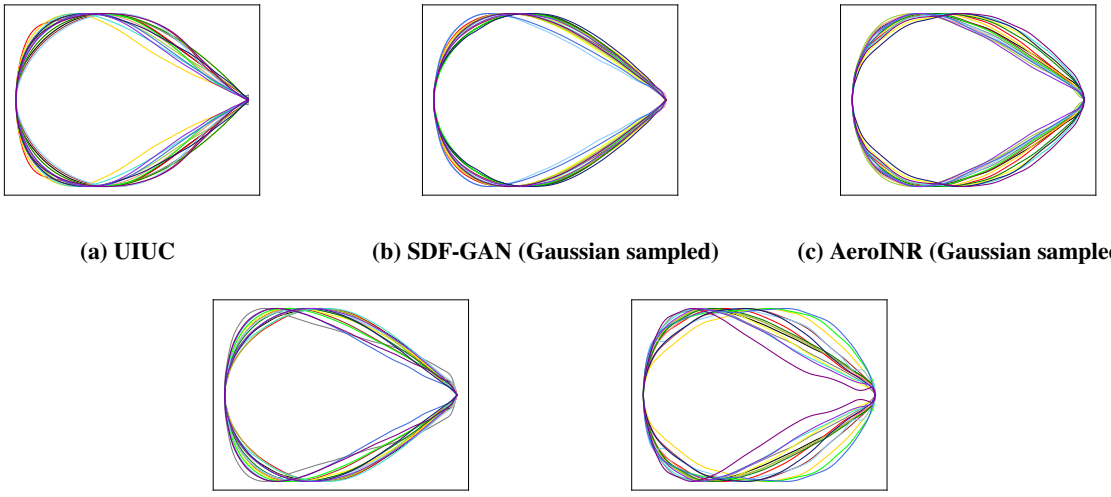
Fig. 8 Visualisation of reconstructed airfoils (blue, solid line) from validation set.

have been standardized in both camber and thickness (e.g. Fig. 9) to disregard the contributions of large scale variations in the diversity evaluation. These larger variations are often pre-specified by a designer based on experience, and it is the smaller variations around the primary shape that is of interest. In addition, traditional parameterizations are known to suffer from an inability to separate low-order shape variations from these higher order variations [59]. To quantify this capability, the maximum area enclosed between any two airfoils from a specified number of randomly sampled airfoils is computed and compared for SDF-GAN and AeroINR relative to the original training set of UIUC airfoils. Only feasible airfoils (as determined by XFOIL convergence) are included, with this also discounting around 20% of the original UIUC set. For each model, airfoils are sampled from both the Gaussian prior and uniform distributions, with the uniform distribution taken from $z \in [-3, 3]$, with this corresponding to over 99% of the prior distribution ($\mathcal{N}(0, 1)$) space. The results show that both SDF-GAN are AeroINR are able to output a greater range of diversity in high frequency shape variations than that of the original UIUC set, as long as the sampled parameters are taken from a uniform distribution. This is unsurprising, since the more diverse airfoil shapes are expected to be towards the edge of the learned parameterization space, whereas the central values represent the mean airfoil from the dataset. AeroINR diversity tends to be somewhat higher than SDF-GAN, with this finding in agreement with the relative diversity results from Table 2. The corresponding max area pairing for each source distribution is shown in Figure 10.

To gain some understanding of the structure of the learned parameterization space, Fig. 11 shows how lift-drag coefficient of the corresponding airfoil at the given position varies throughout the space, for both SDF-GAN and AeroINR. Each sub-plot corresponds to a cross-sectional variation in a two dimensional subspace. All other dimensions are fixed to the origin. Each sub-plot represents unique dimensions, and since both latent spaces have sixteen dimensions, eight sub-plots are required for each to visualize all dimensions. Both plots show a clear structure within the latent space, with distinct regions of the space containing airfoils with similar aerodynamic performance. This is also the case for infeasible airfoils (visualized as empty space), which for both models tend to be clustered towards the edge of

Table 3 Comparison between diversity of source dataset (UIUC), SDF-GAN and AeroINR after standardization of low frequency modes. Diversity is evaluated through the maximum area enclosed between any two of all sampled airfoils.

Airfoil Source	Sampled Distribution	Number Sampled	Max Area Enclosed
UIUC	N/A	980	0.0746
SDF-GAN	Gaussian	500	0.0365
AeroINR	Gaussian	500	0.0625
SDF-GAN	Gaussian	1000	0.0498
AeroINR	Gaussian	1000	0.0655
SDF-GAN	uniform	500	0.0586
AeroINR	uniform	500	0.0822
SDF-GAN	uniform	1000	0.0887
AeroINR	uniform	1000	0.0811



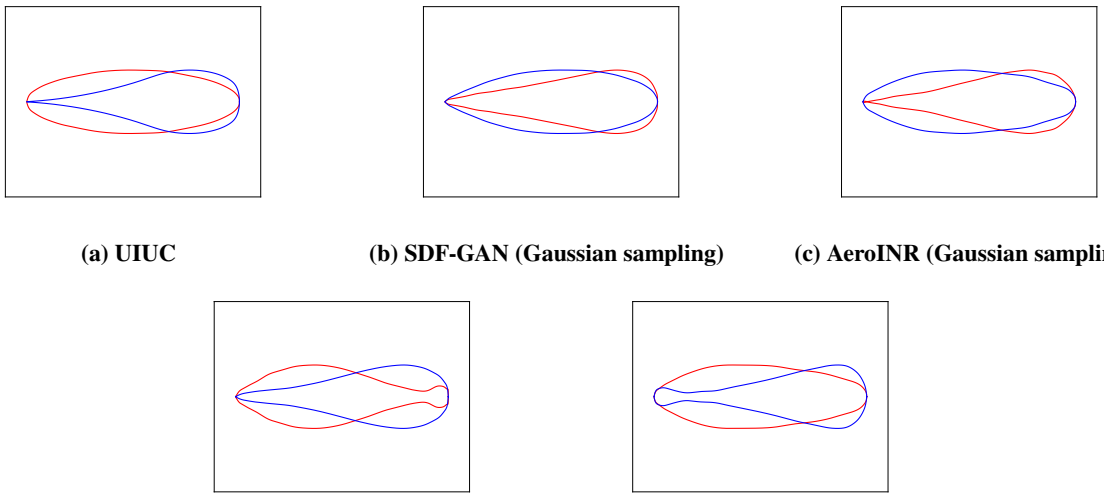
(a) UIUC **(b) SDF-GAN (Gaussian sampled)** **(c) AeroINR (Gaussian sampled)**

(d) SDF-GAN (uniformly sampled) **(e) AeroINR (uniformly sampled)**

Fig. 9 Examples of standardized airfoils from three sources: UIUC, Gaussian-sampled SDF-GAN and AeroINR, and uniform-sampled SDF-GAN and AeroINR.

the space, as expected. Such structure suggests the parameterization should be systematically searchable within an optimization procedure; the discontinuities seen could however be problematic for a gradient-based solver, although in practice these may well be an artifact of the two-dimensional visualization, rather inherent in the spatial structure.

The final result for the 2D parameterization study concerns the scalability of each approach, specifically the growth in training parameter number as geometric resolution is increased. Fig. 12 compares this behaviour for SDF-GAN relative to AeroINR and an auto-decoding variation it. The number of training parameters is taken by scaling up the baseline model input and output layers (as needed) by the minimum number of parameters such that model structure is consistent with the new data resolution. The lack of scalability in the basic grid-based approach of SDF-GAN is



(a) UIUC **(b) SDF-GAN (Gaussian sampling)** **(c) AeroINR (Gaussian sampling)**

(d) SDF-GAN (uniform sampling) **(e) AeroINR (uniform sampling)**

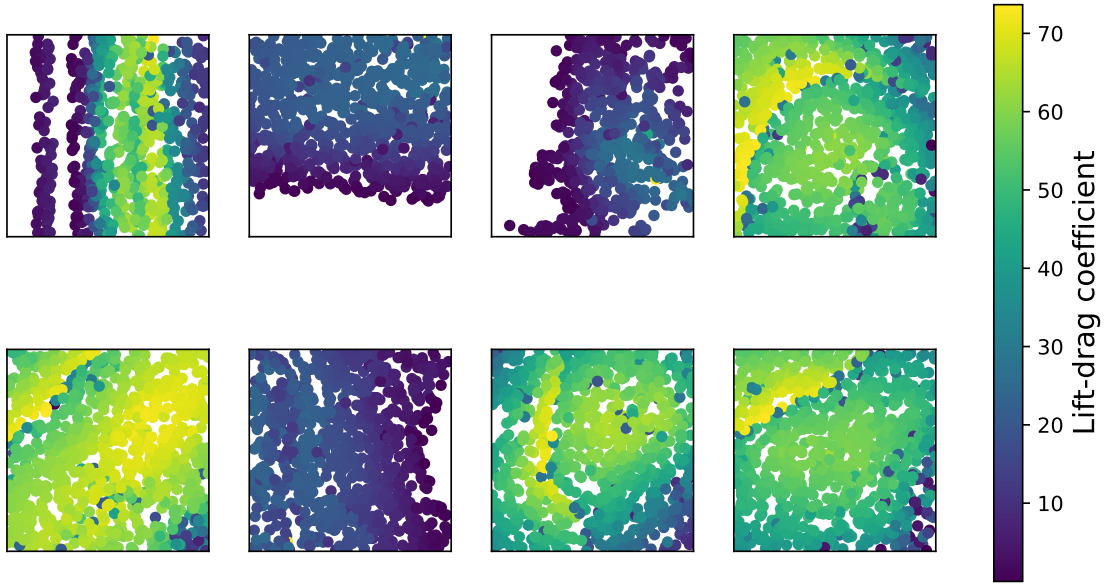
Fig. 10 Standardized airfoil pairs which enclose the maximum area relative to all other sampled pairs. The sampled airfoils come from three sources: UIUC, SDF-GAN and AeroINR with Gaussian sampling, and SDF-GAN and AeroINR with uniform sampling. In all SDF-GAN and AeroINR cases the sampled number is set to 1000.

immediately apparent from the plot. In contrast, both variations of AeroINR are significantly more scalable, with the auto-decoding formulation - which removes the encoder entirely and instead finds the corresponding encoding through direct gradient-based updates of these parameters to minimise the reconstruction loss - having a model size completely independent from the geometric resolution. In practice, some model changes will likely be needed to capture the additional variability able to be represented at higher resolutions, but the lack of enforced dependence on model architecture removes the greatest immediate constraint to scalable learning. In addition, unlike convolutional models which also remove dependence on grid size by learning weights of small kernels, the AeroINR solution does not require processing of the full resolution input grids during training. Such a requirement is a strong constraint due to RAM limitations; instead AeroINR uses INRs to learn efficient, compact representations of the original grids.

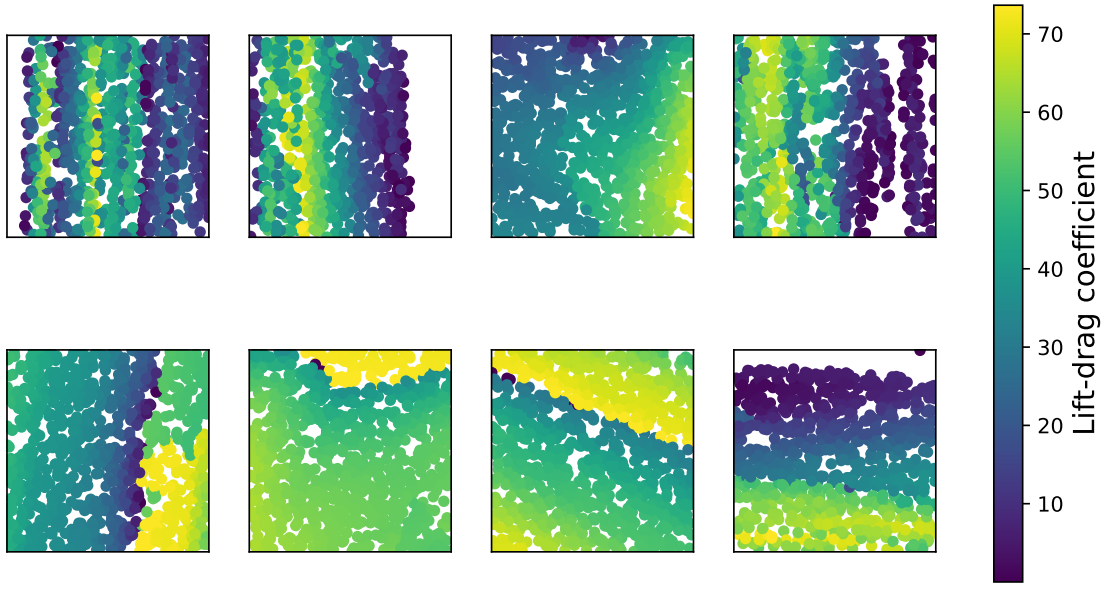
IV. 3D Wing Parameterization

A. Dataset

To train a three-dimensional wing parameterization model, a dataset of realistic wing designs are first required. This is created synthetically through the SolidWorks CAD software. A baseline geometry is first created by linearly lofting between three airfoils placed at specified spanwise positions. The airfoils are aligned at their leading edge, whilst their chord lengths are differed to create a tapered wing. The root airfoil is set to a 100mm chord, with the mid-section being 87.5mm and tip 74.9mm. The tip is placed 500mm away from the root, and this distance is fixed, along with the relative chord lengths. To create a sufficiently varied dataset, the mid-section and tip airfoils are switched randomly



(a) SDF-GAN



(b) AeroINR

Fig. 11 Visualisation of the learned latent space through sampling of 2D cross-sections around the origin. Each point in space corresponds to a single airfoil, colour-coded by its lift-to-drag ratio (light colours correspond to higher values). Note that white regions correspond to infeasible geometries.

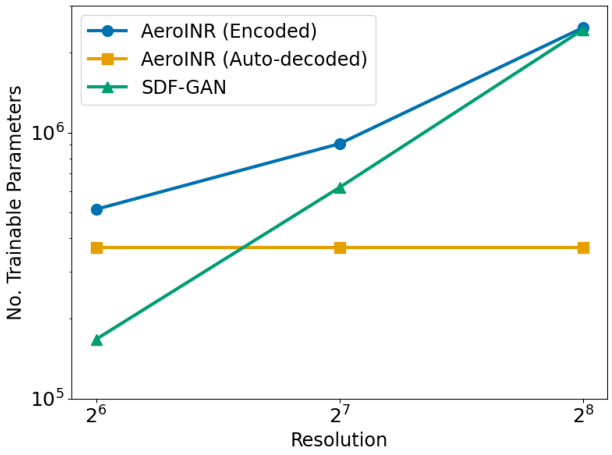


Fig. 12 Trainable parameter growth with resolution for the grid-based SDF-GAN and AeroINR.

with alternative airfoils from the UIUC dataset, whilst the spanwise position of the mid-section airfoil is also varied to allow differences in tapering. The decision to keep all other parameters fixed was made due to the difficulty in ensuring geometrically valid wings whilst automating the creation pipeline. The final synthetic shape is then exported as an STL file.

Figure 13 shows the process by which STL wing descriptions are converted into training data for the AeroINR-3D model. Firstly, the STL is converted to a triangular mesh using the trimesh software package [60] in Python. To convert to an SDF representation, a 128^3 grid is defined over the fixed spatial domain containing all synthetic wing geometries: $x_0 \in [-10, 110], x_1 \in [-15, 15], x_2 \in [-50, 550]$. An internal trimesh algorithm is then utilized to calculate SDF values at each cell centre, which uses a tree-based querying procedure to find the closest triangular face, and projection vectors to compute the exact position and sign.

To train AeroINR-3D, the INR surrogates are created during the pre-processing stage as a distinct step. To improve training performance, all coordinates are first standardized and SDF values normalized. Each synthetic geometry is then used to train independent INRs through these coordinate-SDF pairs. The architectural simplicity allows training to be carried out in just 10 minutes on a single CPU, with each geometry having INR parameters initialized to those of the previous geometry to leverage shared characteristics and minimize inefficiencies. Since the first geometry has no such prior to initialize from, the training time was increased to 30 minutes. A basic L_1 reconstruction loss between actual (ϕ_i and reconstructed ($\widehat{\phi}_i$) field values was used as the training objective, of the form,

$$\text{INR loss} = \frac{1}{N}\sum_{i=1}^{N}|\phi_i - \widehat{\phi}_i| \,, \tag{7}$$

which was preferred to L_2 due to the reduced sensitivity to outliers; since the reconstructed geometry depends only on those values near the boundary, larger errors away from the surface can be tolerated.

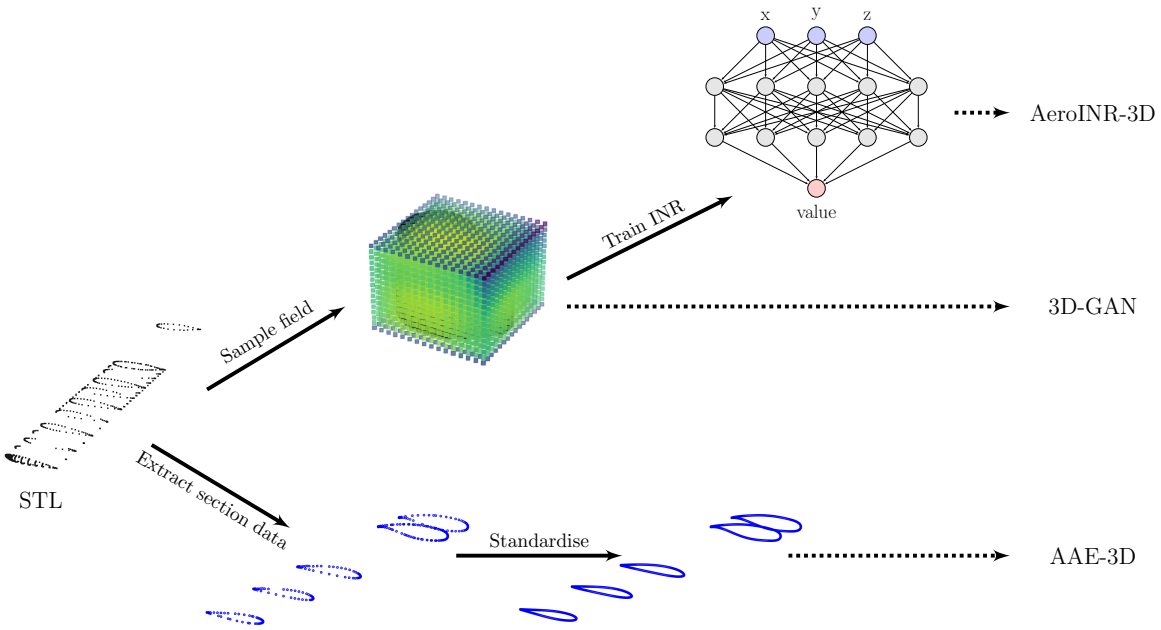


Fig. 13 Data pre-processing pipeline for 3D blade representation consistent with AeroINR-3D and benchmark models.

To assess the trained surrogate quality, an aerodynamic comparison of the underlying geometries is carried out through lift-drag evaluation. This is implemented through application of the Vortex Lattice Method (VLM) solver in AeroSandbox [42, 61]. AeroSandbox is a publicly available toolkit for aerodynamic design optimization of both two and three-dimensional geometries. Here the three-dimensional, inviscid VLM solver is utilized for evaluation of lift and induced drag coefficients, with geometries processed into five sets of airfoil coordinates at given spanwise locations. A zero-mass wing is assumed here, with a speed of 10 m/s and angle-of-attack equal to 5 degrees. Ten panels are used in each planar direction, with convergence verified by ensuring the change in lift-drag coefficient is less than 10% when doubling this number in each direction independently; if convergence is not achieved the geometry is classified as infeasible. To ensure reliability, validation was carried out against the well-known XFLR5 solver for 3D aerodynamic analysis, with predictions at $\alpha = 5°, V = 10$ m/s agreeing to within 10% for lift and around 20% for drag. In addition, the relative ranking of lift-drag for different geometries remained consistent across both solvers, indicating its reliability for the low-fidelity analysis implemented here.

Whilst the individual INR training objectives aim to minimize differences in the field reconstructions, in practice the aim of these learnt representations is to train an effective three-dimensional wing parameterization model. In this context, it is the overall distribution and diversity of training geometries that is of interest - particularly with regards to aerodynamic performance - rather than individual reconstruction quality. To visualise this, Figure 14 shows the original training set lift-drag distribution compared to the INR surrogate equivalent. Whilst some difference can be seen,

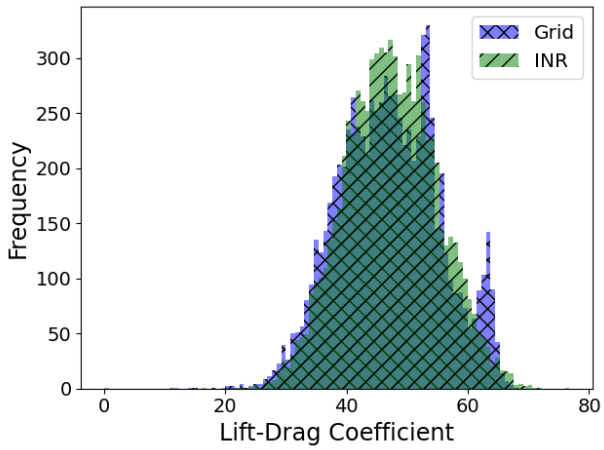


Fig. 14 Lift-drag coefficient distribution for INR blade reconstructions and original SDF grid reconstructions. Both sets of grids were sampled at 128^3 resolution.

particularly with regards to the smoothness of the two distributions, the overall range is broadly similar (with the median INR coefficient being slightly higher). It is possible that the smoother surrogate distribution can be attributed to the specific training implementation used, in which transfer learning is leveraged to reduce train time and thus provides an artificially learned link between independent geometries. Independent training - whilst increasing time costs - may lessen this effect.

In total, 10,000 wing geometries are created. Baseline training for all models use 5000 of these, with the rest used during dataset size ablation studies. 10% of geometries are held back for testing, whilst 10 fold cross validation is used during training to evaluate model performance during hyper-parameter tuning. Note that these fractions remain consistent during dataset size ablations.

B. Benchmarks

3D-GAN For the voxel-based benchmark, the 3D-GAN framework introduced by [62] is used. A small amount of hyper-parameter tuning around the original baseline is carried out - specifically with regards to the latent dimensionality, feature scaling and training process - but for the most part the original implementation is followed. In fact, the model was found to be quite sensitive to hyper-parameter changes (particularly the learning rate), such that remaining faithful to the original work was somewhat enforced. Training was carried out on Nvidia A100 GPU nodes to ensure practical train times, with each having 80 GB of VRAM which was sufficient for most hyper-parameter configurations. Training took place over 9 hours, with off-line evaluation carried out at 3 hour intervals to select the best model.

To mitigate very low feasibility rates (specifically in terms of contour extraction and subsequent sectioning for input into AeroSandbox), some post-processing is required (algorithms 1 and 2). As detailed in algorithm 2, two key elements of post-processing are required. The first is the exclusion of outlying points from the output voxel grid, which

Algorithm 1 Geometry extraction pipeline for 3D-GAN. Note that sort operations are in order of increasing size.

Input: Voxel grid $\{(x^{(i)}, \phi^{(i)})\}_{i=1}^{N}$ representing coordinates and corresponding geometry occupancy values
Compute trimesh object corresponding to surface reconstruction through marching cubes
Define $x_{2,\max} := \max(x_{2,i})\ \forall i \in S$, $x_{2,\min} := \min(x_{2,i})\ \forall i \in \mathcal{S}$, where $\mathcal{S}$ corresponds to the set of mesh vertices
Compute the geometry spanwise length: $\Delta x_2 = x_{2,\max} - x_{2,\min}$
for $x_{2,k} \in \{x_{2,\min}, x_{2,\max}\}$ **do**
    Compute list of unique spanwise positions:
    $x_{2,\text{unique}} := [\text{sort}\left(\text{unique}(x_{2,i})\right)]\ \forall i \in S$, sorted by $|\text{unique}(x_{2,i}) - x_{2,k}|$
    if $x_{2,k} = x_{2,\min}$ **then**
        $x_{2,\text{root}} \leftarrow$ FindDenseSection$(x_{2,\text{unique}}, x_{2,k}, \text{True})$
    else
        $x_{2,\text{tip}} \leftarrow$ FindDenseSection$(x_{2,\text{unique}}, x_{2,k}, \text{True})$
Compute the processed geometry spanwise length: $\Delta x_2 = x_{2,\text{tip}} - x_{2,\text{root}}$
for $\Delta x_{2,k} \in \{0.01, 0.25, 0.5, 0.75, 0.99\} \times \Delta x_2$ **do**
    $x_{2,k} \leftarrow$ ProcessSpanwiseSections$(x_{2,\text{unique}}, \Delta x_{2,k}, x_{2,\text{root}}, \{x^{(i)}\})$
Output: Point cloud at 5 distinct spanwise locations

is typically noisy and containing fragmented contours. A search is carried out to find the spanwise positions of the dense sectional data within the voxel grid for extraction. Secondly, after finding the spanwise positions, airfoil standardization is required for input into AeroSandbox. In practice, direct interpolation of the mesh vertices leads to very poor quality fits due to noise in the data; instead marching squares is carried out directly over the 2D occupany field at each spanwise position. This leads to higher quality airfoils, but typically multiple contours can exist in the 2D plane, such that a filtering process is constructed to determine the curve consistent with that of the full three-dimensional contour.

AAE-3D AAE-3D was introduced by [63] as an AAE framework modified to deal with coordinate data, such that both generation and reconstruction of 3D point clouds is possible. Specifically, the implementation is modified to allow reconstruction loss compatibility with a set of points $\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^{N}$ as input. This is done through utilization of the Chamfer pseudo-distance, which is a point-wise comparison metric known to favour regions of higher density. It is defined through

$$\text{CD}(\mathcal{S}_1, \mathcal{S}_2) = \sum_{\mathbf{x} \in \mathcal{S}_1} \min_{\mathbf{x}' \in \mathcal{S}_2} ||\mathbf{x} - \mathbf{x}'||_2^2 + \sum_{\mathbf{x}' \in \mathcal{S}_2} \min_{\mathbf{x} \in \mathcal{S}_1} ||\mathbf{x} - \mathbf{x}'||_2^2 \ . \tag{8}$$

such that the squared distance between each point in the first set $\mathcal{S}_1$ and its neareast neighbour in $\mathcal{S}_2 \in \mathbb{R}^3$ is computed. Since a point cloud is an unordered set of points, the model output must be independent of input ordering. To implement this, a PointNet-like approach [64] is used, in which a max pooling operation is applied after the convolutional layers of the encoder to ensure latent embeddings are invariant to permutations. Since the generator is a simple multi-layer perceptron, such a transformation is not possible. Instead, it acts as a one-to-one mapping such that order is preserved between input and output.

As with 3D-GAN, low feasibility rates appear to be unavoidable without a targeted post-processing procedure. In

Algorithm 2 Geometry post-processing methods for 3D-GAN. Note that sort operations are in order of increasing size. The length$(\cdot)$ operation computes the number of points in the contour.

function FindDenseSection($x_{2,\text{unique}}$, $x_{2,k}$, edge)
  Set $m = 0$
  while $\sum_{j \in C_k} j < 100$ **do**
    $C_k := \{(x_{0,i}, x_{1,i}, x_{2,i}) \mid x_{2,i} = x_{2,\text{unique}}[m]\}$
    $m \leftarrow m + 1$
  if $|x_{2,\text{unique}}[m] - x_{2,k}|/x_{2,k} > 0.1$ **and not** edge **then**
    Terminate infeasible geometry
  return $x_{2,\text{unique}}[m]$
function ProcessSpanwiseSections($x_{2,\text{unique}}$, $\Delta x_{2,k}$, $x_{2,\text{root}}$)
  $x_{2,k} = x_{2,\text{root}} + \Delta x_{2,k}$
  Interpolate occupancies at coords $\{(x^{(i)}) \mid x_{2,i} = x_{2,k}\}$ and find all contours $\{\Gamma\}$
  if $\Delta x_{2,k} \in \{0.01, 0.99\} \times \Delta x_2$ **then** edge = True **else** edge = False **end if**
  $x_{2,\text{section}} \leftarrow$ FindDenseSection($x_{2,\text{unique}}, x_{2,k}$, edge)
  Define centroid of 3D geometry section: $\mathcal{S}_{k,c} := \{(\overline{x^{(i)}}) \mid x_{2,i} = x_{2,\text{section}}\}$
  Initialise empty array A
  for $\gamma \in \Gamma$ **do**
    if Polygon(γ) contains $\mathcal{S}_{k,c}$ **then**
      $A \leftarrow A + [\text{length}(\gamma)]$
  if $\text{sort}(A)[-1] > \text{sort}(A)[-2] \times 5$ **then**
    Terminate infeasible geometry
  return γ corresponding to element $\text{sort}(A)[-1]$

this case, the generator is unable to output sectional aerofoil coordinates as expected based on the input; rather, the output is a point cloud with points distributed across spanwise coordinate positions. To extract five distinct airfoils at precise spanwise locations - as required by the AeroSandbox solver - points are manually gathered into distinct clusters through the aggregation process detailed in algorithm 3. This allows the construction of airfoils, whilst ensuring validity through a specified cluster-width tolerance, beyond which distinct airfoils are deemed not to exist and the geometry labelled as infeasible.

C. Parameterization Results

Fig. 15 shows the geometry visualization for generated (Gaussian sampled) and reconstructed outputs of each parameterization model. For the field-based models, mesh vertices corresponding to the extracted contour are shown, whilst for AAE-3D output points prior to post-processing are displayed. Whilst very high in feasibility, 3D-GAN is seen to produce much noisier outputs than the alternatives which leads to poor quality meshes, and in addition the variability in output in visually subtle. This is however similar to the training data, so the model appears to have learned the distribution well, but with limited generalization. AAE-3D shows a higher quality output, but overall with a much lower feasibility due to an inability to learn fixed cross-sectional positions at specific points across the span. In contrast, AeroINR-3D has both very high quality mesh outputs (with no post-processing), in addition to high feasibility and novelty in output. Further examples of novel output geometries are shown in Fig. 16, in which geometries sampled from

Algorithm 3 Post-processing Procedure for AAE-3D.

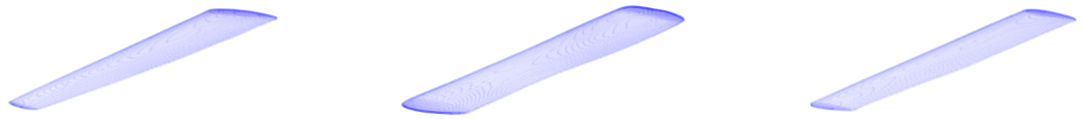
Input: Point cloud $\mathcal{S}$ with coordinates $\{(x_{0,i}, x_{1,i}, x_{2,i})\}$
Compute the geometry spanwise length: $\Delta x_2 = \max(x_{2,i}) - \min(x_{2,i})\ \forall i \in \mathcal{S}$
Apply K-means clustering to partition points into 5 clusters C_k based on the x_2 values.
for each cluster C_k **do**
    Compute the maximum deviation of x_2 values from the mean $\overline{x}_{2,k}$: $\Delta x_{2,k} = \max(|x_{2,j} - \overline{x}_{2,k}|)\ \forall j \in C_k$
    if $\Delta x_{2,k} > 0.05 \times \Delta x_2$ **then**
        Terminate infeasible geometry
    else
        Assign $x_{2,j} = \overline{x}_{2,k}\ \forall j \in C_k$
        Standardize sectional aerofoils through PARSEC interpolation
Output: Point cloud at 5 distinct spanwise locations

the uniform distribution are shown.

Tables 4 and 5 show the quantitative results for the 3D wing parameterization, specifically through aerodynamic evaluation with AeroSandbox and via generative model performance metrics (as in Table 2 for the 2D case) respectively. For Table 4, each result is from evaluation across 100 randomly selected geometries for both generation and reconstruction, whereas for Table 5 constraints from computational memory limit the number evaluated for MMD and relative diversity to 50 geometries. Similar to the 2D case, generative performance metrics are evaluated for feasible geometries only.

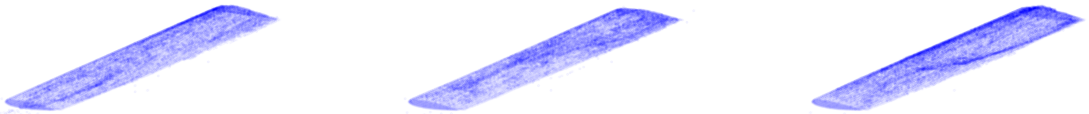
With regards to the aerodynamic evaluation, Table 4 shows both AeroINR-3D and 3D-GAN to have high feasibility rates across the learned space, whereas AAE-3D is significantly lower (even after post-processing). This is true both for generative and reconstruction functionalities, noting that 3D-GAN has no capability for the latter. Surprisingly, AAE-3D actually has lower feasibility when asked to reconstruct a given geometry relative to random sampling. To assess the extent to which each parameterization yields improved geometries relative to those in the training set, the lift-to-drag ratio is evaluated and the fraction of output geometries with higher values than that seen in the entire training set is recorded. Here the AAE-3D outperforms the alternative models, although it should be noted that the lower feasibility means the total number of improved candidate geometries is similar. In addition, despite a much lower feasibility in reconstruction (11% relative to 92%), AAE-3D also has significantly higher error in lift-to-drag for the reconstructed geometries, suggesting the quality is also lower.

In terms of generative capability evaluation, results are somewhat more mixed. The capacity for novelty in the AeroINR-3D is highlighted by the high relative diversity, with the variability in output geometries exceeding that of the training set (see Figure 16), whilst retaining high feasibility and reconstruction capability as shown in Table 4. Latent space consistency is, however, strikingly low for both AeroINR-3D and 3D-GAN, suggesting a disconnect between latent space position and geometric output. Despite this, some structure can be seen in Figure 17, which visualizes two-dimensional subspaces of all latent dimension pairwise combinations, suggesting the space may still be sufficiently structured as to be searchable. In addition, similar to AeroINR-2D (Table 2), it may be that retaining the direct relationship between generative learning and original field values in the meta-learning algorithm (rather than

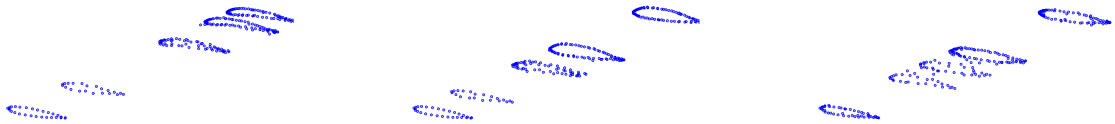


(a) AeroINR-3D - generated

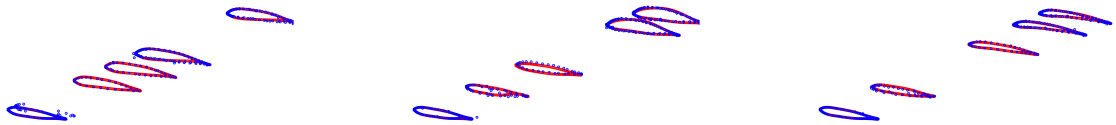
(b) AeroINR-3D - reconstructed



(c) 3D-GAN - generated



(d) AAE3D - generated



(e) AAE3D - reconstructed

Fig. 15 Visualisation of randomly sampled output mesh vertices from AeroINR-3D, 3D-GAN and AAE-3D. Note that reconstructions are drawn from original geometries (red) in the test set.

abstracting this away through direct weight learning) may improve this markedly.

Finally, Fig. 18 shows the lift-to-drag distribution across 10,000 randomly sampled geometries from each of the models. Note that due to the different rates of feasibility, the actual number shown differs across models. AeroINR-3D is shown to have the widest distribution, covering both the smallest and largest lift-to-drag values of all geometries shown. Compared to the benchmarks, this parameterization is capable of capturing the most varied range of geometries,

Table 4 Wing parameterization capability assessed with aerodynamic evaluation of trained model output. Note that reconstruction values are for test set geometries.

	Generation		Reconstruction	
Model	Feasibility	Fraction Improved	Feasibility	Lift-Drag Error
AeroINR-3D	0.70 ± 0.04	0.07 ± 0.01	**0.92 ± 0.03**	**0.20 ± 0.01**
3D-GAN	**0.80 ± 0.02**	0.05 ± 0.02	N/A	N/A
AAE-3D	0.35 ± 0.04	**0.16 ± 0.01**	0.11 ± 0.01	0.30 ± 0.02

Table 5 Generative performance comparison between AeroINR-3D and alternative approaches.

Model	MMD^2	Latent Space Consistency	Relative Diversity
AeroINR-3D	0.1450 ± 0.0068	0.0101 ± 0.0087	**10.3162 ± 6.3284**
3D-GAN	0.0403 ± 0.0004	**0.0863 ± 0.0212**	1.7242 ± 0.2600
AAE-3D	**0.0402 ± 0.0001**	–	0.7515 ± 0.0633

whilst still retaining reconstruction capability of the original dataset. The distribution is also the closest to a Gaussian, which is likely to be desirable in an optimization search due to the smoothness of the frequency curve across geometric performance.

D. Optimization

A well known result from introductory aerodynamics states that a finite wing with no sweep or dihedral has an induced drag coefficient of,

$$\mathrm{Cd}_i = \frac{\mathrm{Cl}^2}{\pi \mathrm{ARe}}, \tag{9}$$

for aspect ratio $\mathrm{AR} = \mathrm{span}^2/\mathrm{area}$ and efficiency factor e. For a fixed lift coefficient and constant aspect ratio, induced drag is mimized for $\mathrm{e} = 1$, such that lift per unit span varies with spanwise position elliptically. In this section, geometric

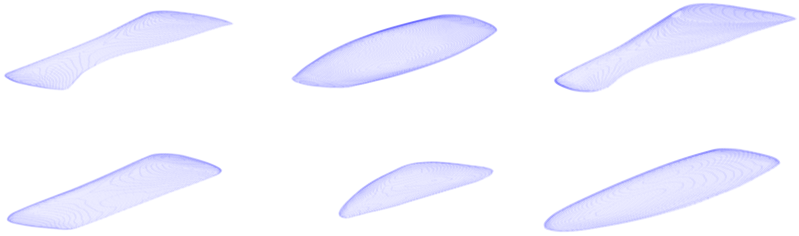


Fig. 16 Sampling of a uniform latent prior distribution $\mathcal{U}(-3, 3)$ with AeroINR-3D shows the diversity in geometric output.

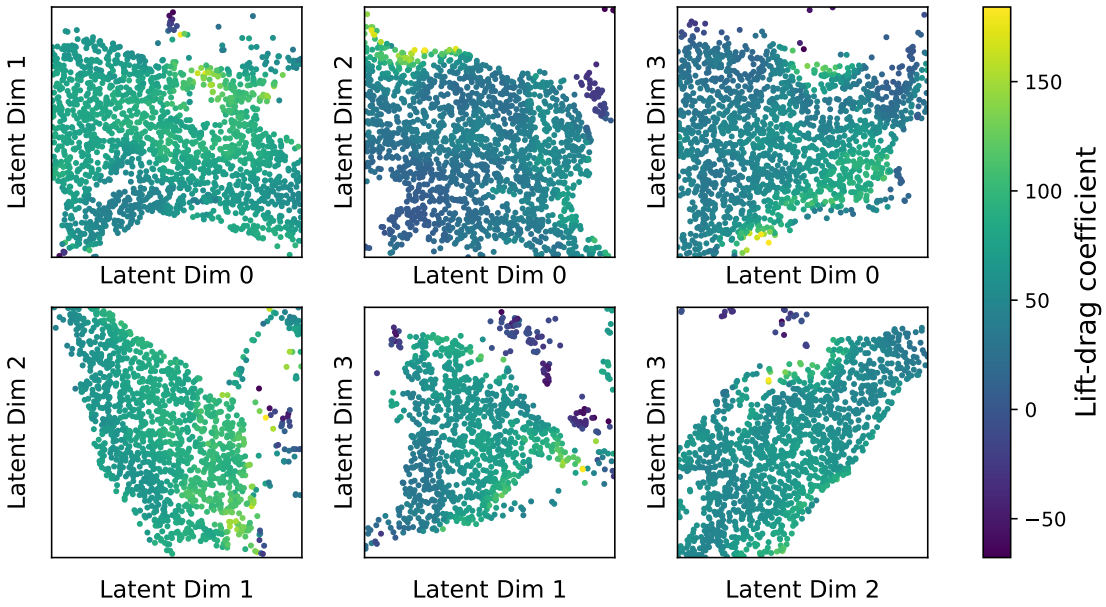


Fig. 17 Visualisation of the learned latent space through sampling of 2D cross-sections around the origin. Each point in space corresponds to a single geometry, colour-coded by its lift-to-drag ratio. Note that white regions correspond to infeasible geometries.

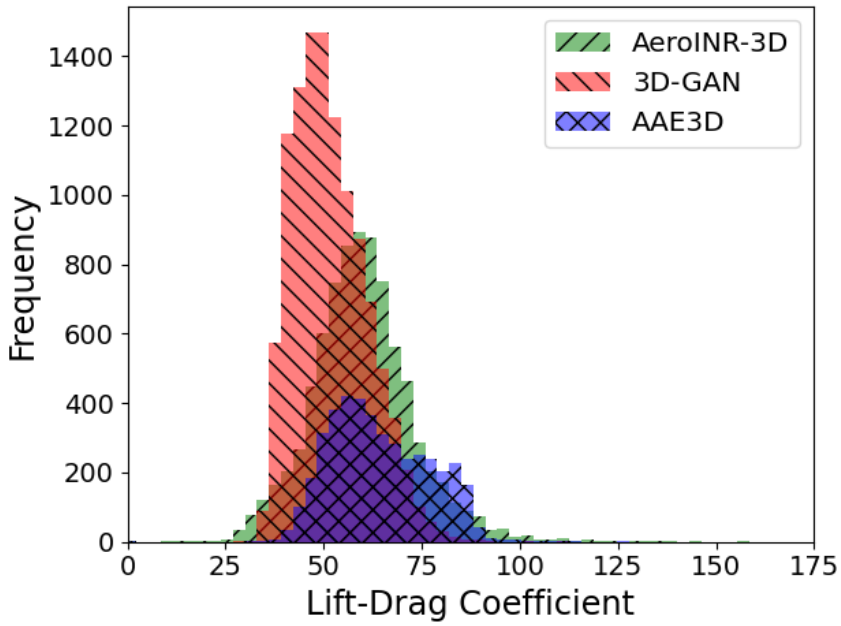


Fig. 18 Lift-drag coefficient distributions of 10,000 randomly sampled parameter-space points from each model. Note that infeasible geometries are not counted in the plot.

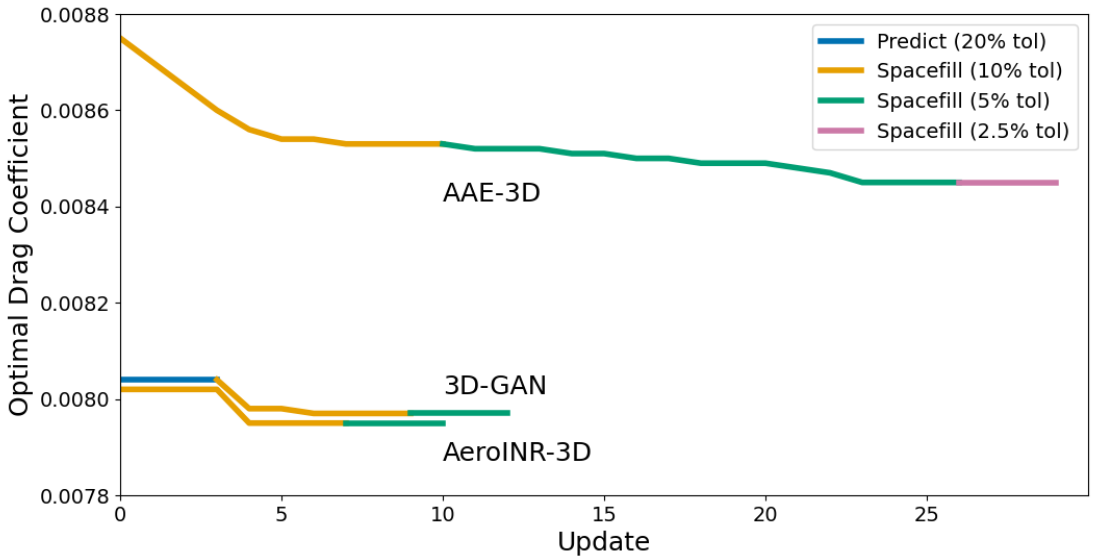


Fig. 19 Optimization history for each model showing optimal performance progression.

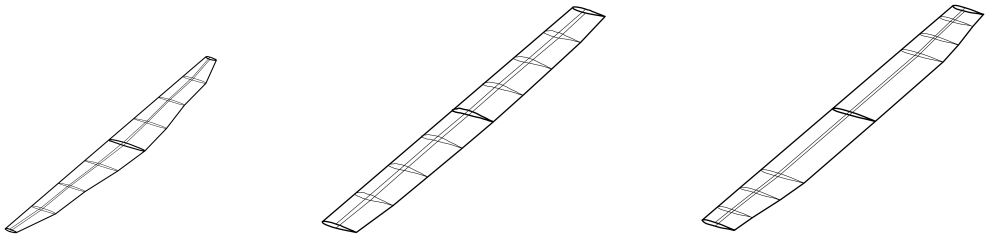
optimisation for a wing with these constraints is carried out using each parameterization model to verify capability in a realistic design problem with a known result.

To ensure realistic results, each wing has its span fixed to five times the maximum chord length, which prevents the solver from increasing the wingspan to arbitrarily high values - not possible in a multi-disciplinary optimization due to mechanical stresses - to solve the problem. The angle-of-attack varies between -20 and 20 degrees to fix the lift coefficient to within a 1% tolerance of 0.5. In addition, the problem is further simplified by requiring the chord length to decrease away from the mid-section, which is a reasonable assumption. All aerodynamic evaluations are carried out the inviscid Vortex Lattice Method solver from AeroSandbox; due to the sensitivity of drag calculations, the number of panels is increased in each dimension to 40, with the number doubled as usual to verify convergence.

A two-stage optimization process is carried out, using the proprietary OPTIMAT solver from Rolls-Royce. The first stage is surrogate-based, in which a classifier and regressor are fit to the feasibility constraint and drag values respectively. Each surrogate is trained on 10,000 latent space positions selected via LHS. Candidate surrogates are taken from the MATLAB regression and classification learner apps, through evaluation of 2000 test geometries against coefficient of determination for the regressor, and true positive/true negative rates for the classifier. In practice, the number of feasible geometries satisfying the problem constraints was too low for AeroINR-3D to learn an effective surrogate. For 3D-GAN, a linear regressor ($R^2 = 0.27$) and narrow neural net classifier are chosen (TPR = 0.21, TNR = 0.95), and AAE-3D a kernel-based SVM regressor ($R^2 = 0.51$) and RUS boosted trees classification model (TPR = 0.85, TNR = 0.80) are chosen. To optimize, a genetic algorithm is used around regions local - within a specified tolerance - to the current optimum. The population size is set to 1600 (400 for 3D-GAN due to memory constraints), the mutation rate to 0.2, and

crossover rate to 0.5 (intermediate method).

The second stage of optimization is a passive space-filling procedure in the hyper-rectangular region around the current optimum . Here, 300 points are selected to maximize the distance in the specified local region to the already sampled points. For both stages, local regions are defined as point within a tolerance (20% and 10% for surrogate and space-filling respectively) of the current optimum. If no new optimum is found after a single iteration, the size of the local region is further reduced by a factor of two. After three consecutive iterations with no update the optimization process moves on to the next stage.



(a) AeroINR-3D **(b) 3D-GAN** **(c) AAE-3D**

Fig. 20 Optimized geometries for each model parameterization.

The optimization history is shown in Fig. 19. No improvements were found using surrogate optimization for AAE-3D or AeroINR-3D, in the latter case due to a lack of suitable surrogates. Despite this, all models were able to improve their original LHS optimum with the space-filling search. The final optimized values show AeroINR-3D to be the most effective parameterization for the problem, despite the lower feasibility with the specified problem constraints (specifically the requirement for chord length to reduce away from mid-section). The increased geometric diversity of the model produced a final geometry differing markedly from the alternative optimal solutions (see Figure 20), which was able to make up for feasibility limitations. In contrast AAE-3D - with low feasibility rates and low diversity - was unable to reduce drag to the same extent, with a 6% efficiency loss compared to the AeroINR-3D and 3D-GAN. The corresponding spanwise lift distributions are shown in Fig. 21, with the optimal distributions close to elliptical as expected. It is worth stating that in a practical design application with a high fidelity CFD solver this optimization approach would be somewhat infeasible. However, given that the objective here is to assess the extent to which practical geometric designs are captured by each parameterization, this limitation was considered acceptable.

V. Discussion

This paper has shown the applicability of direct grid-based learning approaches for geometric parameterization in aerodynamics, with the effectiveness of the basic approach shown in 2D through SDF-GAN. This was shown to be the case even with no smoothing process on output designs. A solution to the problem of lack of scalability with grid-based

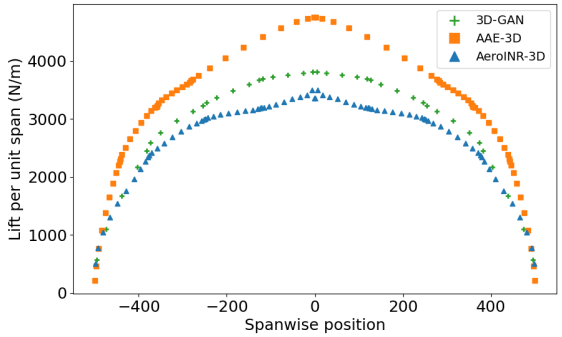


Fig. 21 Lift distribution for each optimized geometry.

learning has also been put forward. A scalable approach using INR surrogates of geometric fields was implemented in 2D with AeroINR, in which a fully-contained meta-learning approach is used to learn both INR surrogate weights and weight distribution across geometries concurrently. Such an approach was shown to be very competitive relative to both the 2D airfoil parameterization benchmarks, and the original SDF-GAN approach, outperforming all on two of the three metrics. This was extended to 3D and compared to deep learning benchmarks for geometric learning, specifically a voxel-based GAN and point-cloud based AAE.

Table 6 provides a summary comparison of AeroINR-3D relative to the 3D benchmark parameterizations. The efficiency of the INR-based approach is immediately apparent, with the ratio of model parameter number to output resolution an order of magnitude or more lower due to the compressed representation, highlighting the scalability of this approach. The quality if geometric output is also clear from the table, with no post-processing required despite the high levels of feasibility seen. AeroINR-3D also more robust across hyper-parameter configurations than the alternatives, where robustness refers to configurations able to learn non-negligible feasibility. The model is very fast to train, taking only minutes for thousands of geometries on a A100 GPU, compared to hours to achieve meaningful performance with 3D-GAN and AAE-3D. The results also reiterate the flexibility of the INR solution, with a technically unlimited output resolution determined by the number of sampled points. Implementation of a wing optimization problem using each of the three approaches has further proven the effectiveness in realistic aerodynamic design context.

In summary, the meta-learning approach for field learning put forward through AeroINR is a scalable, highly effective solution to geometric parameterization for aerodynamic geometries. The approach has been shown to break the dependency between the number of model weights and the input resolution, allowing scalability in both spatial dimension and resolution. This is the case even relative to convolutional architectures, due to the lack of direct grid processing ensuring RAM requirements are consistently manageable. The approach is generic, such all field types are applicable, including geometric (SDF/occupancy) and fluid-based. Finally, the continuous nature of INR surrogates ensures that sampling and learning on non-uniform field data is a trivial extension - demonstrably not the case with

Table 6 Summary of AeroINR-3D against benchmark models. The generative implementation of each model is used for comparison.

Feature	AeroINR-3D	3D-GAN	AAE-3D
Total Parameter No.	34 million	220 million	29 million
Latent Dimensionality	4	64	4
Total No. Output Values	2.1 million	2.1 million	1000
Post-processing	No	Yes	Yes
Robustness	57%	37%	8%
GPU Training Required	No	Yes	Yes
Max Representation Size	Unlimited	256^3	2000
Min Train Time	$\sim$ 1 min	$\sim$ 4 hrs	$\sim$ 9 hrs
Train Time Per Epoch	2 mins	75 mins	2 mins

comparative convolutional architectures. Despite these benefits, further work remains. The INR-based geometric compression means learning doesn't take place on the geometry directly. This can be somewhat mitigated by utilizing the fully-contained meta-learning approach of AeroINR-2D, but still downstream geometric points remain unseen. This may also cause difficulties if attempting to implement losses based on the geometric output itself, with this being several stages of processing removed (field construction->contour->mesh). Finally, the INR training itself could be further improved, specifically with the use of non-uniform sampling or the leveraging of sinusoidal activations for improved modelling of higher frequency components.

Acknowledgments

The funding provided by Rolls-Royce plc and EPSRC in support of this work is gratefully acknowledged. The authors also acknowledge the use of the IRIDIS High Performance Computing Facility at the University of Southampton, and would like to thank the associated support services for their help in the completion of this work.

References

[1] Ruh, M. L., Fletcher, A., Sarojini, D., Sperry, M., Yan, J., Scotzniovsky, L., van Schie, S. P., Warner, M., Orndorff, N. C., Xiang, R., Joshy, A. J., Zhao, H., Krokowski, J., Gill, H., Lee, S., Cheng, Z., Cao, Z., Mi, C., Silva, C., Wolfe, L., Chen, J.-S., and Hwang, J. T., *Large-scale multidisciplinary design optimization of a NASA air taxi concept using a comprehensive physics-based system model*, ???? https://doi.org/10.2514/6.2024-0771, URL https://arc.aiaa.org/doi/abs/10.2514/6.2024-0771.

[2] Abbott, I., and Von Doenhoff, A., *Theory of Wing Sections, Including a Summary of Airfoil Data*, Dover Books on Aeronautical Engineering Series, Dover Publications, 1959. URL https://books.google.co.uk/books?id=DPZYUGNyuboC.

[3] Kulfan, B., "Universal Parametric Geometry Representation Method," *Journal of Aircraft - J AIRCRAFT*, Vol. 45, 2008, pp. 142–158. https://doi.org/10.2514/1.29958.

[4] Hicks, R. M., and Henne, P. A., "Wing Design by Numerical Optimization," *Journal of Aircraft*, Vol. 15, No. 7, 1978, pp. 407–412. https://doi.org/10.2514/3.58379, URL https://doi.org/10.2514/3.58379.

[5] Toal, D. J., Bressloff, N. W., Keane, A. J., and Holden, C. M., "Geometric filtration using proper orthogonal decomposition for aerodynamic design optimization," *AIAA journal*, Vol. 48, No. 5, 2010, pp. 916–928.

[6] Sederberg, T. W., and Parry, S. R., "Free-Form Deformation of Solid Geometric Models," *SIGGRAPH Comput. Graph.*, Vol. 20, No. 4, 1986, p. 151–160. https://doi.org/10.1145/15886.15903, URL https://doi.org/10.1145/15886.15903.

[7] Yang, J., Hu, B., Tao, Y., and Li, J., "A flexible method for geometric design of axial compressor blades," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 0, No. 0, 2022, p. 09544100211063078. https://doi.org/10.1177/09544100211063078, URL https://doi.org/10.1177/09544100211063078.

[8] Poole, D. J., Allen, C. B., and Rendall, T., "Efficient Aero-Structural Wing Optimization Using Compact Aerofoil Decomposition," *AIAA Scitech 2019 Forum*, 2019. https://doi.org/10.2514/6.2019-1701, URL https://arc.aiaa.org/doi/abs/10.2514/6.2019-1701.

[9] Ghosh, S., Padmanabha, G., Peng, C., Andreoli, V., Atkinson, S., Pandita, P., Vandeputte, T., Zabaras, N., and Wang, L., "Inverse Aerodynamic Design of Gas Turbine Blades using Probabilistic Machine Learning," *Journal of Mechanical Design*, 2021, pp. 1–16. https://doi.org/10.1115/1.4052301.

[10] 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, edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[11] Kingma, D. P., and Welling, M., "Auto-Encoding Variational Bayes," *CoRR*, Vol. abs/1312.6114, 2014.

[12] Chen, W., Chiu, K., and Fuge, M., *Aerodynamic Design Optimization and Shape Exploration using Generative Adversarial Networks*, 2019. https://doi.org/10.2514/6.2019-2351, URL https://arc.aiaa.org/doi/abs/10.2514/6.2019-2351.

[13] Chen, W., and Ramamurthy, A., "Deep Generative Model for Efficient 3D Airfoil Parameterization and Generation," *AIAA Scitech 2021 Forum*, 2021. https://doi.org/10.2514/6.2021-1690, URL http://dx.doi.org/10.2514/6.2021-1690.

[14] Yilmaz, E., and German, B., "Conditional Generative Adversarial Network Framework for Airfoil Inverse Design," *AIAA Aviation Forum 2020*, 2020. https://doi.org/10.2514/6.2020-3185, URL https://arc.aiaa.org/doi/abs/10.2514/6.2020-3185.

[15] Du, X., He, P., and Martins, J. R. R. A., "A B-Spline-based Generative Adversarial Network Model for Fast Interactive Airfoil Aerodynamic Optimization," *AIAA Scitech 2020 Forum*, 2020. https://doi.org/10.2514/6.2020-2128, URL https://arc.aiaa.org/doi/abs/10.2514/6.2020-2128.

[16] Lin, J., Zhang, C., Xie, X., Shi, X., Xu, X., and Duan, Y., "CST-GANs: A Generative Adversarial Network Based on CST Parameterization for the Generation of Smooth Airfoils," *2022 IEEE International Conference on Unmanned Systems (ICUS)*, 2022, pp. 600–605. https://doi.org/10.1109/ICUS55513.2022.9987080.

[17] Wang, Y., Shimada, K., and Barati Farimani, A., "Airfoil GAN: encoding and synthesizing airfoils for aerodynamic shape optimization," *Journal of Computational Design and Engineering*, Vol. 10, No. 4, 2023, pp. 1350–1362. https://doi.org/10.1093/jcde/qwad046, URL https://doi.org/10.1093/jcde/qwad046.

[18] Rios, T., Van Stein, B., Bäck, T., Sendhoff, B., and Menzel, S., "Point2FFD: Learning Shape Representations of Simulation-Ready 3D Models for Engineering Design Optimization," *2021 International Conference on 3D Vision (3DV)*, 2021, pp. 1024–1033. https://doi.org/10.1109/3DV53792.2021.00110.

[19] Wei, Z., Yang, A., Li, J., Bauerheim, M., Liem, R. P., and Fua, P., "DeepGeo: Deep Geometric Mapping for Automated and Effective Parameterization in Aerodynamic Shape Optimization," *AIAA Aviation Forum and Ascend 2024*, 2024. https://doi.org/10.2514/6.2024-3839, URL https://arc.aiaa.org/doi/abs/10.2514/6.2024-3839.

[20] Makhzani, A., Shlens, J., Jaitly, N., and Goodfellow, I., "Adversarial Autoencoders," *International Conference on Learning Representations*, 2016.

[21] Davies, T., Nowrouzezahrai, D., and Jacobson, A., "On the Effectiveness of Weight-Encoded Neural Implicit 3D Shapes," *International Conference on Machine Learning (ICML)*, PMLR, 2021. URL http://arxiv.org/abs/2009.09808v3.

[22] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S., "Deepsdf: Learning continuous signed distance functions for shape representation," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 165–174.

[23] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A., "Occupancy networks: Learning 3d reconstruction in function space," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4460–4470.

[24] Dupont, E., Kim, H., Eslami, S. M. A., Rezende, D. J., and Rosenbaum, D., "From data to functa: Your data point is a function and you can treat it like one," *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 162, edited by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, PMLR, 2022, pp. 5694–5725. URL https://proceedings.mlr.press/v162/dupont22a.html.

[25] Bamford, T., Keane, A., and Toal, D., "SDF-GAN: Aerofoil Shape Parameterisation via an Adversarial Auto-Encoder," *AIAA Aviation Forum and Ascend 2024*, 2024. https://doi.org/10.2514/6.2024-3665.

[26] Bamford, T., Toal, D., and Keane, A., "AeroINR: Meta-learning for Efficient Generation of Aerodynamic Geometries," *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2024, Vilnius, Lithuania, September 9–13, 2024, Proceedings, Part IX*, Springer-Verlag, Berlin, Heidelberg, 2024, p. 452–467. https://doi.org/10.1007/978-3-031-70378-2_28.

[27] Guendelman, E., Bridson, R., and Fedkiw, R., "Nonconvex Rigid Bodies with Stacking," *ACM Transactions on Graphics*, Vol. 22, No. 3, 2003, p. 871–878. https://doi.org/10.1145/882262.882358.

[28] Pu, J., Zheng, B., Leader, J. K., Wang, X.-H., and Gur, D., "An automated CT based lung nodule detection scheme using geometric analysis of signed distance field," *Medical physics*, Vol. 35, No. 8, 2008, pp. 3453–3461. https://doi.org/10.1118/1.2948349.

[29] Perry, R. N., and Frisken, S. F., "Kizamu: A System for Sculpting Digital Characters," *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, Association for Computing Machinery, New York, NY, USA, 2001, p. 47–56. https://doi.org/10.1145/383259.383264.

[30] Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., and Kaushik, S., "Prediction of aerodynamic flow fields using convolutional neural networks," *Computational Mechanics*, Vol. 64, No. 2, 2019, pp. 525–545. https://doi.org/10.1007/s00466-019-01740-0.

[31] Yang, S., Lee, S., and Yee, K., "Inverse design optimization framework via a two-step deep learning approach: application to a wind turbine airfoil," *Engineering with Computers*, Vol. 39, No. 3, 2023, pp. 2239–2255. https://doi.org/10.1007/s00366-022-01617-6.

[32] 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. https://doi.org/10.1007/s00158-021-02851-0.

[33] Wang, J., Li, R., He, C., Chen, H., Cheng, R., Zhai, C., 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. https://doi.org/10.1016/j.cja.2021.03.006.

[34] Wang, Z., She, Q., and Ward, T. E., "Generative Adversarial Networks in Computer Vision: A Survey and Taxonomy," *ACM Computing Surveys*, Vol. 54, No. 2, 2021. https://doi.org/10.1145/3439723.

[35] Li, J., Zhang, M., Martins, J., and Shu, C., "Efficient Aerodynamic Shape Optimization with Deep-Learning-Based Geometric Filtering," *AIAA Journal*, Vol. 58, No. 10, 2020, pp. 4243–4259. https://doi.org/10.2514/1.J059254.

[36] Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O., "Autoencoding beyond pixels using a learned similarity metric," *Proceedings of The 33rd International Conference on Machine Learning*, Vol. 48, 2016, pp. 1558–1566.

[37] Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A., "Adversarially Learned Inference," *arXiv*, 2017. https://doi.org/10.48550/arXiv.1606.00704.

[38] Ulyanov, D., Vedaldi, A., and Lempitsky, V., "It takes (only) two: adversarial generator-encoder networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32, 2018. https://doi.org/10.1609/aaai.v32i1.11449.

[39] Donahue, J., Krähenbühl, P., and Darrell, T., "Adversarial Feature Learning," *International Conference on Learning Representations*, 2016.

[40] Rosca, M., Lakshminarayanan, B., Warde-Farley, D., and Mohamed, S., "Variational Approaches for Auto-Encoding Generative Adversarial Networks," *arXiv*, 2017. https://doi.org/10.48550/arXiv.1706.04987.

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

[42] Sharpe, P., and Hansman, R. J., *Core Components of an Optimization Framework for Engineering Systems based on Automatic Differentiation*, ???? https://doi.org/10.2514/6.2022-3937, URL https://arc.aiaa.org/doi/abs/10.2514/6.2022-3937.

[43] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D., "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Comput.*, Vol. 1, No. 4, 1989, p. 541–551. https://doi.org/10.1162/neco.1989.1.4.541, URL https://doi.org/10.1162/neco.1989.1.4.541.

[44] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R., "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *ECCV*, 2020.

[45] Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G., "Implicit Neural Representations with Periodic Activation Functions," *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Curran Associates, Inc., 2020, pp. 7462–7473. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf.

[46] Chen, Z., and Zhang, H., "Learning Implicit Fields for Generative Shape Modeling," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[47] Duvall, J., Duraisamy, K., and Pan, S., "Discretization-independent surrogate modeling over complex geometries using hypernetworks and implicit representations," , 2022. URL https://arxiv.org/abs/2109.07018.

[48] Duvall, J., Joly, M., Duraisamy, K., and Sarkar, S., "Design-Variable Hypernetworks for Flowfield Emulation and Shape Optimization of Compressor Airfoils," *AIAA Journal*, Vol. 62, No. 2, 2024, pp. 741–757. https://doi.org/10.2514/1.J063156, URL https://doi.org/10.2514/1.J063156.

[49] Pan, S., Brunton, S. L., and Kutz, J. N., "Neural Implicit Flow: a mesh-agnostic dimensionality reduction paradigm of spatio-temporal data," *Journal of Machine Learning Research*, Vol. 24, No. 41, 2023, pp. 1–60. URL http://jmlr.org/papers/v24/22-0365.html.

[50] Li, R., Zhang, Y., and Chen, H., "Mesh-Agnostic Decoders for Supercritical Airfoil Prediction and Inverse Design," *AIAA Journal*, Vol. 62, No. 6, 2024, pp. 2144–2160. https://doi.org/10.2514/1.J063387.

[51] Ha, D., Dai, A. M., and Le, Q. V., "HyperNetworks," *International Conference on Learning Representations*, 2017.

[52] Zhao, D., Kobayashi, S., Sacramento, J., and von Oswald, J., "Meta-Learning via Hypernetworks," *4th Workshop on Meta-Learning at NeurIPS 2020 (MetaLearn 2020)*, IEEE, 2020. URL https://doi.org/10.5167/uzh-200298.

[53] Alaluf, Y., Tov, O., Mokady, R., Gal, R., and Bermano, A., "HyperStyle: StyleGAN Inversion With HyperNetworks for Real Image Editing," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 18511–18521.

[54] Selig, M., *UIUC Airfoil Coordinates Database*, 1996. URL https://m-selig.ae.illinois.edu/ads/coord_database.html, (Accessed April 2024).

[55] Masters, D. A., Taylor, N. J., Rendall, T. C. S., Allen, C. B., and Poole, D. J., "Geometric Comparison of Aerofoil Shape Parameterization Methods," *AIAA Journal*, Vol. 55, No. 5, 2017, pp. 1575–1589. https://doi.org/10.2514/1.J054943.

[56] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P., "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets," *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, p. 2180–2188.

[57] Chen, Q., Pope, P., and Fuge, M., "Learning Airfoil Manifolds with Optimal Transport," *AIAA SCITECH 2022 Forum*, 2022. https://doi.org/10.2514/6.2022-2352.

[58] 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. https://doi.org/10.2514/1.J059317.

[59] Grey, Z. J., Doronina, O. A., and Glaws, A., "Separable shape tensors for aerodynamic design," *Journal of Computational Design and Engineering*, Vol. 10, No. 1, 2023, pp. 468–487. https://doi.org/10.1093/jcde/qwac140.

[60] Dawson-Haggerty et al., "trimesh," , ???? URL https://trimesh.org/.

[61] Sharpe, P. D., "Accelerating Practical Engineering Design Optimization with Computational Graph Transformations," Ph.D. thesis, Massachusetts Institute of Technology, 2024.

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

[63] Zamorski, M., Zięba, M., Klukowski, P., Nowak, R., Kurach, K., Stokowiec, W., and Trzciński, T., "Adversarial autoencoders for compact representations of 3D point clouds," *Computer Vision and Image Understanding*, Vol. 193, 2020, p. 102921. https://doi.org/https://doi.org/10.1016/j.cviu.2020.102921, URL https://www.sciencedirect.com/science/article/pii/S107731422030014X.

[64] Qi, C. R., Su, H., Mo, K., and Guibas, L. J., "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.