01 Jan 2021

# A Convolutional Neural Network Model based on Multiscale Structural Similarity for the Prediction of Flow Fields

Yifu An

Xiaosong Du
*Missouri University of Science and Technology*, xdnwp@mst.edu

Joaquim R.R.A. Martins



## Recommended Citation

Y. An et al., "A Convolutional Neural Network Model based on Multiscale Structural Similarity for the Prediction of Flow Fields," *AIAA Aviation and Aeronautics Forum and Exposition, AIAA AVIATION Forum 2021*, article no. AIAA 2021-3061, American Institute of Aeronautics and Astronautics, Inc., AIAA, Jan 2021.
The definitive version is available at https://doi.org/10.2514/6.2021-3061

AIAA AVIATION Forum
August 2-6, 2021, VIRTUAL EVENT
AIAA AVIATION 2021 FORUM

10.2514/6.2021-3061

# A Convolutional Neural Network Model Based on Multiscale Structural Similarity for the Prediction of Flow Fields

Yifu An*, Xiaosong Du†, and Joaquim R. R. A. Martins‡
*University of Michigan, Ann Arbor, MI, 48109, USA*

**We have seen the emerging applications of deep neural networks for flow field predictions in the past few years. Most of the efforts rely on the increased complexity of the model itself or take advantage of novel network architectures, such as convolutional neural networks (CNN). However, reaching low prediction error cannot guarantee the quality of the predicted flow fields in terms of the perceived visual quality. This work introduces the multi-scale structural similarity (MS-SSIM) index method for flow field prediction. First, we train CNN models using the commonly used root mean squared error (RMSE) loss function as the reference. Then we introduce the SSIM loss function to capture the high-level features. Furthermore, we investigate the effects of the MS-SSIM weights on the predictive performance. Our results show that while the pixel-wise prediction error of RMSE-based models is as low as $1.3141 \times 10^{-2}$, the perceived visual quality of the predicted flow fields, such as contour-line smoothness, is poorly represented. In contrast, the MS-SSIM models significantly improve the perceived visual quality with an SSIM loss value as low as $7.370 \times 10^{-3}$, although having a slightly higher prediction error of $1.3912 \times 10^{-2}$. These values are $41.7\%$ lower in the SSIM loss and $5.9\%$ higher in the RMSE than the best RMSE model. In particular, we report that a weight combination of $0.3$ and $0.7$ for the MS-SSIM loss function provides the best predictive performance in our case. Our study has pointed out a possible future endeavor to invent a quality metric based on structural similarity, which should excel in flow-field-related approximations.**



## I. Introduction

Computational fluid dynamics (CFD) simulations play a crucial role in engineering industry. State-of-the-art CFD solvers such as ADflow [1] and OpenFOAM [2–4] facilitate the use of high-fidelity flow field solutions in risk analyses and design optimizations. Still, such a process normally requires costly computational time on high-performance computing resources. Among the branches of research that address this problem is surrogate modeling. Surrogates require potentially costly training, but once trained, they are fast and computationally efficient, and repeatedly running them for predictions will pay off the training expense [5].

Traditional surrogate modeling methods include multi-fidelity models [6], where a combination of high- and low-fidelity models balances the computational cost and accuracy; hierarchical kriging [7], where the kriging model of a low-fidelity function improves the accuracy of the kriging model of the corresponding high-fidelity function; mixture of experts [8] fuses multiple surrogates which model different parts of the input space independently.

As the application of artificial neural networks (ANN) attracts interests in other research areas [9, 10], works using ANN models as surrogates also turn out to be an interesting topic. Papila *et al.* combined the radial basis neural network (RBNN) with polynomial models in the optimization of a supersonic turbine [11]. They trained RBNN models on a small set of CFD solutions and then generated a larger set using the RBNN to augment the polynomial model. Zhang *et.al* used a convolutional neural network (CNN) to predict the lift coefficient of an airfoil with respect to the freestream conditions and airfoil geometry [12]. Sekar *et al.* successfully used deep multilayer perceptron (MLP) models to predict the flow field over an airfoil [13]. The MLP models had 8-12 hidden layers that consisted of 800-1200 neurons each. These models predicted flow field information as a function of the $x$, $y$ coordinates, the angle of attack, the Reynolds number, and 16 parameters for the airfoil geometry. The models also used the mean $L_2$-norm of the point-wise errors as the loss function. Bouhlel *et al.* used modified Sobolev training for artificial neural networks (mSANN) to predict aerodynamic force coefficients for subsonic and transonic regimes [14]. To improve the model's performance, they gradually introduced gradient information during the training. The model had six hidden layers and involved four

*Graduate Student, Department of Aerospace Engineering.
†Post-Doctoral Fellow, Department of Aerospace Engineering, AIAA Member
‡Professor, Department of Aerospace Engineering, AIAA Associate Fellow

different activation functions. The training process incorporated gradient information by using a weighted sum of the $L_2$-norm of the output error and that of the partial derivatives' error as the loss function.

While many of the recent works used ANN to predict scalar outputs (*e.g.* lift coefficient, velocity at a point), some also studied the prediction of 2-D (or higher-dimensional) flow fields. These works utilized CNN-based models because CNN reduces the total number of trainable weights and respects the topology of graphical data [15]. Guo *et al.* used an autoencoder architecture to learn the CFD solutions of flow fields over various objects at low Reynolds numbers [16]. The architecture consisted of an encoder and a decoder: the encoder extracted high-level geometry features, and the decoder predicted the surrogate solution. Bhatnagar *et al.* extended Guo *et al.*'s work to the prediction of flow field solutions over airfoils [17]. They improved the autoencoder model by sharing the decoding layers among output variables. The architecture they proposed had 6 convolutional layers in total, each having 300 feature maps. They trained and validated the CNN on a dataset of 252 RANS simulations: the input spaced was sampled uniformly for 4 Reynolds numbers, 21 angles of attack, and 3 different airfoils. They made few attempts to generalize the model to arbitrary airfoil shapes, as this was not the focus of the work.

Most of the previous works used the mean squared error (MSE) [16, 18] or root mean squared error (RMSE) (including the $L_2$-norm, which is proportional to the RMSE) [14, 19] as the loss functions. However, predictions given by models using such loss functions suffer from non-smooth contour lines and spurious bumps in regions that should be flat and smooth. Previous works [17, 20] addressed this problem by adding a term for gradient difference loss [21] to the MSE/RMSE loss functions. This term measures the error in the difference between adjacent pixels. Hence, the modified loss function is dependent on a neighborhood of a radius of one pixel. Meanwhile, Zhao *et al.*'s study in image restoration inspired a new direction: metrics for image quality, the structural similarity (SSIM) index, and the multi-scale structural similarity (MS-SSIM) index [22] outperformed the $L_2$ error as the loss function for various image-processing CNNs [23].

Our work utilizes the framework laid by Bhatnagar *et al.* but also experiments with loss functions that are novel in flow field prediction, *i.e.* the SSIM and MS-SSIM. The primary contribution is to show the promise of improving the quality of flow field predictions by using structural-similarity-based loss functions. "Quality" here refers to how good the predictions appear to a human observer, or *perceived visual quality* [24]. In our case, a predicted flow field of "high quality" would refer to one with smooth contours that highly resemble the simulation results. High-quality predictions are favorable, especially for this work's possible applications, such as real-time flow field feedback in web-based interactive airfoil optimization. Secondly, we incorporate the state-of-the-art generative adversarial network (GAN) architecture [25] into the model. We realize this by replacing the decoder in previous works [16, 17] with pre-defined BSplineGAN airfoil parameterization [26]. This reduces the number of parameters of the architecture by eliminating the need for decoding layers. Finally, we train the model on a dataset of 44,000 RANS simulations.

The remainder of this paper is organized as follows. In Section II, we introduce our overall methodology and the general workflow. Then we present and interpret the results of the study, including the average error and graphical output of the models in Section III. Finally, we conclude this paper and discuss possible extensions in Section IV.

## II. Methodology

This section describes the general process to obtain structural-similarity-based CNN models for flow field predictions. We detail here the key components, *i.e.*, RANS simulations, image postprocessing, model training and verification, and model comparison.

### A. General Workflow

Figure 1 shows the workflow of the proposed approach using the Extended Design Structure Matrix (XDSM) [27], which consists of the following five components:

1) After running ADflow simulations on the random input parameters ($x_{rnd}$), *i.e.*,B-spline generative adversarial networks (BSplineGAN) [26] parameters ($x_{bsgan}$), angle of attack ($\alpha$), Reynolds number ($Re$), and the Mach number ($\mathcal{M}$), we have over 44,000 instances of flow fields.
2) We then process the contour plots of the raw flow field data (denoted by $IMG$), which have a resolution of 450×450. We lower the resolution of the images down to 150×150 with the guidance of a parametric study provided in the following subsection.
3) We then train various CNN models with different loss functions and numbers of feature maps per layer. The models take the random input parameters, $x_{rnd}$, as the input, and return contour plots of the Mach number, $IMG_{150\times150}$, where the subscript means the images are 150 pixels in width and height respectively. This step

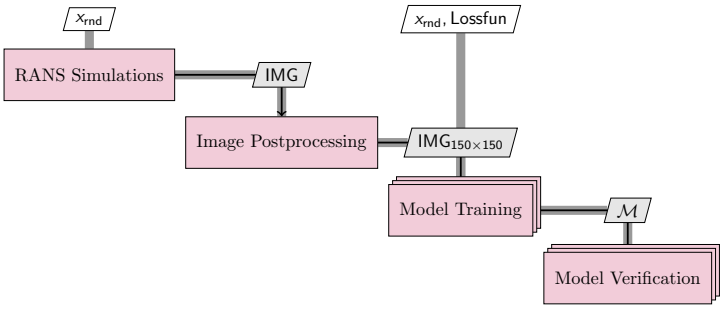



**Fig. 1 The XDSM of the workflow. The gray line represents passing of variables, and the black lines with arrow indicate the direction of the work flow.**



generates a series of models, $\mathcal{M}$.

4) We verify the model by computing its RMSE and SSIM errors on a testing data set. The former is the typical metric for pixel-wise prediction accuracy; the latter indicates how well a human observer can perceive the image [24].

The same workflow applies to all the tested network architectures with varied neuron setups and loss functions.

### B. RANS Simulations

In this work, we use pyHyp* as the mesh generator and ADflow† as the flow solver for the Reynolds-averaged Navier–Stokes equations. The pyHyp toolbox implements hyperbolic volume mesh marching schemes to extrude structured surface meshes into volume meshes [28], achieving mesh orthogonality and cell volume specification. In addition, pyHyp adds spatially-variable advanced techniques, *i.e.*, spatially-variable smoothing coefficient, metric correction procedures, local treatment of severe convex corners, extrapolation treatment of floating and axis boundaries, to improve the quality of the hyperbolic mesh [28]. ADflow is a finite-volume structured multiblock and overset mesh CFD solver distributed under an open-source license [1]. The key advanced methodology implemented in ADflow includes a discrete adjoint solver for efficient gradient calculation [29, 30] and a Python application programming interface for the convenience of case setup. ADflow integrates three numerical schemes to discretize the inviscid fluxes, the scalar Jameson–Schmidt–Turkel artificial dissipation scheme, a matrix dissipation scheme and a monotone upstream-centered scheme. ADlow computes the viscous flux gradients using a Green–Gauss approach and simultaneously solves the mean equations defined by the continuity of mass, momentum, and energy. The turbulence models, including the one-equation Spalart-Allmaras model and the two-equation shear stress transport model can be solved separately or with the mean flow equations. ADflow implements a few algorithms for the residual equation convergence. In this work, we use the fast and robust approximate Newton-Krylov algorithm as a globalization scheme for the full Newton–Krylov solver, which converges rapidly within the Newton basin of attraction.

We provide the pyHyp and ADflow with airfoil shapes generated by BSplineGAN model [26] and flight conditions sampled within broad ranges. BSplineGAN, a type of generative adversarial network (GAN) [25], intelligently generates practical airfoils by matching the data pattern and characteristics in an existing airfoil data set [31]. BSplineGAN, inspired by the Bezier-based GAN model [32, 33], adds a b-spline layer as the last layer of the generator networks. We direct the readers who have interests to the paper by Du *et al.* [26] for more details. The considered ranges of flight conditions include $\alpha$ within [0 deg, 3 deg], $\mathcal{M}$ within [0.3, 0.6] and $\log_{10}Re$ within [4, 10], where $M$ and $Re$ are sampled by Gaussian copula dependence sampling [26].

### C. Image Post-processing

We represent the solutions of the RANS simulations with the Mach number contours in a rectangular region around the airfoil, $(x, y) \in [-1, 2] \times [-1.5, 1.5]$, where the most important physics information exists. The Mach numbers in the physical space are normalized into $[0, 1]$ using minmax scaler such that the neuron networks consider them as gray-scale images. The original images have a resolution of $450 \times 450$. We notice the increase in predictive accuracy is

*https://github.com/mdolab/pyhyp

†https://github.com/mdolab/adflow

diminishing with the increasing number of pixels, while the size of the data grows quadratically. Therefore, we conduct a convergence study of image resolution to reduce the size of the data effectively. Results (Fig. 2) show lowering the resolution by a factor of 3(×3), *i.e.* using images with a resolution of $150 \times 150$, balances the computational cost and loss in accuracy. The success of previous work [17] supports this choice.

In this case, the average root mean square error due to postprocessing is 0.035. Note that, since the pixel-wise grayscale values are normalized into the range of [0, 1], this error corresponds to a 3.5% relative error due to down-sampling. The postprocessing yields a training validation set of 44,000 instances of the flow field and a testing set of 120 instances.

### D. Model Training

We train the neural network models on the training-validation data set. The training and validation sets follow a 90%–10% split of the training-validation set. We implement and train the models following different neuron setups, and loss functions within `TensorFlow` [34]. We train the neural network models for the Mach number fields on one "Skylake" computing node with 48 cores [35].

#### *1. Neuron Setups*

We test three types of neuron setups, namely, "Simple", "Plain", and "Complex". These aliases are a straightforward description of the complexity of the model: the simple model has less than 20,000 trainable parameters, the plain one has around 70,000, and the complex one has more than 272,000. Table 1 shows the architecture of the models we test. All three types of models have a similar structure: a dense layer is fully connected to the input layer, which is then reshaped into the form of feature maps. Three convolutional layers follow, with increasing numbers of feature maps. One final convolutional layer then returns a 150×150 array as the prediction corresponding to the input. The convolutional layers work similarly as an autoencoder network model. In fact, we replace the encoder with 26 parameters from BSplineGAN parameterization, concatenated with three parameters for the freestream condition. This is unlike commonly used autoencoder networks, where the model first encodes the geometry of an airfoil in the form of a signed distance function [16, 17]. Such an improvement reduces the number of parameters of the model by eliminating the need for encoding layers (1/2 of the original architecture).

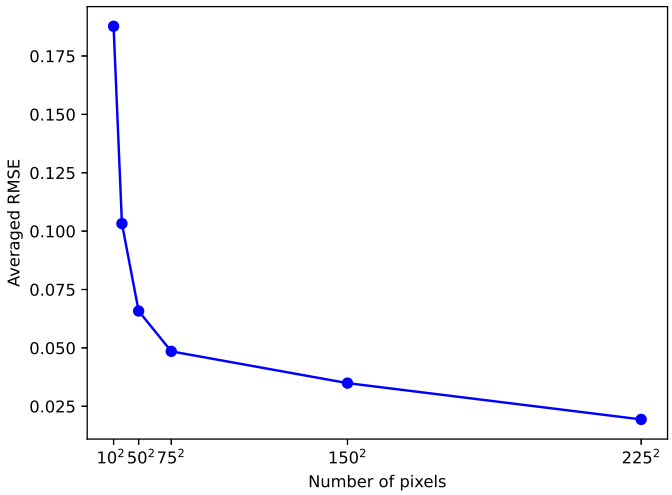



**Fig. 2 Convergence study of resolution reduction for the Mach number. The $y$ axis is the training set average RMSE value of the reduced image with respect to the original $450 \times 450$ image. The $x$ axis is the number of pixels.**

**Table 1 The architecture of models we test. For the "input", "dense" layer, and "reshape" operation, we list the shape of the intermediate tensor and the activation function where applicable; for the convolutional operations, we list the number of feature maps, the size of upsampling, and the activation function. For example, for the plain network, the input layer has the shape $(29,)$. It is fully connected to a dense layer of shape $(64,)$ and reshaped into a tensor of shape $(2, 2, 16)$. The three convolutional layers have $16, 32, 64$ feature maps, respectively, and the upsampling sizes ensure that these layers will end up with a tensor with the expected shape (resolution). The last convolutional layer is the output layer, so upsampling is not used here. We use $relu$ as the activation function for the dense layer and $sigmoid$ for the convolutional layers.**

| Operation/layer | Model Alias and Architecture | | |
|---|---|---|---|
| | "Simple" | "Plain" | "Complex" |
| **Input** | $(29,)$ | $(29,)$ | $(29,)$ |
| **Dense** | $(32,)$<br>relu | $(64,)$<br>relu | $(128,)$<br>relu |
| **Reshape** | $(2, 2, 8)$ | $(2, 2, 16)$ | $(2, 2, 32)$ |
| **Conv2D, 1** | 8<br>$3 \times 3$<br>sigmoid | 16<br>$3 \times 3$<br>sigmoid | 32<br>$3 \times 3$<br>sigmoid |
| **Conv2D, 2** | 16<br>$5 \times 5$<br>sigmoid | 32<br>$5 \times 5$<br>sigmoid | 64<br>$5 \times 5$<br>sigmoid |
| **Conv2D, 3** | 32<br>$5 \times 5$<br>sigmoid | 64<br>$5 \times 5$<br>sigmoid | 128<br>$5 \times 5$<br>sigmoid |
| **Conv2D, Output** | 1<br>/<br>sigmoid | 1<br>/<br>sigmoid | 1<br>/<br>sigmoid |

*2. Loss Functions*

We train models with different loss functions: the RMSE, the SSIM, and the MS-SSIM based on 2 scales. The RMSE loss function is given as

$$\mathcal{L}_{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_{\text{true},i} - y_{\text{pred},i}\right)^2}, \tag{1}$$

where $y_{\text{true},i}$, $y_{\text{pred},i}$ are the true and predicted values at the $i$-th pixel, and $N$ is the total number of pixels. The SSIM loss function is given as [22]

$$\begin{aligned}\mathcal{L}_{\text{SSIM}} &= \frac{1}{N}\sum_{i=1}^{N} 1 - \text{SSIM}\,(i) \\ &= \frac{1}{N}\sum_{i=1}^{N} 1 - \frac{2\mu_{\text{true}}\mu_{\text{pred}} + C_1}{\mu_{\text{true}}^2 + \mu_{\text{pred}}^2 + C_1} \cdot \frac{2\sigma_{\text{true,pred}} + C_2}{\sigma_{\text{true}}^2 + \sigma_{\text{pred}}^2 + C_2} \\ &= \frac{1}{N}\sum_{i=1}^{N} 1 - l\,(i) \cdot cs\,(i)\,,\end{aligned} \tag{2}$$

where $l(i)$ is called the luminance comparison, and $cs(i)$ the product of the contrast comparison and structure comparison at pixel $i$. The local statistics, $\mu$ and $\sigma$, are computed using a smooth windowing approach [24]. We use the same values of the constants $C_1 = 1^{-4}, C_2 = 9^{-4}$ as those in the original paper. We highlight a property of the SSIM loss function

here, that $\mathcal{L}_{\text{SSIM}} = 0$ if and only if pred = true. The MS-SSIM loss function is [22]

$$\begin{aligned}\mathcal{L}_{\text{MS-SSIM}} &= \frac{1}{N}\sum_{i=1}^{N} 1 - \text{MSSSIM}\,(i) \\ &= \frac{1}{N}\sum_{p=i}^{N} 1 - l_M\,(i)\cdot\prod_{j=1}^{M} cs_j^{w_j}\,(i)\,,\end{aligned} \tag{3}$$

where we compute the product of the contrast and structure comparison on $M$ scales and $w_j$'s adjust the relative weight of the scales. The image at one of the intermediate scales is an image downsampled by a factor of 2 from the previous scale. In contrast, the first scale ($M = 1$) is the original resolution. In our 2-scale case, $(w_1, w_2) \in \{(1,0),(0.9,0.1),(0.8,0.2),\ldots,(0,1)\}$. We keep the sum of the weights to one to study the relative importance of the first and second scales while searching through the space discretely.

*3. Hyperparameter Setups*

We train all models with the Adam optimizer [36] with a learning rate of $10^{-3}$, and parameters $(\beta_1, \beta_2) = (0.9, 0.999)$. We train them using a batch size of 16. We train simple models for 140 epochs, plain ones for 160 epochs, and complex ones for 80 epochs. After these numbers of epochs, the corresponding models converge in terms of the training and validation errors, without overfitting, under such settings.

### E. Model Verification

We verify the models on the testing set of 120 flow field instances. We compute the RMSE and SSIM values on the whole set to measure the quality of the model. For the individual cases to be shown in the next section, we will also compute its RMSE and SSIM values for richer predictive insights. When comparing models trained using different loss functions, one has to refer to the two metrics simultaneously. This is because models naturally perform better when evaluated with the loss function it is trained with.

## III. Results and Discussion

This section compares the models trained using the SSIM and MS-SSIM as the loss function with the baseline models trained via RMSE loss function. We complete studies on the predictive performance with respect to increasing numbers of feature maps. We also investigate the effects of the MS-SSIM weights on 2 scales. At the end of this section, we compare the trained models on a series of example cases. All neural networks models predict the Mach fields.

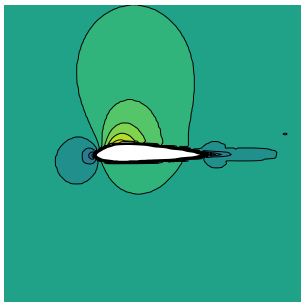
### A. Loss Function Comparison

We train a plain model and a complex one using the RMSE loss function and similarly using the SSIM loss function (refer to Section II.D.3 for the other hyperparameters; this statement will be omitted from now on). Table 2 summarizes the performance of the four models in terms of their RMSE and SSIM errors on the testing set. All the models have an RMSE value around 0.015, which translates to about 5% relative error if we normalize with the minimum freestream Mach number 0.3, The best model comes from the complex architecture: the complex RMSE model has a 4.4% relative error, whereas the complex SSIM model has a 5.1% relative error. The models trained with the RMSE loss function have a lower average RMSE on the testing set, while those trained with the MS-SSIM loss function with a lower SSIM. However, switching to the SSIM loss function on a plain model reduces the SSIM loss by 41.2%, at the cost of increasing the RMSE by a mere 16.3%; the numbers are 40.1% and 16.9% for the complex architecture. Figure 3 shows the results of an arbitrary case. The results explain our motivation to find new loss functions for the CNN: the RMSE-based models, while more accurate pixel-wise, systematically fail to generate smooth contours and produce "splotchy artifacts" in "flat" regions between adjacent contour levels [23]. The SSIM loss function remedies this by using statistics around a pixel (Eqn. 2), rather than only the pixel itself. Due to the smoothened contours, the predictions appear better to human viewers even with a slight decrease in pixel-wise accuracy. This is also an illustration of the aforementioned report on changes in model accuracy: the benefits of the SSIM loss over the RMSE loss are substantial, but the penalty is minor.

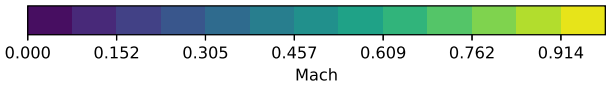
**Table 2 Prediction errors of various models on the testing set.**

| Model | $RMSE$ $(10^{-2})$ | $1 - SSIM$ $(10^{-2})$ |
|---|---|---|
| Plain RMSE | 1.4755 | 1.3301 |
| Plain SSIM | 1.7153 | 0.7818 |
| Complex RMSE | 1.3141 | 1.2633 |
| Complex SSIM | 1.5367 | 0.7570 |

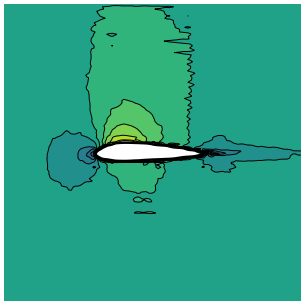




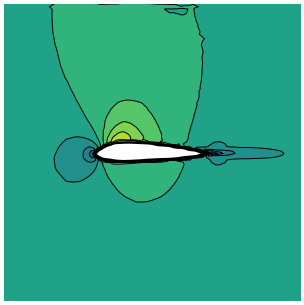
**(a) Truth (from ADflow).**



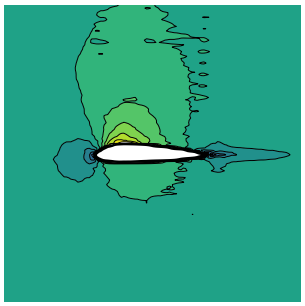

**(b) The color bar.**



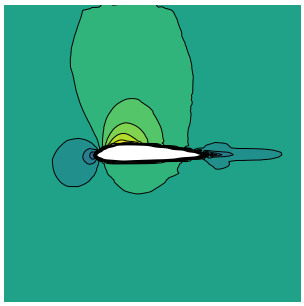
**(c)** $RMSE = 1.0843 \cdot 10^{-2}$, $SSIM = 1.5540 \cdot 10^{-2}$.



**(d)** $RMSE = 1.4452 \cdot 10^{-2}$, $SSIM = 0.9931 \cdot 10^{-2}$.



**(e)** $RMSE = 0.9795 \cdot 10^{-2}$, $SSIM = 1.4097 \cdot 10^{-2}$.



**(f)** $RMSE = 1.1714 \cdot 10^{-2}$, $SSIM = 0.9424 \cdot 10^{-2}$.

**Fig. 3 Truth and model prediction for the Mach numbers over a test case. The freestream conditions are** $AoA = 2.1142°$, $Re = 9.0791 \cdot 10^4$, $\mathcal{M} = 0.50511$**. (c) is the plain RMSE model; (d) the plain SSIM model; (e) the complex RMSE model; (f) the complex SSIM model.**

### B. Model Complexity Study

In our attempts to improve the model accuracy, the plain models do not capture the complex physics. Figure 4 reveals that the model accuracy achieved through the SSIM loss function increases with the increasing numbers of feature maps. Meanwhile, the number of parameters also increases super-linearly, increasing the computational cost. Figure 4 also suggests that the benefits we get by increasing the numbers of feature maps diminish. Therefore, we will use the complex model as our most accurate architecture. However, we may further increase the number of feature maps in future work.

Figure 5 shows the performance of the simple, plain, and complex models in the same case. We observe that increasing the model complexity decreases both the SSIM error and the RMSE simultaneously. The shapes of the contours also appear to be more and more visually realistic.

### C. Multi-Scale Structural Similarity

We study the effects that MS-SSIM weights have on the flow field prediction. The weighting factor of the first scale ($w_1$) sweeps uniformly in $[0, 1]$, while $w_2$ equals $1 - w_1$ to maintain unity. Note that when $w_1 = 1.0$, we obtain the original SSIM loss function. Table 3 lists the accuracy of the 2-scale, complex models on the testing set, as well as the result of using the weighting factors proposed by Wang et al. [22]. The latter corresponds to a 5-scale loss function, whose weighting factors are calibrated using a cross-scale calibration that subjectively depends on human interviewees. Table 3 shows that some pair of weight factors do not improve the accuracy at all, while others improve one of RMSE and SSIM, if not both. We show the extreme case of $(0, 1.0)$, which is the SSIM on a low-resolution (25%) version of the original image, as well as the original MS-SSIM proposed by Wang et al. [22], which leads to a 36% higher RMSE compared with the single scale version. The original weighting factors do not work well in our case because the calibration in the original paper does not involve images of abstract objects, such as a flow field solution. On the other hand, weight factors $(0.3, 0.7)$ reduces the RMSE the most, by 9.5%. Figure 6 shows the predictions of these models on an arbitrary test case. We exclude three models with the highest RMSE to accommodate the rest.

We observe that not only does the weights $(0.3, 0.7)$ perform the best on the testing set in terms of the RMSE, it also produces the most visually pleasing result on such an individual case (Fig. 6i). Therefore, we will use $(w_1, w_2) = (0.3, 0.7)$ to represent 2-scale SSIM loss function family in this work. Note that we do not refer to the SSIM loss value for model quality because it is biased for being in the same family as the MS-SSIM.

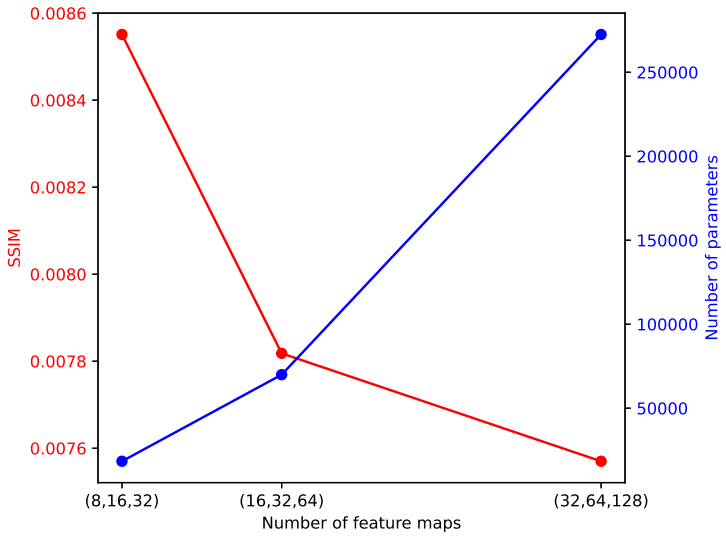
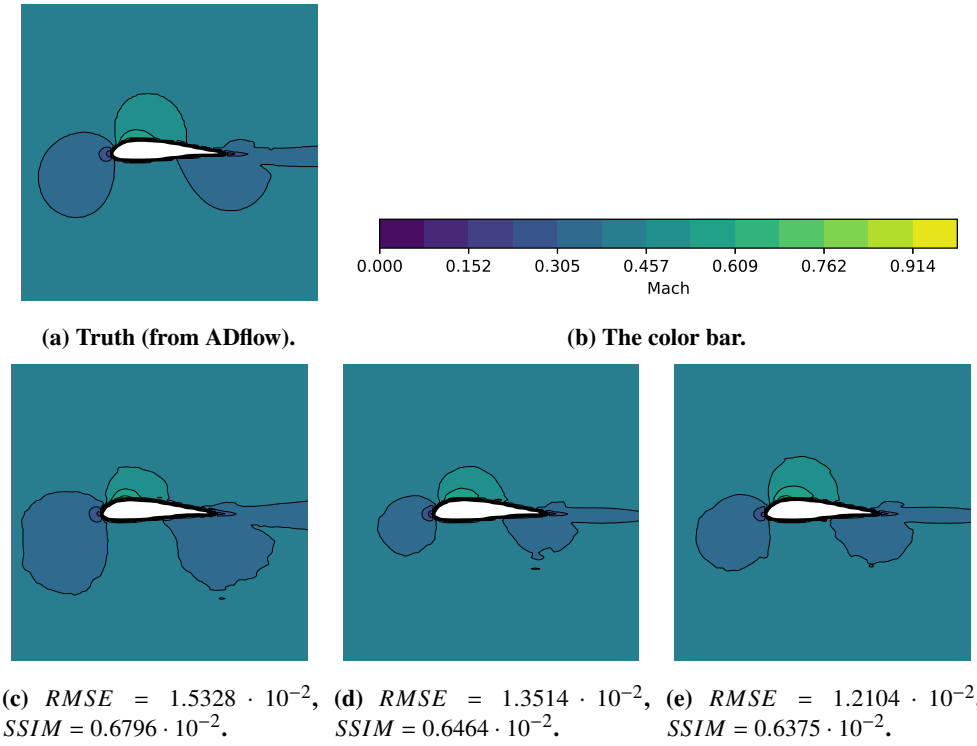



**Fig. 4 The SSIM on the tesing set (left) and the number of parameters (right) as a function of the number of feature maps.**



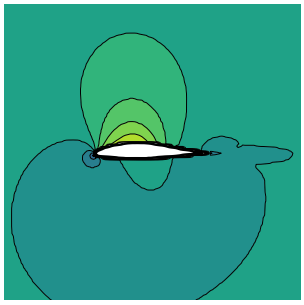

**(a) Truth (from ADflow).** **(b) The color bar.**

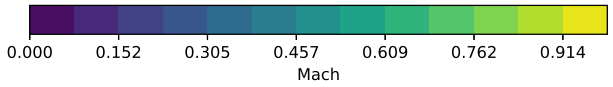
**(c)** $RMSE = 1.5328 \cdot 10^{-2}$, $SSIM = 0.6796 \cdot 10^{-2}$. **(d)** $RMSE = 1.3514 \cdot 10^{-2}$, $SSIM = 0.6464 \cdot 10^{-2}$. **(e)** $RMSE = 1.2104 \cdot 10^{-2}$, $SSIM = 0.6375 \cdot 10^{-2}$.

**Fig. 5 Truth and model prediction for the Mach number over an airfoil for the freestream conditions** $AoA = 1.3427°$, $Re = 9.2386 \cdot 10^6$, $\mathcal{M} = 0.32178$. **(c) the simple SSIM model; (d) the plain SSIM model; (e) the complex SSIM model.**

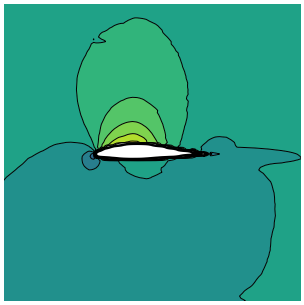
**Table 3 Prediction errors of complex models with MS-SSIM as the loss function on the testing set.**

| Loss function (Weights $(w_1, w_2)$) | $RMSE$ $(10^{-2})$ | $1 - SSIM$ $(10^{-2})$ |
|---|---|---|
| Wang et al. [22] $(0.0448, 0.2856, 0.3001, 0.2363, 0.1333)$ | 2.0863 | 0.8880 |
| SSIM $(1.0, 0)$ | 1.5367 | 0.7570 |
| $(0.9, 0.1)$ | 1.7424 | 0.7732 |
| $(0.8, 0.2)$ | 1.5977 | 0.7473 |
| $(0.7, 0.3)$ | 1.4951 | 0.7447 |
| $(0.6, 0.4)$ | 1.4717 | 0.6972 |
| $(0.5, 0.5)$ | 1.3926 | 0.7364 |
| $(0.4, 0.6)$ | 1.4448 | 0.7528 |
| $(0.3, 0.7)$ | 1.3912 | 0.7370 |
| $(0.2, 0.8)$ | 1.5257 | 0.7368 |
| $(0.1, 0.9)$ | 1.5391 | 0.7673 |
| $(0, 1.0)$ | 6.9201 | 7.7780 |

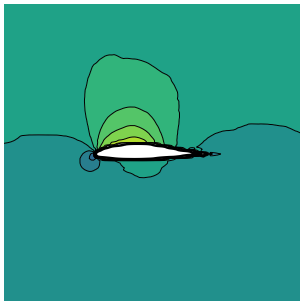




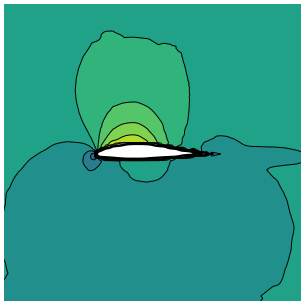

**(a) Truth (from ADflow).** **(b) The color bar.**



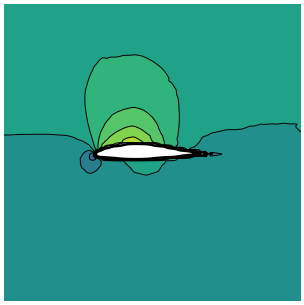
**(c) SSIM.** $RMSE = 1.2110 \cdot 10^{-2}$**,** $SSIM = 1.1168 \cdot 10^{-2}$**.**



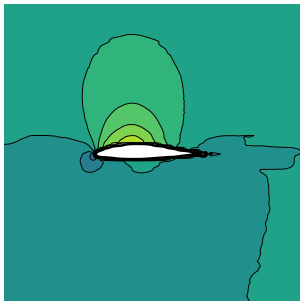
**(d)** $(0.8, 0.2)$**.** $RMSE = 1.2278 \cdot 10^{-2}$**,** $SSIM = 1.0196 \cdot 10^{-2}$**.**



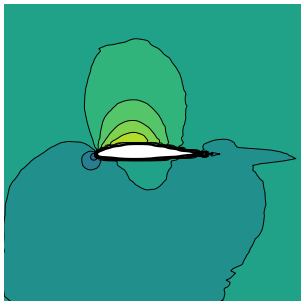
**(e)** $(0.7, 0.3)$**.** $RMSE = 1.1722 \cdot 10^{-2}$**,** $SSIM = 1.1341 \cdot 10^{-2}$**.**



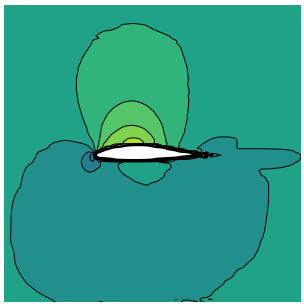
**(f)** $(0.6, 0.4)$**.** $RMSE = 1.3753 \cdot 10^{-2}$**,** $SSIM = 1.1072 \cdot 10^{-2}$**.**



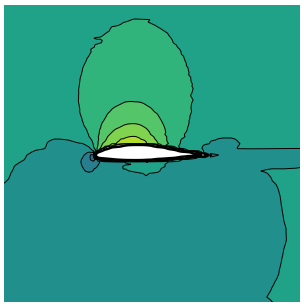
**(g)** $(0.5, 0.5)$**.** $RMSE = 1.0170 \cdot 10^{-2}$**,** $SSIM = 1.0645 \cdot 10^{-2}$**.**



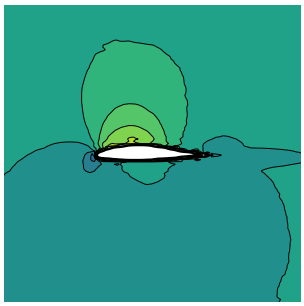
**(h)** $(0.4, 0.6)$**.** $RMSE = 1.1639 \cdot 10^{-2}$**,** $SSIM = 1.1456 \cdot 10^{-2}$**.**



**(i)** $(0.3, 0.7)$**.** $RMSE = 1.0935 \cdot 10^{-2}$**,** $SSIM = 1.0656 \cdot 10^{-2}$**.**



**(j)** $(0.2, 0.8)$**.** $RMSE = 1.0661 \cdot 10^{-2}$**,** $SSIM = 1.0504 \cdot 10^{-2}$**.**



**(k)** $(0.1, 0.9)$**.** $RMSE = 1.2345 \cdot 10^{-2}$**,** $SSIM = 1.1555 \cdot 10^{-2}$**.**

**Fig. 6 Truth and model prediction for the Mach numbers of a test case. The freestream conditions are** $AoA = 0.7558°$**,** $Re = 1.0624 \cdot 10^7$**,** $\mathcal{M} = 0.46393$**. (c) through (k) are the best 9 models listed in Table 3, which are denoted by their weight factors.**

**Table 4 Comparison of representative models.**

| Model | $RMSE$ $(10^{-2})$ | $1 - SSIM$ $(10^{-2})$ |
|---|---|---|
| Complex RMSE | 1.3141 | 1.2633 |
| Complex SSIM | 1.5367 | 0.7570 |
| Complex MS-SSIM, weights $(0.3, 0.7)$ | 1.3912 | 0.7370 |

### D. Comparison of Models

To further demonstrate our results, we present predictions of the complex RMSE model, the complex SSIM model, and the complex 2-scale SSIM model with weights $(0.3, 0.7)$. As the prediction is a stochastic process, it is important to note that a model with a lower RMSE on the entire testing set may perform worse than a model with a higher RMSE on the testing set for an arbitrary individual case. Therefore, to make the visual results as unbiased as possible, we find the case on which the models have a low normalized total standard deviation ($\hat{\sigma}$), defined as

$$\begin{aligned} \sigma_{RMSE} &= \sqrt{\frac{1}{3}\sum_{i=1}^{3}\left(\epsilon_{RMSE,i} - \bar{\epsilon}_{RMSE,i}\right)^2}, \\ \sigma_{SSIM} &= \sqrt{\frac{1}{3}\sum_{i=1}^{3}\left(\epsilon_{SSIM,i} - \bar{\epsilon}_{SSIM,i}\right)^2}, \\ \hat{\sigma} &= \sqrt{\hat{\sigma}_{RMSE}^2 + \hat{\sigma}_{SSIM}^2}, \end{aligned} \tag{4}$$

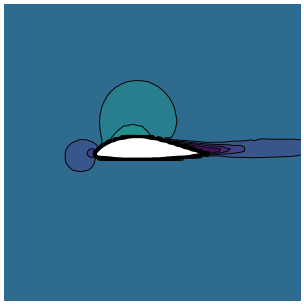
where $\bar{\epsilon}_{Err,i}$ is the "$Err$" ($Err \in \{RMSE, SSIM\}$) error of model $i$ on the testing set, $\epsilon_{Err,i}$ is the case-specific "$Err$" error; $\sigma_{Err}$ is the (absolute) standard deviation in the corresponding loss function ($Err \in \{RMSE, SSIM\}$) for the three models and $\hat{\sigma}_{Err}$ is the min-max normalized standard deviation. We need such a normalization to combine the two metrics of model quality because the two absolute $\sigma$'s are not of the same level of magnitude. Following Eqn. 4, the lower $\hat{\sigma}$ is, the closer the predictions of a specific model are to the average; thus, the more representative the case is. Figure 7 shows the predictions of the three models on the case with $\hat{\sigma} = 0.04455$, $\sigma_{RMSE} = 0.00086$, $\sigma_{SSIM} = 0.00058$. Figure 8 shows the absolute RMSE of the predictions in Fig. 7.

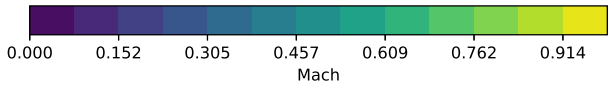
We observe that the plain RMSE model yields pixel-wise-accurate predictions, but they are plagued by spiky contours and bumps in supposedly flat regions. Switching to the SSIM loss function smoothens the contours while sacrificing a little accuracy. Introducing a second scale to the loss function and assigning appropriate weights improve the predictions further in terms of both the RMSE and SSIM. Figure 8 supports this observation in that using the SSIM loss eliminates bad predictions outside the wake region and that using the MS-SSIM loss with weight factors $(0.3, 0.7)$ further reduces the magnitudes of the error. We also list the models that appear in Fig. 7 in Table 4. We quantify our observations as follows: we improve the SSIM loss by 40.0% by switching from the RMSE to the SSIM loss function, at the cost of having a 14.5% higher RMSE. By introducing a second scale, it is possible to improve the RMSE by 9.5%, and the SSIM loss by 2.6%. Finally, compared with the best (complex) RMSE model, the complex MS-SSIM model has a 5.9% higher RMSE and a net 41.7% decrease in the SSIM loss.

## IV. Conclusion

This work shows that the MS-SSIM loss function improves the perceived visual quality of flow field predictions. Results show that the predictive performance of autoencoder-based CNN models increases with the increasing network architecture complexity. The accuracy metrics considered to measure predictive performance include root mean squared error (RMSE) and the structural similarity (SSIM) value. The MS-SSIM-based models predict smooth contours, which the RMSE models fail systematically. With the same neuron setup, the MS-SSIM model with appropriate weight factors achieves a much lower SSIM loss by slightly sacrificing the RMSE loss.

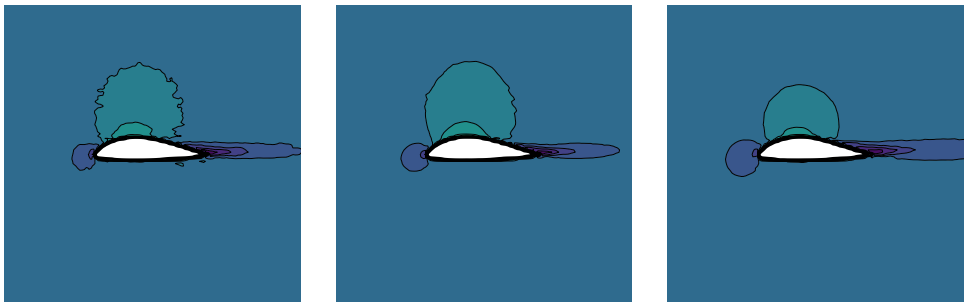
We see the following possible extensions for this work: (1) Our work has achieved satisfactory accuracy compared with the results reported in the work of Sekar *et al.* [13]. Compared with their architecture, ours have fewer hidden layers. Some hyperparameters such as the learning rate have not been systematically investigated. It is possible further to improve the accuracy of our models by fine-tuning. For example, the effects of sharing the decoder for various






**(a) Truth (from ADflow).**

**(b) The color bar.**

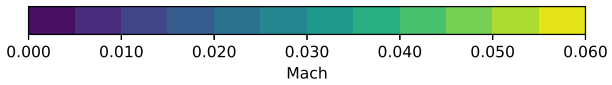


**(c) RMSE.** $RMSE = 1.1896 \cdot 10^{-2}$, $SSIM = 1.1255 \cdot 10^{-2}$.

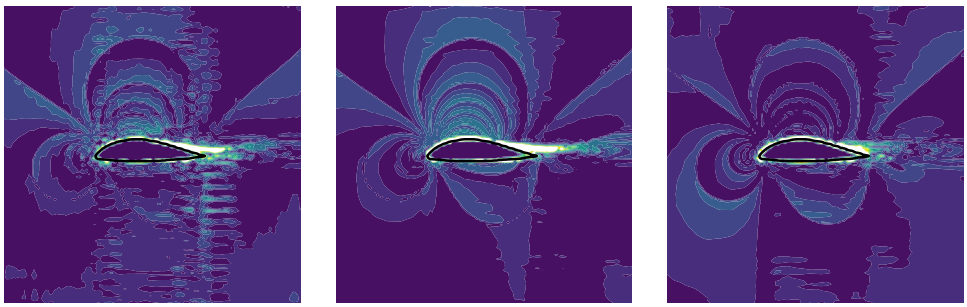
**(d) SSIM.** $RMSE = 1.5864 \cdot 10^{-2}$, $SSIM = 0.7050 \cdot 10^{-2}$.

**(e) MS-SSIM,** $(0.3, 0.7)$. $RMSE = 1.3338 \cdot 10^{-2}$, $SSIM = 0.6384 \cdot 10^{-2}$.

**Fig. 7 Truth and complex model predictions for the Mach numbers of a test case. The freestream conditions are** $AoA = 2.0612°$, $Re = 1.1055 \cdot 10^7$, $\mathcal{M} = 0.58986$. **(a) through (c) are predictions of the 3 models for comparison.**




**(a) The color bar for the absolute RMSE error.**



**(b) Absolute RMSE error for the complex RMSE model.**

**(c) Absolute RMSE error for the complex SSIM model.**

**(d) Absolute RMSE error for the complex MS-SSIM model with weight factors** $(0.3, 0.7)$.

**Fig. 8 Contours of the absolute RMSE error in the predictions of the models for comparison, with the contour of the airfoil overlaid. The freestream conditions are the same as in Fig. 7. Note that the contours of the absolute error use a color map different from the others in this paper; the maximum value is** $0.06$**, which is around 4 times the average RMSE.**

variables [17] is worth investigating. (2) The MS-SSIM loss function improved flow field prediction compared with the RMSE loss function. However, it still treats flow field solutions as generic images. It might be worthwhile to invent a new loss function that evaluates the structural similarity of a prediction and incorporates the physics of the flow.

## Acknowledgement

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper. URL: `http://www.tacc.utexas.edu`

## References

[1] Mader, C. A., Kenway, G. K. W., Yildirim, A., and Martins, J. R. R. A., "ADflow: An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization," *Journal of Aerospace Information Systems*, Vol. 17, No. 9, 2020, pp. 508–527. https://doi.org/10.2514/1.I010796.

[2] Jasak, H., Jemcov, A., and Tuković, Z., "OpenFOAM: A C++ Library for Complex Physics Simulations," *International Workshop on Coupled Methods in Numerical Dynamics, IUC*, Citeseer, 2007.

[3] He, P., Mader, C. A., Martins, J. R. R. A., and Maki, K. J., "An Aerodynamic Design Optimization Framework Using a Discrete Adjoint Approach with OpenFOAM," *Computers & Fluids*, Vol. 168, 2018, pp. 285–303. https://doi.org/10.1016/j.compfluid.2018.04.012.

[4] He, P., Mader, C. A., Martins, J. R. R. A., and Maki, K. J., "DAFoam: An Open-Source Adjoint Framework for Multidisciplinary Design Optimization with OpenFOAM," *AIAA Journal*, Vol. 58, No. 3, 2020. https://doi.org/10.2514/1.J058853.

[5] Martins, J. R. R. A., and Ning, A., *Engineering Design Optimization*, Cambridge University Press, 2021. URL https://www.researchgate.net/publication/352413464_Engineering_Design_Optimization.

[6] Peherstorfer, B., Willcox, K., and Gunzburger, M., "Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization," *SIAM Review*, Vol. 60, No. 3, 2018, pp. 550–591. https://doi.org/10.1137/16M1082469.

[7] Han, Z.-H., and Görtz, S., "Hierarchical Kriging Model for Variable-Fidelity Surrogate Modeling," *AIAA Journal*, Vol. 50, No. 9, 2012, pp. 1885–1896. https://doi.org/10.2514/1.j051354, URL https://doi.org/10.2514/1.j051354.

[8] Liem, R. P., Mader, C. A., and Martins, J. R. R. A., "Surrogate Models and Mixtures of Experts in Aerodynamic Performance Prediction for Aircraft Mission Analysis," *Aerospace Science and Technology*, Vol. 43, 2015, pp. 126–151. https://doi.org/10.1016/j.ast.2015.02.019.

[9] Angelova, A., Krizhevsky, A., and Vanhoucke, V., "Pedestrian detection with a Large-Field-Of-View deep network," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 704–711. https://doi.org/10.1109/ICRA.2015.7139256.

[10] Chan, W., Jaitly, N., Le, Q., and Vinyals, O., "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4960–4964. https://doi.org/10.1109/ICASSP.2016.7472621.

[11] Papila, N., Shyy, W., Griffin, L., and Dorney, D., "Shape Optimization of Supersonic Turbines Using Response Surface and Neural Network Methods," *39th Aerospace Sciences Meeting and Exhibit*, Vol. 18, 2001. https://doi.org/10.2514/6.2001-1065.

[12] Zhang, Y., Sung, W., and Mavris, D., "Application of Convolutional Neural Network to Predict Airfoil Lift Coefficient," 2017.

[13] Sekar, V., and Khoo, B., "Fast Flow Field Prediction over Airfoils using Deep Learning Approach," *Physics of Fluids*, Vol. 31, No. 5, 2019, pp. 057103(1)–057103(14).

[14] Bouhlel, M. A., He, S., and Martins, J. R. R. A., "Scalable gradient-enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes," *Structural and Multidisciplinary Optimization*, Vol. 61, 2020, pp. 1363–1376. https://doi.org/10.1007/s00158-020-02488-5.

[15] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P., "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, Vol. 86, 1998, pp. 2278 – 2324. https://doi.org/10.1109/5.726791.

[16] Guo, X., Li, W., and Iorio, F., "Convolutional Neural Networks for Steady Flow Approximation," 2016, pp. 481–490. https://doi.org/10.1145/2939672.2939738.

[17] 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.

[18] Pan, S., and Duraisamy, K., "Long-Time Predictive Modeling of Nonlinear Dynamical Systems Using Neural Networks," *Complexity*, Vol. 2018, 2018, pp. 1–26. https://doi.org/10.1155/2018/4801012.

[19] Czarnecki, W., Osindero, S., Jaderberg, M., Swirszcz, G., and Pascanu, R., "Sobolev Training for Neural Networks," 2017.

[20] Lee, S., and You, D., "Prediction of laminar vortex shedding over a cylinder using deep learning," 2017.

[21] Mathieu, M., Couprie, C., and LeCun, Y., "Deep multi-scale video prediction beyond mean square error," *CoRR*, Vol. abs/1511.05440, 2016.

[22] Wang, Z., Simoncelli, E. P., and Bovik, A. C., "Multiscale structural similarity for image quality assessment," *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, Vol. 2, 2003, pp. 1398–1402 Vol.2. https://doi.org/10.1109/ACSSC.2003.1292216.

[23] Zhao, H., Gallo, O., Frosio, I., and Kautz, J., "Loss Functions for Image Restoration With Neural Networks," *IEEE Transactions on Computational Imaging*, Vol. 3, No. 1, 2017, pp. 47–57. https://doi.org/10.1109/TCI.2016.2644865.

[24] Zhou Wang, Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P., "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, Vol. 13, No. 4, 2004, pp. 600–612. https://doi.org/10.1109/TIP.2003.819861.

[25] 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 27*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Curran Associates, Inc., 2014, pp. 2672–2680.

[26] 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 Forum*, AIAA, Orlando, FL, 2020. https://doi.org/10.2514/6.2020-2128.

[27] Lambe, A. B., and Martins, J. R. R. A., "Extensions to the Design Structure Matrix for the Description of Multidisciplinary Design, Analysis, and Optimization Processes," *Structural and Multidisciplinary Optimization*, Vol. 46, 2012, pp. 273–284. https://doi.org/10.1007/s00158-012-0763-y.

[28] Secco, N., Kenway, G. K. W., He, P., Mader, C. A., and Martins, J. R. R. A., "Efficient Mesh Generation and Deformation for Aerodynamic Shape Optimization," *AIAA Journal*, Vol. 59, No. 4, 2021, pp. 1151–1168. https://doi.org/10.2514/1.J059491.

[29] Kenway, G. K. W., Mader, C. A., He, P., and Martins, J. R. R. A., "Effective Adjoint Approaches for Computational Fluid Dynamics," *Progress in Aerospace Sciences*, Vol. 110, 2019, p. 100542. https://doi.org/10.1016/j.paerosci.2019.05.002.

[30] Kenway, G. K. W., Mader, C. A., He, P., and Martins, J. R. R. A., *CFD discrete adjoint benchmarks*, Mendeley Data, 2019. https://doi.org/10.17632/w4hsj8wzm8.

[31] Li, J., Bouhlel, M. A., and Martins, J. R. R. A., "Data-based Approach for Fast Airfoil Analysis and Optimization," *AIAA Journal*, Vol. 57, No. 2, 2019, pp. 581–596. https://doi.org/10.2514/1.J057129.

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

[33] Chen, W., Chiu, K., and Fuge, M., "Aerodynamic Design Optimization and Shape Exploration using Generative Adversarial Networks," *AIAA SciTech Forum*, San Diego, USA, 2019.

[34] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X., "TensorFlow: A system for large-scale machine learning," *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283. URL https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[35] Texas Advanced Computing Center (TACC), The University of Texas at Austin, "Stampede2 User Guide," , 10 2020.

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