Embedded Parameter Information in Conditional Generative Adversarial Networks for Compressor Airfoil Design

Andy J. Keane* and Ivan I. Voutchkov†
University of Southampton, Southampton, England SO17 1BJ, United Kingdom

https://doi.org/10.2514/1.J061544

The use of conditional generative adversarial networks has become popular with the advancement of computer power, and in particular graphics processing units. Large hardware and software companies such as Google, IBM, and Facebook have been experimenting in this field for over a decade for facial recognition, image classification and pattern recognition, and deep fake facial and object design. One essential key to their success is access to large quantities of tagged and classified images, with millions of images typically being used. In contrast, relatively few similar advances have been seen in the engineering sector, mainly because engineering analyses that produce suitable images are often very expensive processes that absorb a considerable amount of effort to generate. In addition, feeding synthetically generated image data back into traditional engineering workflows is not entirely straightforward. In this paper we show how adversarial networks can be used if order 10^4 images are available and focused on the problem in hand. In particular we show how such images can then be augmented with histograms and glyphs to enhance the image content with pictorial representations of numerical data. This is shown to significantly assist the network training process on our data when used in contexts where numerical data categorizing individual images are available and numerical performance measures of products must be predicted. Crucially, it allows for follow-on analyses without the need to use artificially generated flow fields.

I. Introduction

IN [1] we discussed a study in which a simple two-dimensional gas-turbine aerodynamic section was varied using 10 shape variables and 16 noise variables to produce many varying cross sections, which were subsequently run through a commercial-grade computational fluid dynamics (CFD) Reynolds-averaged Navier–Stokes (RANS) solver to extract pressure fields and pressure loss values. We demonstrated the use of various surrogate models to support robust design optimization problems in this area. We found and kriging and cokriging to be the most accurate for smaller amounts of data and shallow neural networks to be the best tool to use with more than 10,000 data points, being mainly driven by the high computational costs of training krigs with large data sets. Run times to train models on 70,000 results ranged from 118 s using shallow neural nets to 2350 h using Gaussian process regression krigs.

At present, most engineering design workflows start with a parametric geometry model (often based in large commercial computer-aided design [CAD] systems). The geometry models are typically meshed and then submitted for fluid dynamic or structural analyses to establish likely product performance. Such outputs are routinely used with various classes of optimization software to drive performance improvement. In such workflows, designing and implementing the parameterization plays a crucial role and is often the most laborious step in setting up the process. It often requires very skilled and intricate work to program a CAD engine to generate a robust system capable of creating the desired geometry for each new set of parameters. In many cases parameterizations will fail for various parameter combinations due to various interactions that would produce an infeasible geometry.

Even if the CAD is programmed to perfection and all possible conditions that create geometrical conflicts are foreseen and dealt with, the final geometry will always be defined by the sophistication of the parameterization, which will be finite in its scope—if too few parameters are used, only relatively narrow sets of geometry are possible, whereas if too many are deployed, the search space to be studied becomes impossibly large. This leads to two fundamental issues. First, there may be designs that will never be explored because of the tightly defined nature of the initial parameterization or the size of the space created. Secondly, workflows designed with one parametric study in mind are often not easy or impossible to reuse for other studies, where a fresh parameterization is required. As a result, research and development teams end up with large amounts of data that are similar but not parametrically matched, and therefore the common practice is that when a new study is required and parameters are added or changed in definition, the entire design search process starts from the beginning, discarding potentially useful information.

One type of output that researchers commonly generate, along with various numerical files, is images such as those of flow and stress fields. Often they are intended to allow the rapid appraisal of results or to include in reports and presentations. These images contain a lot of information, essentially encoded in pixels. At the same time such images can often show strong similarities between studies of related problems. For example, the pressure field around a section of a compressor blade can be very similar to the pressure field around other turbine blades or any other airfoil. This creates the opportunity to reuse images from different studies and even resolutions, by training convolution neural networks (CNNs). In this paper we aim to utilize such data in the form of image information arising from the earlier cited CFD study to create conditional generative adversarial networks (cGANs). Note that cGANS are a newly emerging deep learning approach that can both classify and generate designs based on CNN-based image processing; see, e.g., [2–8]. These papers illustrate the rapidly emerging field of imaged-based approaches to aerodynamics. In this work we address two key aspects of the use of cGANs in engineering design: first, if a new design is generated in the form of an image with associated flow-field, how do we accurately extract numerical performance data from the image, and, secondly, if we wish to validate the cGAN-generated results or perform other analyses, how do we accurately extract suitably accurate shape information for input to other engineering software? We also confirm that, given GPU processing, cGANs can be trained in similar times to the surrogates considered previously.

Received 15 December 2021; revision received 11 July 2022; accepted for publication 18 July 2022; published online 1 August 2022.

*Professor of Computational Engineering, Aeronautics and Astronautics.
†Senior Research Fellow, Aeronautics and Astronautics.

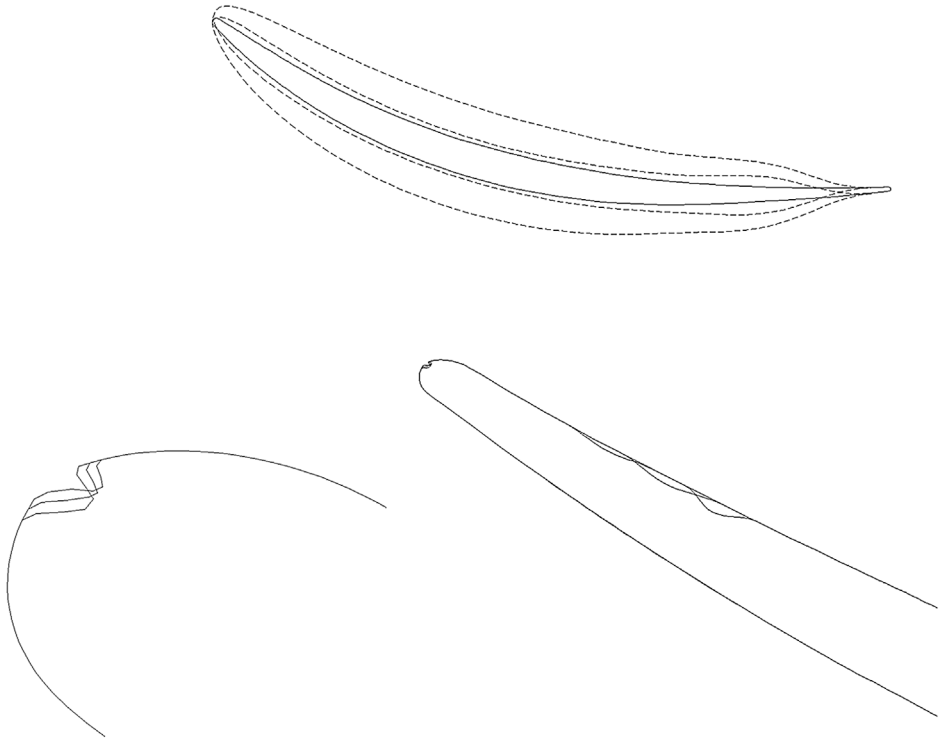

Fig. 1 Maximum extent of overall geometry variation (upper), leading-edge damage (lower left), flank erosion (lower right).

II. Airfoil Images

To begin this study we start from the base design of a gas turbine compressor airfoil section, where we use a series of Hicks–Henne functions[‡] [9] to alter its overall shape, using 10 design variables (as in Fig. 1, upper). We additionally use 16 noise variables to represent manufacturing, damage, and erosion effects (as in Fig. 1, lower). The noise variables make three sets of changes to the baseline airfoil that attempt to represent modeling of manufacturing uncertainty, modeling of foreign object damage, and modeling of flank erosion: 1) The manufacturing variability changes use multiple Hicks–Henne functions that affect the entire section, but paying particular attention to the leading edge. 2) The foreign object damage changes are localized to the leading edge using a single Hicks–Henne function that can only remove material, but which can vary in position and shape as well as depth. 3) The erosion changes are localized to the flank, again using a single Hicks–Henne function that can vary in position and shape as well as depth and can only remove material. These changes are all carried out using the proprietary Rolls-Royce Parametric Design and Rapid Meshing (Padram) code [10].

The Padram geometry modification process also permits the automated adaptation of high-quality OCH meshes around the airfoil section in a setting appropriate for application to gas-turbine blade design (here "OCH" refers to the shapes of the various mesh blocks, their being ring like, cup shaped, or with legs). These are distorted around the localized erosion geometries introduced when studying leading-edge damage or flank erosion. Here the "O" mesh is four cells deep and slightly over 27,000 cells were used in total. The topology of this mesh was unchanged throughout the image generation process so that any effects caused by re-meshing were avoided.

Having set up the mesh, a full steady-state RANS flow analysis can be carried out using the parallel version of the Rolls-Royce Hydra RANS CFD code in around 2 minutes on a 12-core personal computer [11]. The section is run in isolation with boundary conditions of inlet temperature 290 K, inlet total pressure 63,400 Pa, whirl angle −37.3 deg, and outlet static pressure 52,000 Pa. Typical root-mean-square (RMS) flow residuals are 4.7E-9 and RMS turbulence residuals 2.4E-15. Compressor designers use a variety of total pressure loss measures such as mixed out (MO) average total pressure loss, mass averaged total pressure loss, and total Denton loss; see, e.g., Cumpsty [12]. Here we adopt the MO average pressure loss as the main performance metric to categorize the pressure flowfield images, normalized by dividing by that for the datum design to preserve confidentiality.[§] This formulation averages the area- and mass-based losses and is found to best represent the overall design performance. Figure 2 illustrates the variation of normalized MO pressure loss seen during these studies, while Fig. 3 (left) shows a typical flowfield.

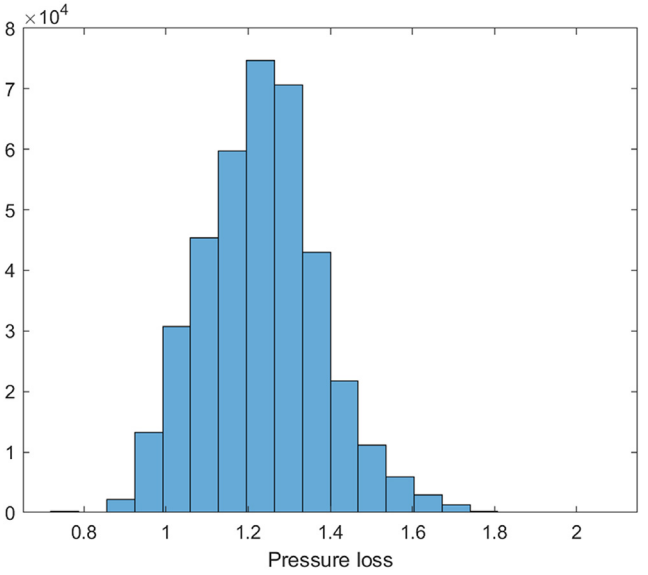

Fig. 2 A histogram showing the variation of normalized mixed out total pressure loss seen across the data set used in this work.

[‡]Note that $z = z_{\text{initial}} + \sum_{i=1}^{n} a_i \sin^{t_i}(\pi x^{\ell_n(0.5)/\ell_n(h_i)})$ for i Hicks–Henne functions of amplitude a_i, width t_i, and nondimensional location h_i wrapped around the initial shape z.

[§]A full description of the procedure needed to calculate the MO pressure loss lies outside the scope of this paper, but see Prasad [13] for details of the required calculations.

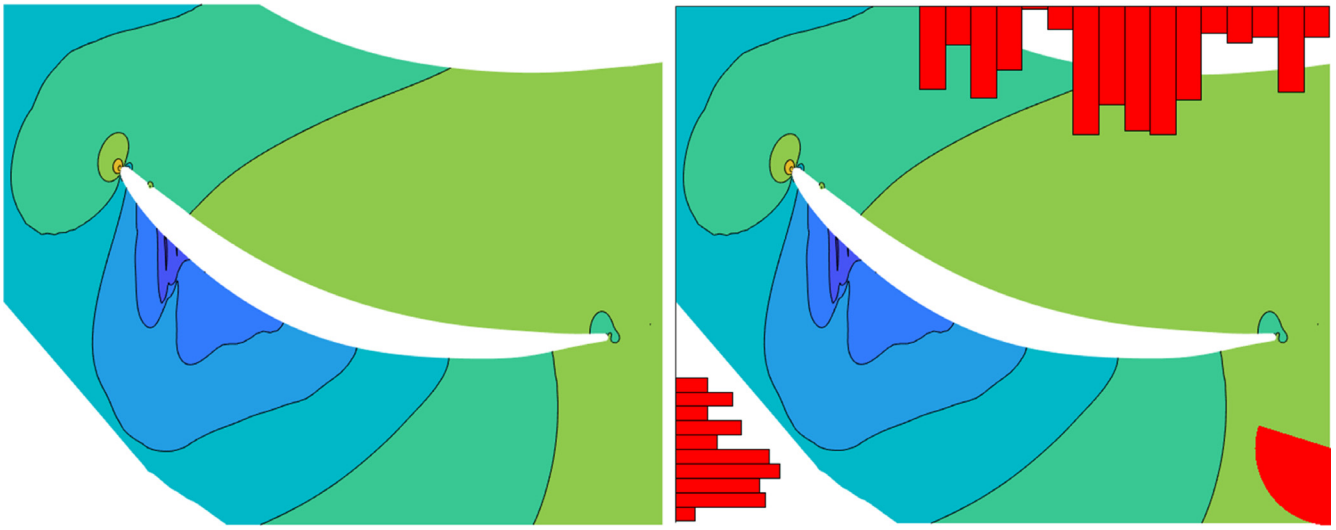

Fig. 3 Left: An image produced by the Rolls-Royce plc CFD (Hydra) code. Right: Pressure field with bars for the design (lower left) and noise (upper right) variables and a half-sector glyph for the mixed out total pressure loss value (specific variable and contour values are not provided as these results are industrially sensitive).

This design problem has been extensively studied in a series of previous papers, where full details of the geometry definition, meshing, and CFD processes can be found [14–17]. At the start of the present work we therefore had access to a large quantity of results files that we could draw on.

We have to-date accumulated over 370,000 images of this type, where the shape of the airfoil varies with each design. Due to memory limitations on the GPU devices we use for training (high-end gaming GPU cards—NVIDIA RTX 3080), we need to scale these images down to 256×256 pixels or less for convolution network training because, even at this limit, the mini-batch size used in training can hold only 47 images on these GPUs. Higher resolution images lead to smaller batches and then the batch renormalization statistics become too noisy, giving rise to fluctuations that swamp the training process. This gives rise to the fundamental problems noted above: at resolutions suitable for CNN training, the variations of the shapes in the airfoil cannot be read back from the generated image files with sufficient accuracy for direct CFD analysis given the subtle changes being investigated. Rolls-Royce typically defines blade geometry to 0.1 mm accuracy or better, and for the blades in question this would require a resolution of at least 2048×2048 pixels. In addition, it is very difficult to impute section performance directly from a pressure field without knowing a great many other details of the local flowfield. Thus even if a cGAN can generate a likely design and flowfield, with current generations of hardware, one is not able to simply assess its quality—sections pressure losses cannot be read directly from the images and the shape cannot be recovered sufficiently accurately to compute such quantities using CFD.

To tackle these problems, we have enriched our images by overlaying histograms and half-sector glyphs on them, as in Fig. 3 (right). Here, each bar represents a scaled design or noise variable value and the half-sector glyph represents the value of the section pressure loss, rather like a car fuel gauge. This overlay has been applied to all our images to provide a database, ready for CNN training, placing the added material in areas where either there is no other data or the pressure field variation is very small between images.[¶] The images are also tagged into 10 groups corresponding to their normalized pressure loss values as computed during initial creation—category 1 being the lowest (best) loss and category 9 being the highest loss. Category 10 is used for infeasible designs where the value of computed loss is unrealistically high but a flowfield has still been computed, which is generally caused by meshing or convergence problems. The width of each category is chosen so that each of the first nine contains a minimum of 10,000 designs in our database. These overlays serve two purposes: First, they provide extra information to the cGAN during training by providing additional discrimination between similar images. Secondly, when the cGAN generates a new image it allows the design and noise parameters to be readily extracted from the image, and these can then either be used directly or inserted into the original CFD process to create a fully accurate flowfield for the extracted parameter set.

III. Conditional Generative Adversarial Networks

Generative adversarial networks are a deep learning image processing architecture that combines two independent CNN networks that are trained against each other. They allow the generation of new images based on, and similar to, a training set. *Conditional* generative adversarial networks (as illustrated in Fig. 4), in addition, allow the user to steer the output so that generated images tend to lie in particular regions of the input image set as defined by a category variable. They consist of a discriminator, which is trained to classify if an image belongs to a predefined category (In essence, is it a fake or a true image, and does it belong to the required category?), and a generator, which is trained to generate an image of a particular category [18]. The exact image generated depends on both the desired category used to steer the generator and a set of latent variables used to provide variety in its outputs. During training, these latent variables are generally set as random numbers with a normal distribution so that a wide range of possible images are considered.[**]

Each of the networks contains a number of neural layers, as described in the MATLAB tutorial and illustrated in Fig. 5 with settings as detailed in Table 1, as recommended in reference [18]. The training process consists of updating the weight and biases of each neuron in these layers using an ADAMS optimizer that attempts to minimize a predefined loss function [19]. As noted in [20]:

> The discriminator learns to classify input images as "real" or "generated". The output of the discriminator corresponds to a probability $\hat{Y}$ that the input images belong to the class "real". The generator score is the mean of the probabilities corresponding to the discriminator output for the generated images:

$$\text{score}_{\text{Gen}} = \text{mean}(\hat{Y}_{\text{Gen}})$$

[¶]Note that cGANs are based on convolution methods that are designed to deal with two-dimensional image data rather than the vector structures used in traditional surrogate modeling approaches, and this is why the additional data are embedded in image format.

[**]See https://arxiv.org/abs/1701.00160 for an excellent tutorial introduction to GANs and their underlying mechanics.

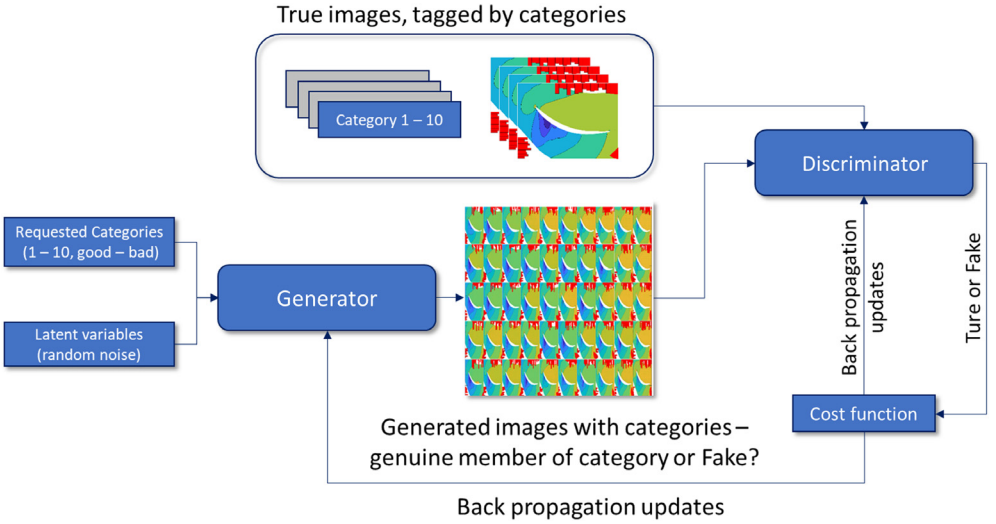

Fig. 4 Conditional generative adversarial network (cGAN).

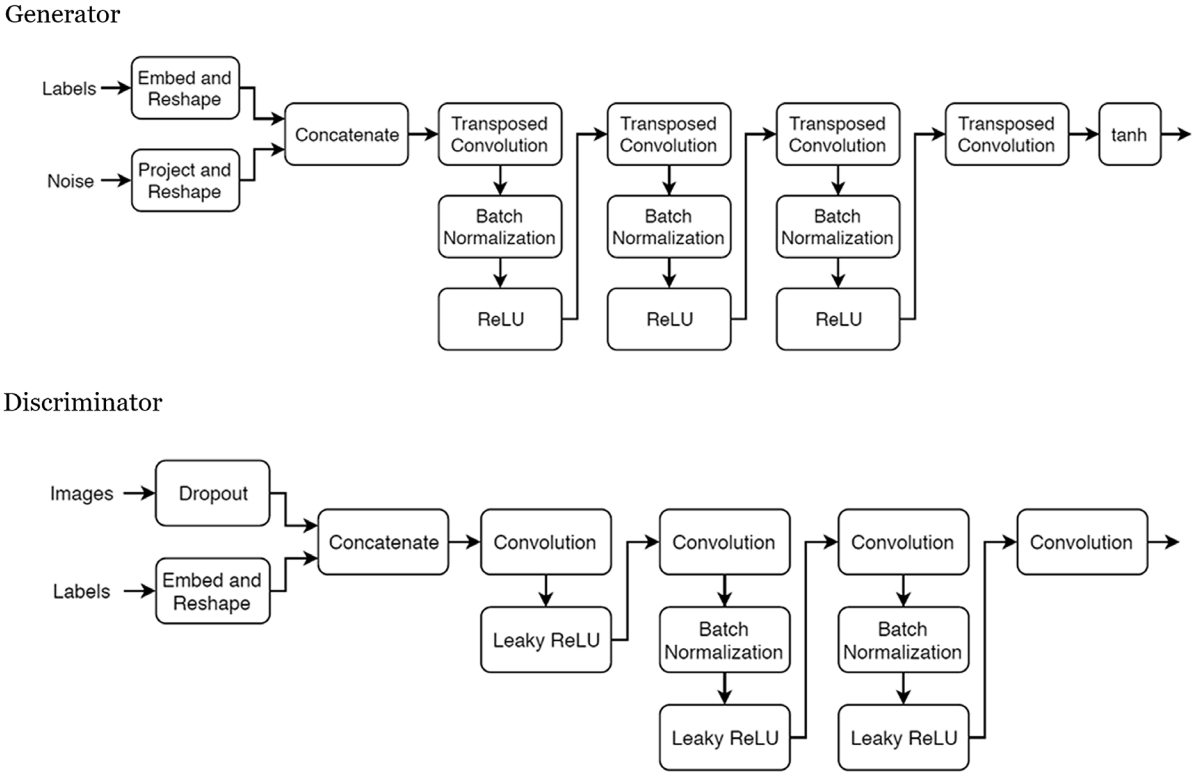

Fig. 5 Network design of cGAN; see Train Conditional Generative Adversarial Network (CGAN)–MATLAB & Simulink–MathWorks United Kingdom.

where $\hat{Y}_{\text{Gen}}$ contains the probabilities for the generated images. Given that $1 - \hat{Y}$ is the probability of an image belonging to the class "generated", the discriminator score is the mean of the probabilities of the input images belonging to the correct class:

$$\text{score}_{\text{Disc}} = \frac{1}{2}\text{mean}(\hat{Y}_{\text{Real}}) + \frac{1}{2}\text{mean}(1 - \hat{Y}_{\text{Gen}})$$

where $\hat{Y}_{\text{Real}}$ contains the discriminator output probabilities for the real images and the numbers of real and generated images passed to the discriminator are equal.

In our work the discriminator loss is given by $\text{loss}_{\text{Disc}} = -(\text{mean}(\ell_n(0.9 \times \text{score}_{\text{Real}})) + \text{mean}(\ell_n(1 - \text{score}_{\text{Gen}}))$ and the generator loss by $\text{loss}_{\text{Gen}} = -\text{mean}(\ell_n(\text{score}_{\text{Gen}}))$.

There are a number of learning parameters that can be adjusted to help the learning process, which also vary with the size of the images the network is being trained on (see Table 1). This training process can be slow and is generally only feasible if a powerful GPU is used. For this study we have used NVIDIA cards. At each epoch the network is presented with all available images but they are not loaded into memory all at once, being handled in smaller batches. The size of the batches is determined by the amount of GPU memory available on the card in use. Smaller batch sizes require a higher number of updates and eventually lead to instability in calculating renormalization statistics, which leads to instability in the training process. When using 128×128 images the maximum batch size possible on our GPUs is 84, whereas if we increase the image size to 256×256 it decreases to 47. With an image size of 512×512 this reduces down to a batch size of just two, which is completely unstable. In future work we intend to study image sizes that are not square and in powers of two, but this will involve further code development. In our case,

Table 1 Conditional generative adversarial network parameter settings

Image size	128 × 128	256 × 256
Number of classes	10	10
Number of latent inputs	20	20
Mini batch size	84	47
Number of filters	96	96
Embedding dimension	50	50
Projection size	[12,12,1024]	[28,28,1024]
Filter size	5	5
Stride generator	[1,2,2,2]	[1,2,2,2]
Stride discriminator	[2,2,2,2]	[2,2,2,2]
Learn rate discriminator	0.0001	0.0001
Learn rate generator	0.0002	0.0002
Gradient decay factor	0.5	0.5
Flip factor	0.5	0.5
Squared gradient decay factor	0.999	0.999

typical training times are 300 h if the full set of images is used at a resolution of 256 × 256. We have found, however, that good results can be obtained with as few as 20,000 images in the training set, provided that they are well chosen to represent the range of performance categories being studied and then training can be completed in 70 h. By way of comparison, in our previous study [1] Gaussian process regression models took between 650 and 2350 h to train on data sets containing 70,000 CFD runs.

It is not possible to validate the network directly on a predefined validation set, because the output is random, due to the nature of the latent space. Instead, our validation process is based on generating 200 images from the first three categories and 100 for the remaining categories of the cGAN being tested, then extracting the design parameters from the embedded histograms to create a full blade geometry that can be analyzed through the CFD process that generated the original training set of 20,000. The values of pressure loss extracted from the cGAN-generated image are then compared with the freshly computed CFD value.

The different learning rates for the discriminator and the generator can also have significant impact on the training process, as well as the definition of the loss and gradient functions, such as by dampening the score of the discriminator such that it always is a bit unsure of its classification. This can be used to prevent the discriminator learning faster than the generator. A good balance, however, takes some care to achieve, especially with varying number of neurons in the layers, which is determined by the image sizes, the stride, number of filters, the filter size, the number of latent variables, etc., detailed in Table 1. We have observed, however, that the score values for the generator and the discriminator can become divergent, without significant detrimental impact on the quality of the final generated images (see Fig. 6). The subtleties of choosing these and other tuning factors will be subject of future publications.

IV. Validation and Parameter Extraction

At each point during training, the designs generated by the networks can be tested and validated against CFD runs. The images shown in Fig. 7 illustrate the following sequence: First, a trained generator network is used to create an image of category 1 (best

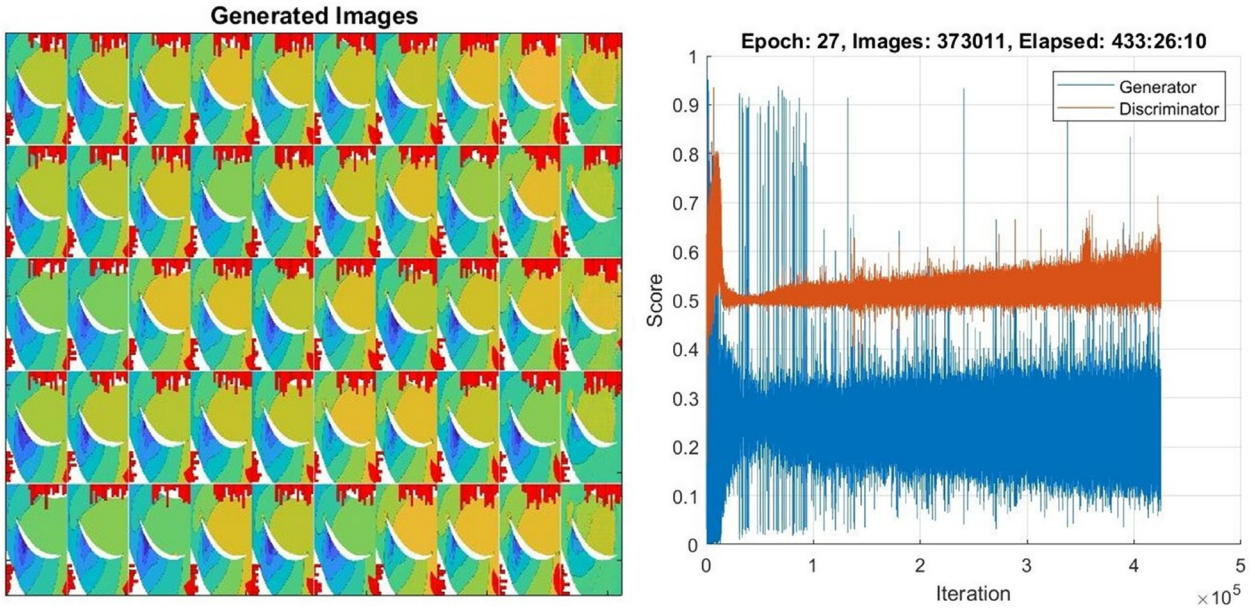

Fig. 6 Training of cGAN, showing typical generated images and the training history after 27 epochs using all 373,011 images in the data set (the total elapsed training time being 433 h).

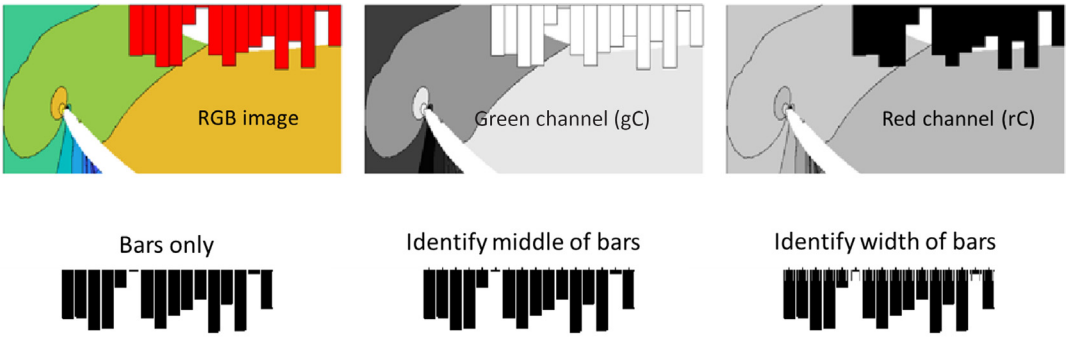

Fig. 7 Reading the areas of the bars in pixels, by separating the color channels to extract the bars and then identifying the areas, midpoints, and widths.

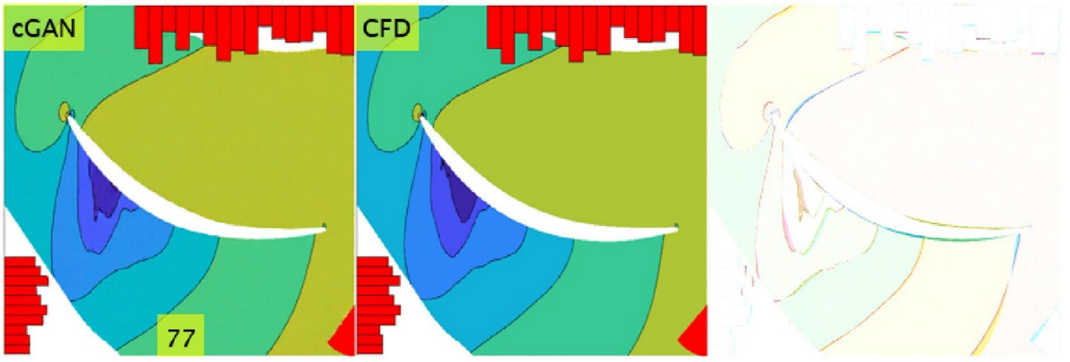

Fig. 8 Comparison of cGAN- vs CFD-generated images and difference plot. (Note: Specific variable and contour values are not provided as these results are industrially sensitive; also the image resolution shown here is that used in the cGAN process and is thus deliberately more pixilated than would normally be used in journal paper publications.)

pressure loss values). Because the network is trained on images that contain histograms and sector glyphs, these are also present in the generated image. The number of pixels representing the 10 shape variables and the 16 noise variables are extracted from the red bars and converted to real values, as shown in the figure. Color images consist of red, green, and blue channels, and these can be readily separated. With some subtraction and filtering operations, the red color of the histograms can be separated, and the number of pixels representing the length of a bar extracted: this Boolean filtering operation is illustrated in Fig. 7, where gC and rC are the matrices for the green and the red channels, respectively. This process and its accuracy are addressed shortly.

The extracted values are then used to reconstruct the shape of the airfoil parametrically, following the original 26-parameter-based method detailed earlier, to construct a mesh and finally run the CFD, which results in a new pressure loss value and pressure field. An example is shown alongside the original image in Fig. 8. Crucially the parameter encoding obviates the need to try and interpret the airfoil recorded in the image directly and allows the shape required for analysis to be constructed to the same precision as the originals used in the training step. Note that the histogram bars are essentially identical between the two images because they represent the same values (the very slight differences are due to the pixel resolution used to encode real values), but the sector glyph representing the resultant loss values differs. In this study we do not need a quantified metric for the accuracy of the flowfield image because new designs are created using the embedded data bars and then validated by subsequent CFD analysis—the whole purpose of the present study is to escape from the need for highly accurate flowfield predictions. However, the two images may be subtracted from each other, and here the difference is displayed in the third image of Fig. 8. Although the magnitudes of the differences are not given here to preserve the confidentiality of the data, this figure allows the reader to see where differences occur. White areas represent no difference. Similar differences are found across the whole range of cGAN-generated images.

It takes only few seconds to generate large numbers of images from the cGAN generator, even without using a GPU. This is useful to know, as the process only requires a GPU to train the network, but not to exploit it, allowing a trained network to be deployed on a wide range of devices—lower-end desktop computers, laptops, or tablets, giving a designer the ability to quickly predict what the CFD pressure field might look like for various designs, without running the actual CFD. This is made possible because the section performance is also directly available from the generated images by counting the pixels in the sector glyph.

Following the process described above, and to assess the accuracy of the cGAN and its embedded information, we have generated 1200 images—200 for the first three categories and 100 for each of the rest, using a uniform $[-1; 1]$ latent variable range, and analyzed their performance using CFD. In Fig. 9 we plot the pressure loss versus run number—the categories are easily recognizable, showing the ability of the network to recreate images for a given category and the strong correlation of the histograms to the desired parameter settings. The dashed lines in the figure show the width of each category. Circled designs are those that have cGAN-predicted losses within 2% of the actual CFD result, providing a visual illustration of the most accurately generated designs. It is, of course, difficult to achieve designs that have very low loss values, and as the number of designs in the initial database with very low loss was not large, the width of the first category has to be set wider to include sufficient points.

Figure 10 (left) shows the accuracy of the loss values indicated by the generated pressure loss glyph by plotting these losses against those computed by CFD using the design and noise variables extracted from the histograms. The R^2 correlation between cGAN and CFD values comes to 0.924—such a strong correlation allows the user to have confidence that generated designs will live up to their predictions when subject to further analysis. Note that the accuracy of this prediction is of course impacted by the resolution of the generated images and the ability to read the values of the design and noise variables accurately—we examine this aspect shortly.

It is interesting to also note that there are a good number of CFD-generated pressure loss values that are better than the cGAN-predicted equivalents. These are the points below the 45° line, which we have then used to further enrich the data pool and subsequently populate category 1 with more designs allowing us to reduce the width of this category while still containing a large number of images and make the network prediction more accurately in this crucial region—a form of design optimization and self-learning that has been popular in other fields of artificial intelligence (AI).

A. Effect of Latent Space

The cGAN networks used here have been trained using latent variables with a simple normal distribution as is recommended for such work in [18]—this leads to values lying in the range ± 4.3. The subsequent designs studied in Figs. 9 and 10 (left) were obtained using latent variables with a uniform random distribution in the range of ± 1 to capture the peak of the bell-shaped curve of possible outputs. This follows several experiments where various ranges and distributions were used while trying to focus on the best possible outputs. Figure 10 (right) shows the effect of increasing this latent variable range. Wider ranges produced a wider spread of cGAN-generated pressure loss values and even greater range of calculated CFD loss values. In our experiments, the use of a uniform ± 1 latent variable range provided a good balance between exploitation and exploration, increasing the percentage of CFD better than cGAN designs from 38 to 45%. Due to the random nature of the latent space, the spread of the values is expected and even desirable. The aim of this study was to see if we can train a network that can produce low-pressure loss designs. Clearly, the spread of the designs can be controlled to some extent by restricting or relaxing the bounds of the random latent variables.

Reducing the range even further to ± 0.03 increased the percentage of designs with improved loss to 74% as shown in Fig. 11 (left). We have also used a genetic algorithm optimizer (GA) to manipulate the latent variables to minimize the cGAN pressure loss prediction. This also results in narrowed bands for the cGAN prediction as seen in Fig. 11 (right). Note that although reducing the width of the latent

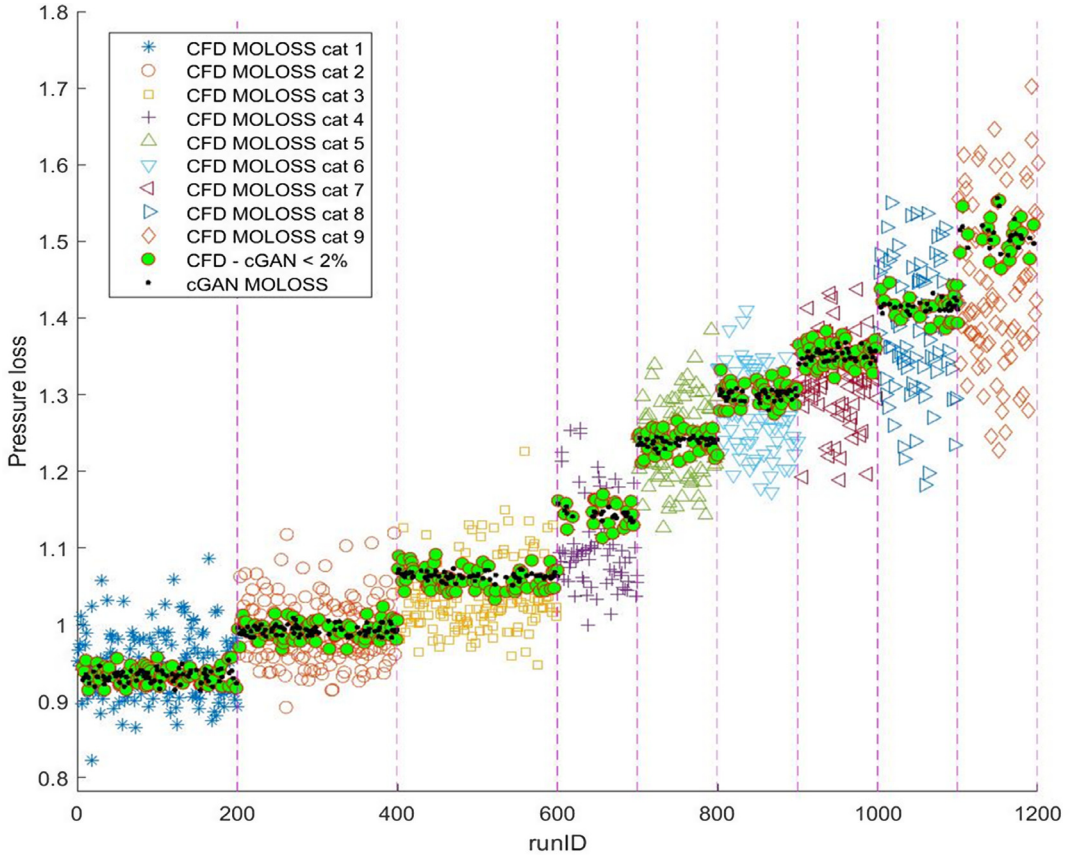

Fig. 9 The cGAN-predicted and CFD-calculated normalized mixed out total pressure loss values (loss versus run ID and bin class—circled designs lie within 2% of the predicted loss).

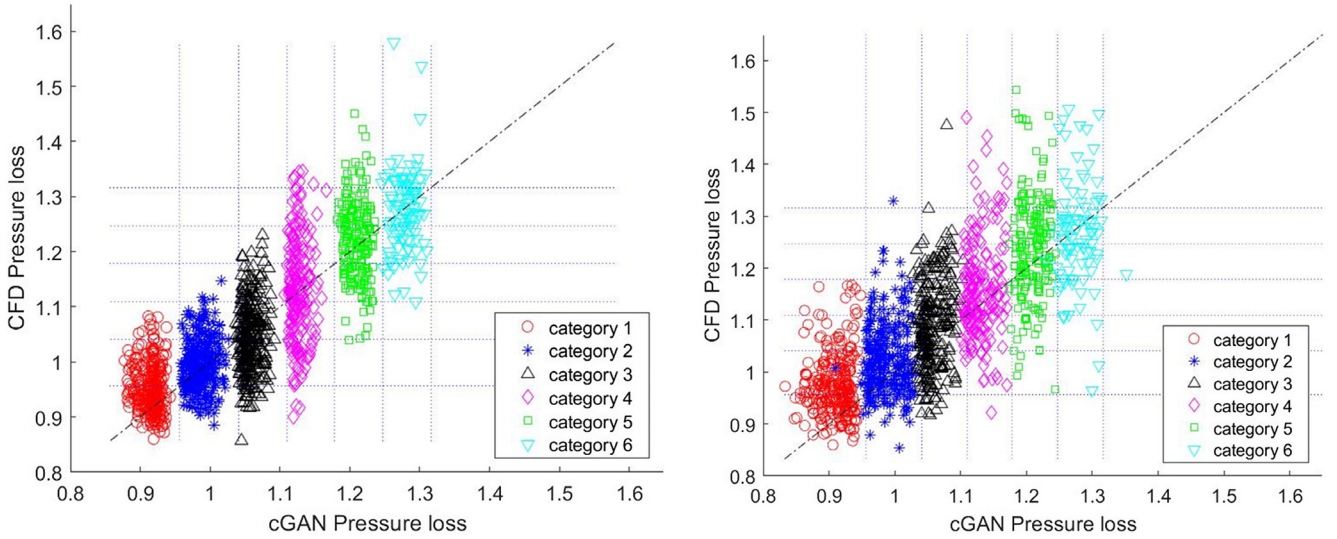

Fig. 10 The effect of latent space sampling: (left) [−1: 1]; (right) [−4.3: 4.3]; the dotted lines are the 45° lines of perfect correlation.

variable space when sampling for new designs necessarily reduces the range of predicted loss values, the equivalent CFD loss value ranges cannot be controlled anything like so well due to the inherent limitations of the cGAN models being built—the use of higher resolution images with more powerful GPUs will be investigated in future studies to see if increased image resolution in training helps to better focus the results from subsequently generated images. Our general observation is that, using the adopted resolution images, narrowing the ranges of the latent space does not necessarily produce better designs but can steer the designs toward the bounds of each category.

These various processes can also be used to target and enrich the data pool for a particular category as desired by the user. The increased quantity of data can then be used to narrow the width of that category, which leads to improved cGAN prediction and higher correlation to CFD results in any given category. In addition, these approaches can be combined with the shallow neural network methods detailed in [1] that directly link design parameters to pressure loss performance *without* the use of image data.

B. Accuracy of the Parameter Extraction Process

To understand the R^2 values obtained from the cGAN v's CFD runs, we also need to factor in the errors introduced by the process of extracting parameter values from the embedded histograms. Before we started the learning process, described above, we first converted

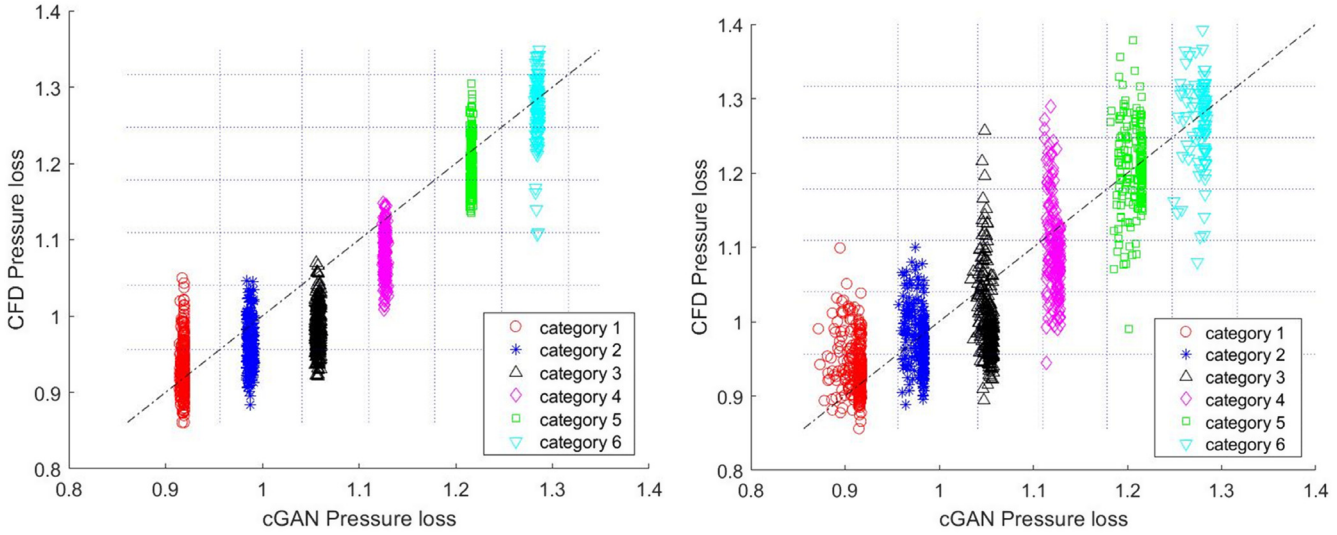

Fig. 11 Optimizing over the latent space. Left: Narrow range random uniform sampling [−0.03; 0.03]. Right: GA optimized with latent variables allowed to vary in the same narrow range; the dotted lines are the 45° lines of perfect correlation.

all Portable Network Graphic (PNG) images to include the histograms and the sector glyphs as in Fig. 3. During this process the number of pixels that represent each value was recorded for different image sizes and a pixel-to-value-to-pixel map created. To evaluate the accuracy of this process, a set of 1000 images were randomly selected. Each one of these has its own particular loss value. Using the process described above, the pixel counts for the bars were extracted, and a new design was generated and the corresponding CFD result recorded as the CFD pressure loss. The correlation between the two losses was established and R^2 calculated, as in Fig. 12.

It was observed that the $R^2 = 0.9237$ obtained during this process was very similar to the $R^2 = 0.9243$ values for the cGAN versus CFD predictions described earlier, which leads us to believe that if we could increase the accuracy of the data embedding process, we could possibly improve on the accuracy of the cGAN prediction; i.e., the cGAN accuracy is being limited by our ability to extract parameter values from the generated images, even using the embedded histogram technique. This is why we use a half-sector glyph to represent the crucial pressure loss property—by using such a glyph we can encode the loss more accurately with finite pixel resolution than is possible with simple histogram bars—it is area and angle based, and each value is thus represented by a higher number of pixels that change in a more graduated fashion than simple bars, hence improving its accuracy. We would ideally use such glyphs for all the quantities of interest, but this would require the room to place 16 such items in the already crowded image space.

C. Impact of Embedded Data on cGAN Training

We have also investigated whether embedding data into the image sets has any impact on the training process used to set up the cGAN. To do this we used a subset from our data set containing 50,000 images and trained cGANs both with and without the embedded information. The training histories are presented in Figs. 13 and 14, respectively. The training times are essentially the same, but it is clear from the discriminator score for the images without added data that training stalls—the score stabilizes at 0.5 after around 10,000 iterations (20 epochs), meaning that the discriminator cannot tell the difference between fake and real, and the relevant gradients in the training process have collapsed to zero. Moreover, close scrutiny of the resulting generated images reveals a much more grainy texture to the flowfields. It seems that if the images are too similar to each other this impedes training as might be expected—the presence of the histograms that show significant changes from image to image seems to help the process work. The cGAN generated without embedded data can, however, be used to seed the training of the enhanced data set—this speeds up the subsequent training process substantially. To test this we used the network trained without the histograms and glyphs to initiate training of the images with the added data.

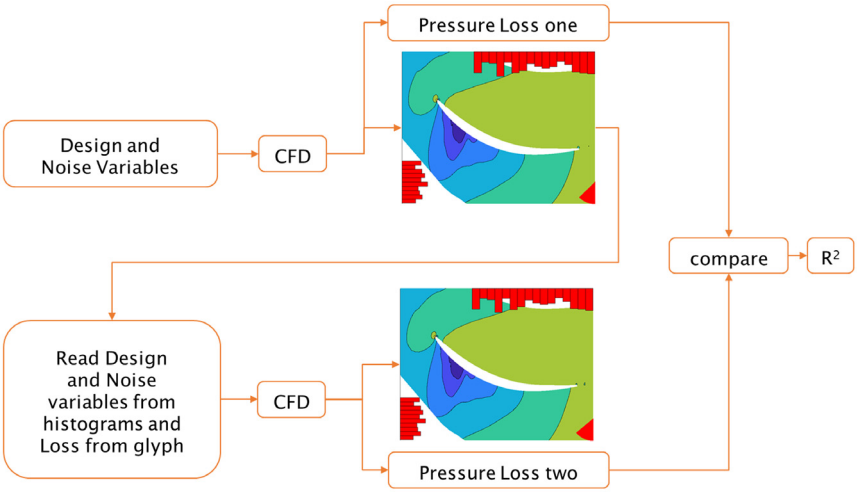

Fig. 12 Assessing the accuracy of the histograms and the glyph.

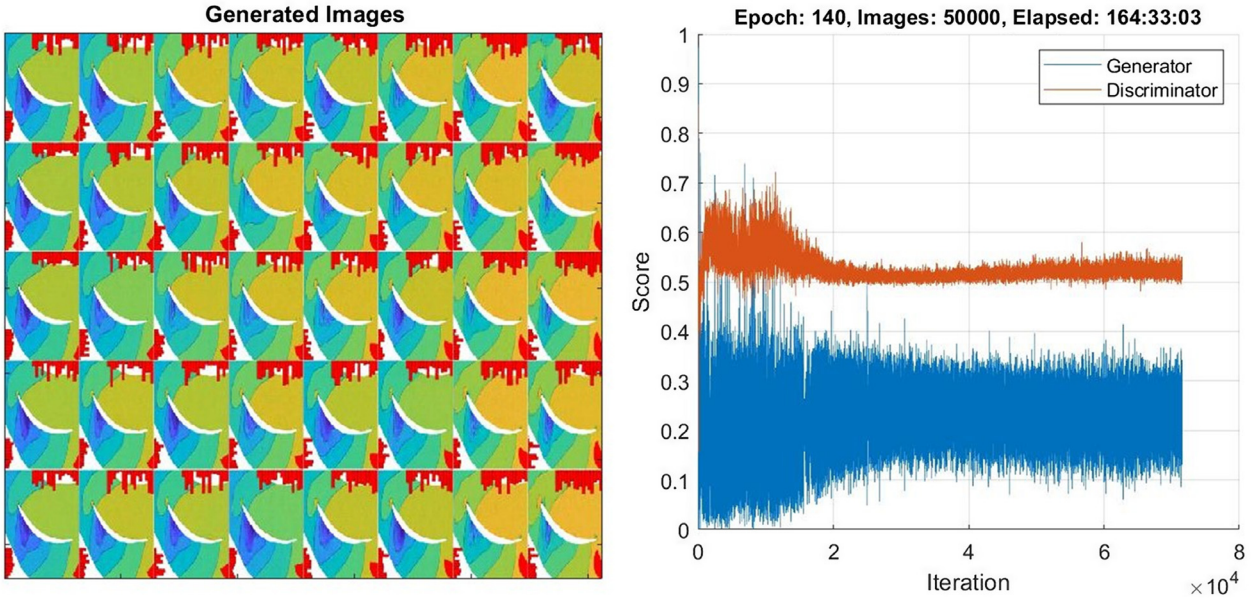

Fig. 13 Images and training history for 50,000 images with embedded data.

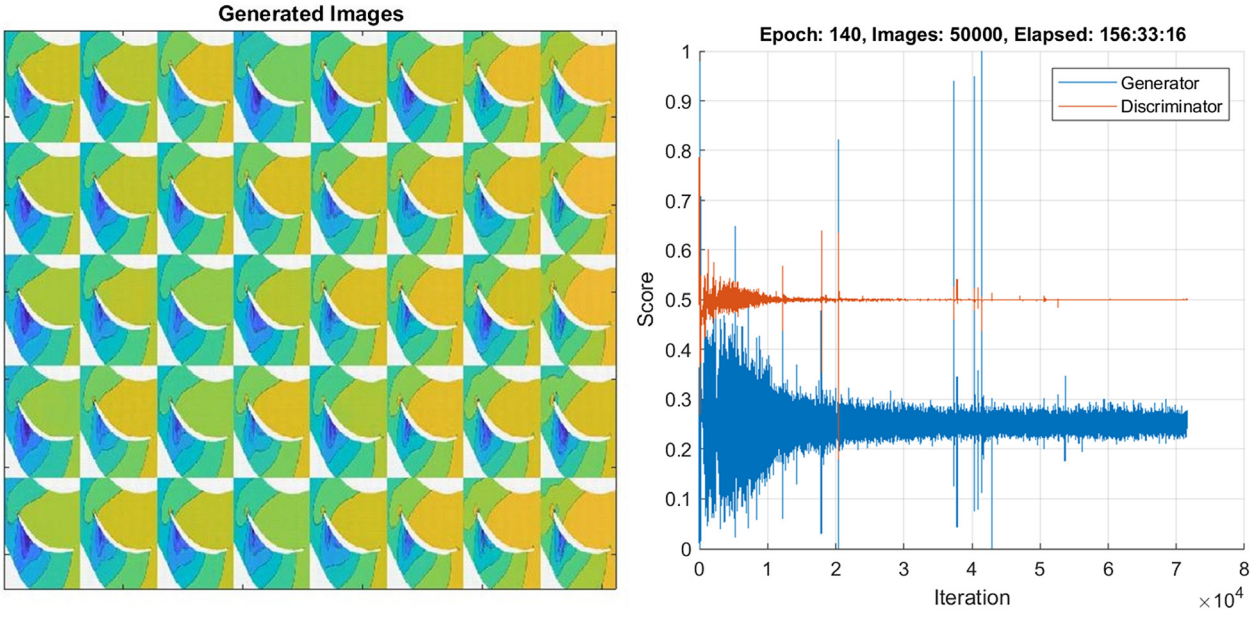

Fig. 14 Images and training history for 50,000 images without embedded data.

Compared to training from scratch, the training process adapted to the presence of this extra information within seven epochs, while a fresh network required 200 epochs for converged training.

V. Conclusions

Until recently, graphical information was mainly used for visualization purposes in engineering design workflows and was typically deemed insufficiently accurate to give a more detailed representation in quantitative terms. Image information is generally based on pixels as opposed to CAD representations that are mainly based on nonrational B-splines and numerical information stored in matrices and spreadsheets. With the advancement of cGANs, computer power, and specifically GPUs, it has become possible to utilize image information for machine learning purposes in an engineering sense. This paper has demonstrated the use of deep learning cGAN methods to generate airfoil shapes and pressure fields that have been validated against much more expensive CFD runs. We have shown a method using histograms and glyphs to encode numerical information inside the training image set that provides a bridge between a purely pixelated image and a parameterized CAD geometry. It can be used in situations where the variation of the geometry is too small to directly read with sufficient accuracy from an image. It also seems to help support the cGAN training process itself. These methods augment more traditional surrogate modeling process using Gaussian process regression or shallow neural networks.

Acknowledgments

The authors would like to thank Rolls-Royce plc for providing approval to publish this work. The research leading to these results has received funding from Innovate UK, under the Collaboration Across Business Boundaries (COLIBRI) project (Ref no. 113296).

References

[1] Keane, A. J., and Voutchkov, I. I., "Surrogate Approaches for Aerodynamic Section Performance Modeling," *AIAA Journal*, Vol. 58, No. 1, 2020, pp. 16–24.
https://doi.org/10.2514/1.J058687

[2] Guo, X., Li, W., and Iorio, F., "Convolutional Neural Networks for Steady Flow Approximation," *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Assoc. for Computing Machinery, New York, Aug. 2016, pp. 481–490.
https://doi.org/10.1145/2939672.2939738

[3] Yilmaz, E., and German, B., "A Convolutional Neural Network Approach to Training Predictors for Airfoil Performance," *Emerging Methods, Algorithms and Software Development II*, AIAA Paper 2017-3660, June 2017.
https://doi.org/10.2514/6.2017-3660

[4] Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., and Kaushik, S., "Prediction of Aerodynamic Flow Fields Using Convolutional Neural Networks," *Computational Mechanics*, Vol. 64, 2019, pp. 525–545.
https://doi.org/10.1007/s00466-019-01740-0

[5] Chen, W., Chiu, K., and Fuge, M., "Aerodynamic Design Optimization and Shape Exploration Using Generative Adversarial Networks," *AI and HMI in Engineering Design*, AIAA Paper 2019-2351, Jan 2019.
https://doi.org/10.2514/6.2019-2351

[6] Achour, G., Sung, W. J., Pinon-Fischer, O. J., and Mavris, D. N., "Development of a Conditional Generative Adversarial Network for Airfoil Shape Optimization," *AIAA Scitech 2020 Forum*, AIAA Paper 2020-2261, Jan. 2020.
https://doi.org/10.2514/6.2020-2261

[7] 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*, AIAA Paper 2020-2128, 2020.
https://doi.org/10.2514/6.2020-2128

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

[9] Hicks, R., and Henne, P., "Wing Design by Numerical Optimization," *Journal of Aircraft*, Vol. 15, July 1978, pp. 407–412.
https://doi.org/10.2514/3.58379

[10] Shahpar, S., and Lapworth, L., "Parametric Design and Rapid Meshing Systems for Turbomachinery Optimization," *ASME Turbo Expo 2003*, ASME Paper GT2003-38698, 2003, pp. 579–590.

[11] Moinier, P., and Giles, M. B., "Preconditioned Euler and Navier-Stokes Calculations on Unstructured Grids," *Proceedings 6th ICFD Conference on Numerical Methods for Fluid Dynamics*, Univ. of Oxford, Oxford, U.K., 1999, pp. 87–95.

[12] Cumpsty, N. A., *Compressor Aerodynamics*, Longman Scientific & Technical, Harlow, Essex, U.K., 1989, Chap. 4.

[13] Prasad, A., "Calculation of the Mixed-Out State in Turbomachine Flows," *ASME Journal of Turbomachinery*, Vol. 127, No. 3, July 2005, pp. 564–572.
https://doi.org/10.1115/1.1928289

[14] Keane, A. J., "Comparison of Several Optimisation Strategies for Robust Turbine Blade Design," *Journal of Propulsion and Power*, Vol. 25, No. 5, 2009, pp. 1092–1099.
https://doi.org/10.2514/1.38673

[15] Keane, A. J., "Use of Cokriging for Robust Design Optimization," *AIAA Journal*, Vol. 50, No. 11, 2012, pp. 2351–2364.
https://doi.org/10.2514/1.J051391

[16] Kumar, A., Nair, P. B., Keane, A. J., and Shahpar, S., "Robust Design Using Bayesian Monte Carlo," *International Journal for Numerical Methods in Engineering*, Vol. 73, 2008, pp. 1497–1517.
https://doi.org/10.1002/nme.2126

[17] Kumar, A., "Robust Design Methodologies: Application to Compressor Blades," Ph.D. Thesis, Univ. of Southampton, Southampton, England, U.K., 2006.

[18] MATLAB User Manual on cGAN, Ver. 2021, https://uk.mathworks.com/help/deeplearning/ug/train-conditional-generative-adversarial-network.html.

[19] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. "Going Deeper with Convolutions," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Soc., Washington, D.C., June 2015, pp. 1–9.

[20] MATLAB Online Help for Training cGANs, https://uk.mathworks.com/help/deeplearning/ug/monitor-gan-training-progress-and-identify-common-failure-modes.html [retrieved 27 July 2020].

P. G. Tucker
Associate Editor