Scalable gradient-enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes

Mohamed Amine Bouhlel[a], Sicheng He[a], Joaquim R. R. A. Martins[a]

[a] *University of Michigan, Department of Aerospace Engineering, Ann Arbor, MI, USA*

Abstract

Airfoil shape design is one of the most fundamental elements in aircraft design. Existing airfoil design tools require at least a few minutes to analyze a new shape and hours to perform shape optimization. To drastically reduce the computational time of both analysis and design optimization, we use machine learning to create a model of a wide range of possible airfoils at a range of flight conditions, making it possible to perform airfoil design optimization in a few seconds. The machine learning consists of gradient-enhanced artificial neural networks where the gradient information is phased in gradually. This new gradient-enhanced artificial neural network approach is trained to model the aerodynamic force coefficients of airfoils in both subsonic and transonic regimes. The aerodynamics is modeled with Reynolds-averaged Navier–Stokes (RANS) based computational fluid dynamics (CFD). The proposed approach outperforms an existing airfoil model that uses a mixture of experts technique combined with a gradient-based Kriging surrogate model. The approach yields to similar airfoil shape optimization solutions than high-fidelity CFD optimization solutions with a difference of 0.01 count and 0.12 count for C_d in subsonic and transonic regimes, respectively. Airfoil optimization problems are solved in a few seconds (instead of hours using CFD-based optimization), making the design process much more interactive, as demonstrated in the Webfoil airfoil design optimization tool.

Keywords: Surrogate Modeling, Artificial Neural Networks, Gradient-Enhanced Modeling, Airfoils

1. Introduction

Surrogate modeling methods are a powerful tool that have applications in engineering analysis and design tasks. If an outcome of interest is time-consuming to evaluate using numerical simulations, a fast surrogate model that approximates the output can be used instead. Many types of surrogate models exist in the literature, such as kriging [51, 48, 9, 8, 32, 17], radial basis functions (RBF) [46, 13, 24], and artificial neural networks (ANN) [26, 23]. Gradient-enhanced surrogate modeling methods use the gradient information to improve the accuracy of the model [34, 7]. These methods have been adapted to many surrogate model types, such as kriging [7, 5, 16] and RBF [57, 31].

Email addresses: mbouhlel@umich.edu (Mohamed Amine Bouhlel), hschsc@umich.edu (Sicheng He), jrram@umich.edu (Joaquim R. R. A. Martins)

Laurent et al. [34] provide an overview of existing gradient-enhanced modeling and their applications. However, most of the gradient-enhanced surrogate modeling methods do not scale well in number of samples (more than tens of thousands), which limits the achievable accuracy, especially for higher dimensional design spaces.

ANN have become increasingly popular due to the increased availability of training data and computing capability. One of the advantages of ANN is their scalability with the number of samples [21]. ANN are a computational model inspired by the way biological neural networks in the human brain process information and were originally invented by McCulloch and Pitts [43]. Goodfellow et al. [21] describes the evolution of ANN, which have had a history that has alternated between progress and stagnation. Today, ANN are used in many practical applications and active research topics such as autonomous vehicles [2], computer vision [3], and speech recognition [30].

ANN have been also used in aerospace applications. Rai and Madavan [47] used ANN and a polynomial model to approximate the pressure distribution of a turbine blade in the subsonic regime involving 15 design variables. They adapted the neural net dynamically during the optimization, which made it time-consuming. To manage the trade-off between accuracy and efficiency, they based the surrogate model on Euler data at the initial stage of optimization and switched to RANS data in the final stage. Papila et al. [45] optimized the performance of a supersonic turbine using radial basis neural networks combined with polynomial models using data generated by an unsteady RANS solver. The ANN was trained with the CFD data and was then used to generate more data points for the polynomial model. Zhang et al. [56] analyzed the image of an airfoil and estimated its lift coefficient using convolutional neural networks with training data generated from a RANS solver. Xu et al. [53] used ANN as a surrogate relating an airfoil geometry and its buffeting magnitude, frequency and time-average aerodynamic load coefficients based on unsteady RANS simulations.

ANN have been extended to multi-information sources modeling. However, there has been little research on ANN that use gradient information to solve engineering problems. Giannakoglou et al. [19] analyzed a number of 2D and 3D aerodynamic shape design problems using gradient-enhanced ANN. Their error metric was a combination of function error and gradient error. They used an Euler equation model and thus viscous effects were neglected. In addition, only low Mach numbers (up to 0.5) were considered.

Liu and Batill [38] developed a gradient-enhanced ANN method that does not require weighting the residuals of the targets and gradients, but they only solved analytic benchmarks with up to 15 input variables. Pan and Duraisamy [44] used ANN to model nonlinear dynamical systems with gradient information. In that work, the gradient was penalized to obtain a smooth response instead of matching the gradient of the ANN model with the gradient of the data. Another approach using gradient-enhanced ANN was developed by Czarnecki et al. [15], where the loss function was defined as a weighted sum of the output estimation error and derivatives estimation error.

Airfoil shape design is one of the most fundamental elements in aerospace engineering. Given the current computer power, it is possible to compute steady flow for airfoils using CFD to solve RANS equation in minutes, which opens the door to performing the thousands of airfoil solutions required for machine-learning processes. Furthermore, since the adjoint method efficiently computes the gradient of performance metrics with respect to shape and flow variables efficiently, the computed gradients provide an additional rich

set of data. For a cost that is equivalent to less than a flow solution, the adjoint method provides a potentially large vector of data.

ANN frameworks that operate at large scale and in heterogeneous environments have improved dramatically in the last few years and are easier to use than ever. TensorFlow for example, has a high-level Python interface that has become popular owing to the combination of power and relative ease of installation and use [1].

Enabled by the above developments, we developed a new gradient-enhanced ANN approach and applied it to airfoil shape design optimization. We compared this newly-developed approach to the surrogate-model approach developed by Li et al. [35] for the prediction of the drag, lift, and moment coefficients. The surrogate-model approach required the division of the airfoil shape and flight condition into two flow regimes (subsonic and transonic) because the surrogate modeling techniques could not simultaneously achieve acceptable accuracy for both regimes. The ANN approach developed herein, however, managed to model both subsonic and transonic airfoils in the whole flight regime with one single model.

The remainder of the manuscript is structured as follows. We describe the gradient-enhanced ANN and our improvements in Section 2. We present the results in Section 3, where we compare our previous surrogate model approach against the new gradient-enhanced ANN model. Finally, we present two airfoil shape optimization problems for the subsonic and transonic regimes in Section 4.

2. Gradient-enhanced ANN

In principle, using the training derivatives in addition to the training output values should improve the accuracy of the output prediction in surrogate modeling. Using this additional information should be even more advantageous when the computation of the gradients is efficient.

Many physics models now include adjoint methods. ADflow computes gradient information using the adjoint method, which computes the derivatives of a function of interest with respect to input variables at a cost that is independent of the number of those input variables [41, 33]. Similarly, DAFoam [1] which is developed based on openFoam, also has an adjoint implementation [27, 52]. This is in contrast to other gradient computation methods, such as finite-difference and complex-step methods, which scale linearly with the number of input variables [42].

In this work, we use ADflow, an open-source RANS CFD solver that includes an adjoint method and efficient aerodynamic shape optimization [54, 33]. ADflow has been successfully used to perform aerodynamic shape optimization for airfoils [29, 35], wings [39, 14, 55, 6, 40], and full aircraft configurations [14, 50, 49]. ADflow is also coupled to other disciplines to solve multidisciplinary design optimization problems [36, 11, 22, 18, 12, 28].

Our approach to exploiting the gradient information is based on the gradient-enhanced ANN method developed by Czarnecki et al. [15]. They introduced the Sobolev training for artificial neural networks (SANN) method that incorporates the gradient information in the loss function with the training samples during the training phase.

[1] https://github.com/mdolab/adflow

2.1. Description of SANN

To approximate a function f using SANN, we use the matrix of training input values

$$\mathbf{X} = \left[\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n_t)}\right]^T, \tag{1}$$

where each training input is

$$\mathbf{x}^{(i)} = \left[x_1^{(i)}, \ldots, x_d^{(i)}\right]^T, \tag{2}$$

and hence $\mathbf{X}$ is a $(n_t \times d)$ matrix, where d is the number of dimensions of the input and n_t is the number of training samples. The corresponding vector of training output values is

$$\mathbf{y} = \left[y^{(1)}, \ldots, y^{(n_t)}\right]^T, \tag{3}$$

where $y^{(i)} = f\left(\mathbf{x}^{(i)}\right)$. Then, the gradient of the function that computes the outputs is

$$\nabla f\left(\mathbf{x}^{(i)}\right) = \left[\frac{\partial f}{\partial x_1}\left(\mathbf{x}^{(i)}\right), \ldots, \frac{\partial f}{\partial x_d}\left(\mathbf{x}^{(i)}\right)\right]^T, \tag{4}$$

which we write as a matrix when evaluated for each training point as

$$\mathbf{df} = \nabla f(\mathbf{X}) = \left[\nabla f\left(\mathbf{x}^{(1)}\right), \ldots, \nabla f\left(\mathbf{x}^{(n_t)}\right)\right]^T. \tag{5}$$

To train a neural network model m using a set of parameters θ, we minimize a loss function with respect to θ, i.e.,

$$\begin{aligned} &\min_{\theta} y_{\text{loss}}(\mathbf{X}, \mathbf{y}|\theta), \\ &\quad y_{\text{loss}}(\mathbf{X}, \mathbf{y}|\theta) \coloneqq l\left(m\left(\mathbf{x}^{(i)}|\theta\right), y^{(i)}\right), \end{aligned} \tag{6}$$

where l is a loss function.

To incorporate the gradients into the training of an ANN, Czarnecki et al. [15] proposed adding a second term to the loss function above

$$\begin{aligned} &\min_{\theta} \bar{y}_{\text{loss}}(\mathbf{X}, \mathbf{y}, \mathbf{df}|\theta), \\ &\quad \bar{y}_{\text{loss}}(\mathbf{X}, \mathbf{y}, \mathbf{df}|\theta) \coloneqq l\left(m\left(\mathbf{x}^{(i)}|\theta\right), y^{(i)}\right) + \sum_{j=1}^{d} l_j\left(\frac{\partial m}{\partial x_j}\left(\mathbf{x}^{(i)}|\theta\right), \frac{\partial f}{\partial x_j}\left(\mathbf{x}^{(i)}\right)\right), \end{aligned} \tag{7}$$

where l_j is the loss function with respect to the j^{th} partial derivative. In this equation, the second term involving the derivatives is dominant in the training of the SANN model, especially when d is high. This slows down the convergence of the SANN training because it emphasizes the minimization of the gradient error.

2.2. Efficient approach for training SANN

To accelerate the training convergence, we propose to introduce the gradient information gradually during the learning process. We do this by incorporating a parameter λ_k,

with $k \in \mathbb{N}^*$, in Equation (7), yielding

$$\begin{aligned} &\min_{\theta} \bar{y}_{\text{loss}}(\mathbf{X}, \mathbf{y}, df|\theta), \\ &\bar{y}_{\text{loss}}(\mathbf{X}, \mathbf{y}, \mathbf{df}|\theta) \coloneqq \sum_{i=1}^{n_t} l\left(m\left(\mathbf{x}^{(i)}|\theta\right), y^{(i)}\right) + \\ &\lambda_k \sum_{j=1}^{d} l_j\left(\frac{\partial m}{\partial x_j}\left(\mathbf{x}^{(i)}|\theta\right), \frac{\partial f}{\partial x_j}\left(\mathbf{x}^{(i)}\right)\right), \end{aligned} \tag{8}$$

where the parameter $\lambda_k \in [0, 1]$ is a weight that controls how much gradient information to use in the SANN training.

We use TensorFlow [1], which is a large-scale machine learning framework, to compute the derivative $\partial m/\partial x_j\left(\mathbf{x}^{(i)}|\theta\right)$. TensorFlow uses reverse mode automatic differentiation to compute the derivatives.

We call this approach modified SANN (mSANN). The key idea is to gradually increase λ for each epoch in the training. Algorithm 1 describes the main steps in mSANN.

Algorithm 1: mSANN

Input: $(\mathbf{X}, \mathbf{y}, \mathbf{df}, N)$; `// N is the total number of epochs`
Define the ANN architecture; `// Layers, units, and activation functions`
Define λ_k; `// List containing values of λ`
Initialize θ in Equation (8); `// Use a normal random variable` $\mathcal{N}(0, 0.1)$
for $i \leq N$ **do**
  for $\lambda \in list(\lambda_k)$ **do**
    Solve minimization problem (8); `// Use backpropagation algorithm`
  end
end
Output: $\hat{y}(\mathbf{x})$; `// Final prediction`

For a better illustration of the benefits of this approach, we use the Rosenbrock function

$$\sum_{i=1}^{15} \left[(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right], \quad -2 \leq x_i \leq 2, \quad \text{for } i = 1, \ldots, 16. \tag{9}$$

to compare the performance of our approach with SANN and a standard ANN method without gradients.

We first build similar ANN architectures for all methods using eight layers: one input layer, six hidden layers, and one output layer. Figure 1, which is generated using the NN-SVG online tool [2], depicts the architecture and activation functions of the ANN. In general, there is no rule of thumb to know in advance the best ANN architecture . In our case, the architectures are chosen based on trial and error after many experiments. Similar to the ANN architecture, the tuned parameters for training an ANN model, such

[2] http://alexlenail.me/NN-SVG/LeNet.html

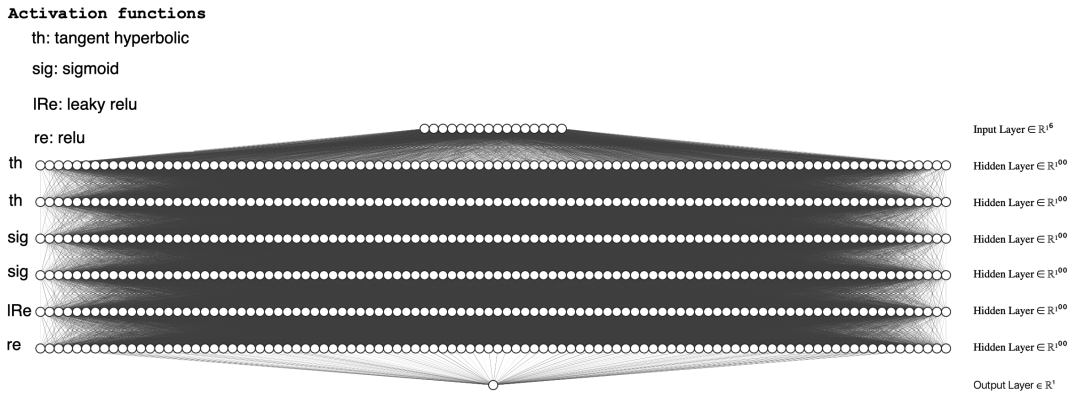


Figure 1: This ANN architecture is composed of one input layer, six hidden layers, and one output layer. The activation functions are indicated next to the left edge of each hidden layer. The number of units for each hidden layer is 100.

as the learning rate, the number of layers, the number of units, and so on, is problem-dependent. Tuning the parameters λ_k for solving Equation (8) is also problem-dependent. In our case, we set $\lambda_k = 0, 0.1, 0.2, \ldots, 1$. We use TensorFlow to construct the ANN, SANN, and mSANN models.

After building the ANN, we sample 42,000 training points and 22,000 validation points using the Latin hypercube sampling method implemented by Bouhlel et al. [10]. Then, we use the L_2-norm ($||.||_2$) for both l and l_j. Finally, we use three criteria to compare between the various ANNs:

- The y_{loss} and $\bar{y}_{\text{loss}}$ for each epoch at the training samples, which shows the evolution of the training process. The evolution of $\bar{y}_{\text{loss}}$ is only applicable for SANN and mSANN.

- The L_2-norm of relative errors (ϵ_{L_2}) for both training and validation samples

$$\epsilon_{L_2} = \frac{||\hat{\mathbf{y}} - \mathbf{y}||_2}{||\mathbf{y}||_2} 100, \tag{10}$$

  where $\mathbf{y}$ is a vector containing the real values and $\hat{\mathbf{y}}$ is a vector containing the prediction of $\mathbf{y}$. This provides a measure of the final solution accuracy.

- The L_2-norm of relative errors ($\bar{\epsilon}_{L_2}$) of the gradient predictions for both training and validation samples

$$\bar{\epsilon}_{L_2} = \frac{||\,\mathrm{d}\hat{\mathbf{y}} - \mathrm{d}\mathbf{y}||_2}{||\,\mathrm{d}\mathbf{y}||_2} 100, \tag{11}$$

  where $\mathrm{d}\mathbf{y}$ is a vector containing the real values of the gradient and $\mathrm{d}\hat{\mathbf{y}}$ is a vector containing the prediction of $\mathrm{d}\mathbf{y}$. The vectors $\mathrm{d}\mathbf{y}$ and $\mathrm{d}\hat{\mathbf{y}}$ are shaped in a way to contain all the derivative values into one vector column; i.e., $\mathrm{d}\mathbf{y}$ and $\mathrm{d}\hat{\mathbf{y}}$ are both vectors for training and validation, respectively.

Figure 2 compares the results of ANN, SANN and mSANN models. The y_{loss} and $\bar{y}_{\text{loss}}$ loss functions drop off rapidly with mSANN especially for the first epochs (an epoch in

Table 1: The L_2-norm of the relative errors results given by Equations (10) and (11) for both training and validation on the Rosenbrock function.

	Training		Testing	
	ϵ_{L_2}	$\bar{\epsilon}_{L_2}$	ϵ_{L_2}	$\bar{\epsilon}_{L_2}$
mSANN	1.20	5.27	1.25	5.33
SANN	4.00	14.18	4.06	14.20
ANN	4.40	25.50	4.78	25.60

ANN is one forward pass and one backward pass of all the training examples). In addition, mSANN shows a better convergence in all cases by at least two orders of magnitude. The final solutions obtained are: 8.88 (mSANN), 11.30 (SANN), and 11.52 (ANN) for y_{loss} and 11.57 (mSANN) compared to 13.60 (SANN) for $\bar{y}_{\text{loss}}$.

As shown in Table 1, mSANN outperforms ANN and SANN in both the prediction of the output of interest and the prediction of the gradient for training and validation samples.

3. Application to airfoil analysis

As previously mentioned, Li et al. [35] developed a data-based approach to provide fast aerodynamic analysis and design optimization tool for airfoils in the subsonic and transonic flow regimes using two separate surrogate models for each of these flow regimes. In this section, we first describe the methodology followed by Li et al. [35]. Then, we describe our approach for constructing a fast airfoil analysis and design tool in both subsonic and transonic using a single mSANN model. We compare the performance of the mSANN model with the results given of Li et al. [35] for the subsonic case and demonstrate the performance of mSANN in the transonic regime. We compare the mSANN to the GE-KPLS method only on particular airfoil cases in transonic regime. This is because Li et al. [35] developed a surrogate model in a small design space in transonic regime while our developed method covers a much larger design space.

3.1. Data-based approach in subsonic flow regime

The data-based approach developed by Li et al. [35] used a new airfoil shape parameterization defined by camber-thickness mode shapes. They first selected 1172 subsonic airfoils from the Webfoil database [3] to cover as large an airfoil design space as possible. Then, they computed the camber-thickness mode shapes of these airfoils using the singular value decomposition method. Seven camber modes and seven thickness modes were sufficient to accurately regenerate the initial geometry of the airfoils. This reduces the number of the airfoil shape parameters to 14 while covering a large design space with different geometry complexities of airfoils. The free-stream Mach number (M) was allowed to vary between 0.3 and 0.6 while the angle of attack (α) ranged from -2 to 6 degrees.

They generated new sampling airfoils data into many clusters (defined by dividing the design space into sub-domains with respect to M, α, one aerodynamic coefficient, and the first camber and thickness modes) with well-defined bounds of mode shapes.

[3] http://webfoil.engin.umich.edu

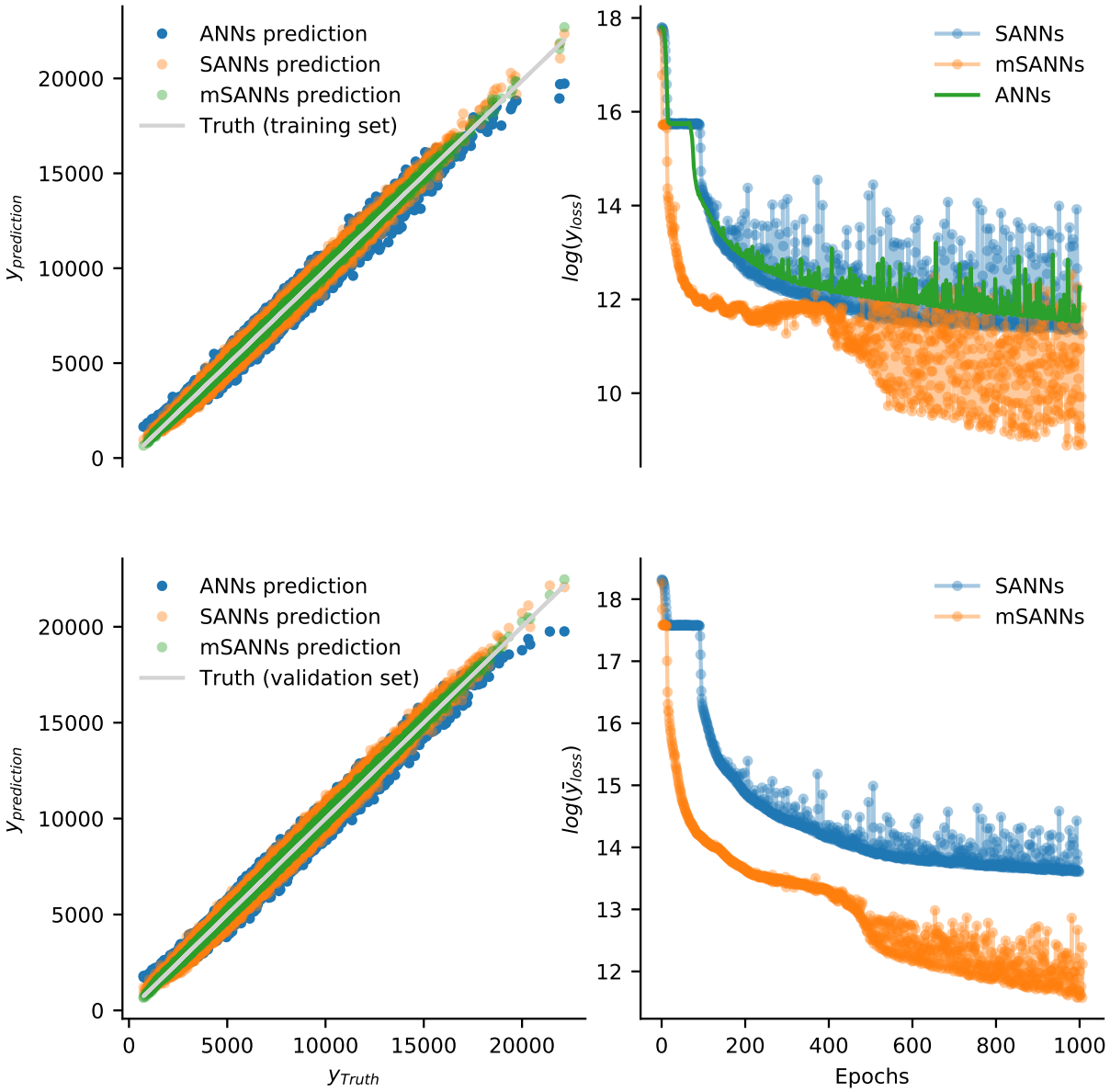


Figure 2: Rosenbrock function results. Upper left: comparison between the training predictions of ANN, SANN, and mSANN. The best predictions are close to the diagonal (black line). mSANN shows a better prediction on the training samples than both ANN and SANN. Bottom left: Comparison between the validation predictions. Similar to the training predictions, we obtain better predictions using mSANN. Upper right: The mSANN model reduces the loss function faster than ANN and SANN especially during the first iterations, and yields to a better convergence. Bottom right: mSANN is faster and exhibits better convergence than SANN.

The airfoil aerodynamic force coefficients and their respective gradients are computed using ADflow, which solves the RANS equations with a Spalart–Allmaras turbulence model [54].[4] The RANS CFD meshes for all airfoils are automatically generated with the open-source package pyHyp. [5] The flow solution residuals are converged by 15 orders of magnitude below the initial residual using an approximate Newton–Krylov method at the initial stage and then switching to an exact Newton–Krylov method at the final stage

Since the gradient information is required for the construction of the gradient-enhanced surrogate model, the ADflow adjoint solver [33] is also called to compute the gradient with respect to all modes, Mach number, and angle of attack. Later we also use this data set to train the Neural Network presented in Section 3.2.

Finally, Li et al. [35] used a mixture of experts method [25, 4] with three main steps:

1. Train a local gradient-enhanced surrogate model for each cluster, using gradient-enhanced kriging combined with partial-least-squares model (GE-KPLS) [7]. This surrogate model scales well in inputs and it is among the few gradient-enhanced kriging-based surrogates that scale well with training data.

2. Compute the cluster posterior probability for each cluster with a supervised learning algorithm to compute the proportion of each cluster using a regularized Gaussian classifier Liem et al. [37].

3. Build a surrogate model as a linear combination of all local surrogate models using the mixing proportion computed in Step 2.

For simplicity, this methodology is named GE-KPLS in this section because it is based on the GE-KPLS model.

3.2. mSANN approach for both subsonic and transonic regimes

Li et al. [35] used two different shape parametrizations for the subsonic and transonic airfoils. Instead of the seven camber and seven thickness mode shapes used for the subsonic airfoils, the shape parameterization for the transonic airfoils used four camber modes and four thickness modes, which have no relation to the subsonic ones.

Since we want to use both databases to create a single model using the mSANN approach, we need to find a way to deal with the distinct sets of mode shapes. The common inputs in subsonic and transonic databases are M and α. To fuse both parameterizations, we developed a sparse geometry parametrization. For each regime, we have matrices of data ($\mathbf{X}$) with size ($n \times d$), where n is the number of samples and d is the number of input variables (two flow conditions plus the camber and thickness mode coefficients). We decompose each matrix as $\mathbf{X} = [\mathbf{X}_{\text{ct}}, \mathbf{X}_{\alpha M}]$ to separate the values for the mode coefficients (ct) and flow conditions (αM). Then, we merge the matrices corresponding to the subsonic (sub) and transonic (trans) data into one matrix with a size of $(n_{\text{sub}} + n_{\text{trans}}) \times (d_{\text{sub}} + d_{\text{trans}} - 2)$ containing both subsonic and transonic data to yield

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_{\text{ct}_{\text{trans}}} & \mathbf{0} & \mathbf{X}_{\alpha M_{\text{trans}}} \\ \mathbf{0} & \mathbf{X}_{\text{ct}_{\text{sub}}} & \mathbf{X}_{\alpha M_{\text{sub}}} \end{bmatrix}, \tag{12}$$

[4] https://github.com/mdolab/adflow
[5] https://github.com/mdolab/pyhyp

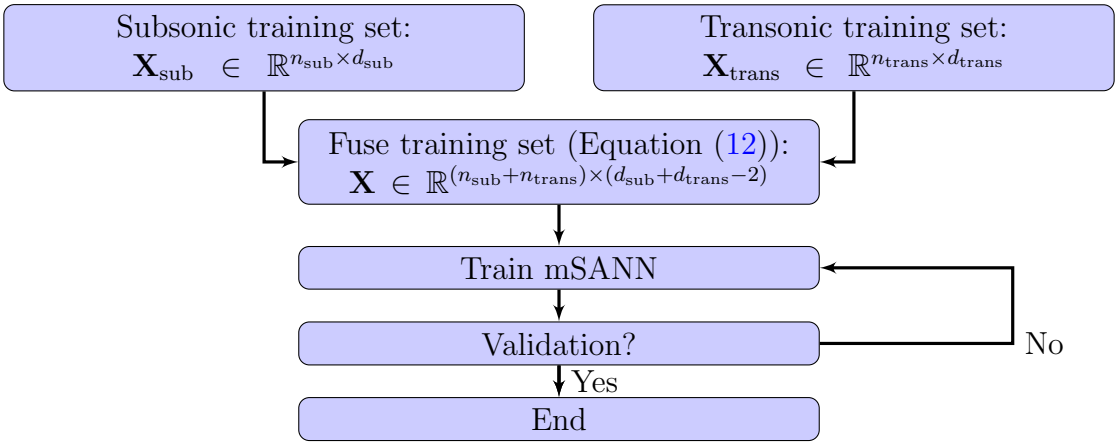


Figure 3: Workflow of training mSANN for both subsonic and transonic regimes.

where the two matrices of zeros have the appropriate sizes to fill the corresponding blocks. The flowchart in Figure 3 shows the workflow of training mSANN for both the subsonic and transonic regimes. First, we set both the subsonic and transonic training sets. Second, we fuse both training sets into a single training set $\mathbf{X}$ using Equation (12). Third, we run the training of mSANN on $\mathbf{X}$. Finally, we check the validation of the model: if the desired accuracy is reached we stop the training, otherwise, we continue the training of the model.

Our objective is to train an mSANN model using both subsonic and transonic databases to predict the aerodynamic force and moment coefficients: C_l, C_d, and C_m.

3.3. Validation procedure

We use a mSANN architecture similar to the one shown in Figure 1 with $\lambda_k = 0, 0.1, 0.2, \ldots, 0.9, 1$. There are 42,039 and 4,120 training samples for the subsonic and transonic regimes, respectively. The training database used for mSANN in the subsonic regime is a subset of the one used for GE-KPLS by Li et al. [35], but the testing and validation samples are the same. To validate the performance of the mSANN model, we use 21,870 and 1,544 validation samples for subsonic and transonic regimes, respectively. To verify the generalization of the model, the validation set is split into two smaller sets: 10,935 validation samples and the same number of test samples for the subsonic regime, and 787 validation samples and 757 test samples for the transonic regime. If the accuracy of the test set is sufficient, we stop the training; otherwise, we continue the training. For the subsonic set, $M \in [0.3, 0.6]$ and $\alpha \in [-0.5, 6]$ degrees, while for the transonic set $M \in [0.7, 0.75]$ and $\alpha \in [-1.5, 4.5]$ degrees.

To assess the performance of the mSANN model, we compute the L_2-norm of the relative error (10) for both validation and test sets. We also compute the L_1-norm of the relative error of each sample (for both validation and test samples), which is given by

$$\epsilon^i_{L_1} = \left| \frac{y^{(i)}_{\text{pred}} - y^{(i)}_{\text{true}}}{y^{(i)}_{\text{true}}} \right|. \tag{13}$$

In the subsonic regime, we compare the global performance of mSANN to GE-KPLS on the same testing and and validation sample sets. In addition, we show the performance of only mSANN in transonic regime. In addition, we compare the aerodynamic

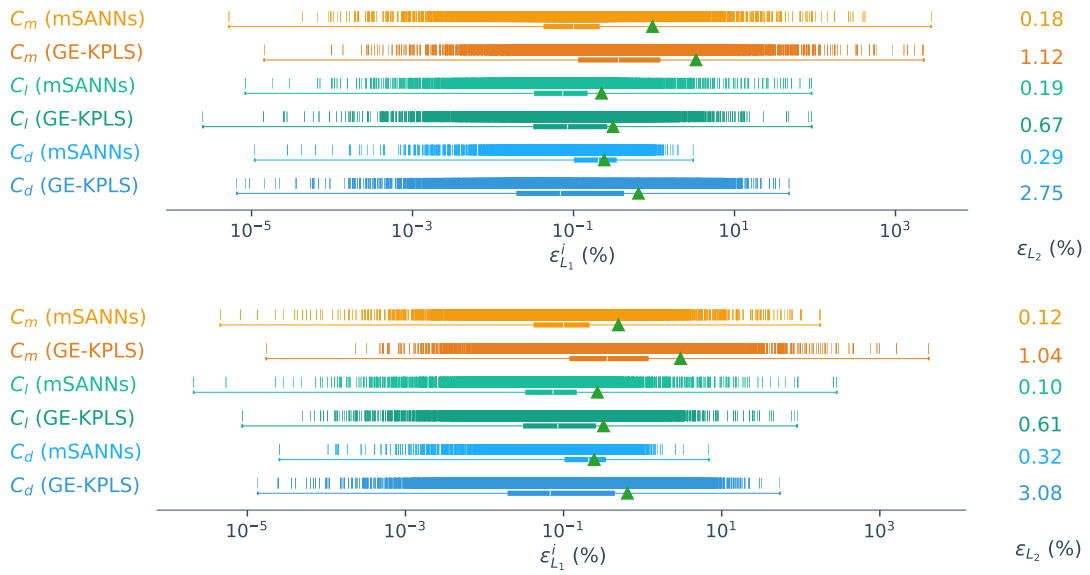


Figure 4: Comparison of the L_1-norm of the relative errors and the L_2-norm of the relative errors between GE-KPLS and mSANN for validation (top) and testing (bottom) in subsonic flow regime. The mean of the L_1-norm of the relative errors are given by the triangles. The white line in the box represents the median of the L_1-norm of the relative errors. mSANN yields to better results than the GE-KPLS model, especially for the drag coefficient.

coefficients curves versus angle of attack between the GE-KPLS and mSANN models on three selected subsonic airfoils with different geometry complexities. For transonic airfoil optimization, Li et al. [35] optimized three transonic airfoils with $M = 0.73$. We compare the performance of mSANN to GE-KPLS on the same transonic airfoils used by Li et al. [35].

3.4. Comparison between GE-KPLS and mSANN in subsonic flow regime

GE-KPLS and mSANN are compared in Figure 4. The mSANN model yields to better results than GE-KPLS for all aerodynamic coefficients for both validation and testing, especially for C_d (e.g.; 0.29% error for mSANN versus 2.75% for GE-KPLS in terms of ϵ_{L_2}). For C_d in the test samples, there is an improvement in maximum $\epsilon^i_{L_1}$ of about 1.5 orders of magnitude. Moreover, mSANN yields a better mean of the L_1-norm of the relative errors in all cases, especially for C_m, where there is an improvement of almost one order of magnitude. This shows that gradient-enhanced neural networks are able to learn complex models with different physical behavior over the design space.

We select three airfoils of different types that do not exist in the database used for training the model: NACA 4412 (max thickness 12% at 30% chord and max camber 4% at 40% chord), Clark Y (max thickness 11.7% at 28% chord and max camber 3.4% at 42% chord), and NLF(1)-1015 (max thickness 15% at 39.8% chord and max camber 4.3% at 62.8% chord). Figure 5 shows the airfoils shapes and their aerodynamic coefficient curves. For C_l and C_d, the mSANN and GE-KPLS curves match all the CFD simulations. For C_m, mSANN outperforms GE-KPLS on both NACA 4412 and NLF(1)-1015, especially for the last airfoil, where the GE-KPLS over-predicts compared to the CFD simulations. For example, we have an error of 0.01% with mSANN versus an error of 1.91% with

GE-KPLS for NLF(1)-1015 at $\alpha = 5.87$. In addition, the mSANN curve is smoother compared to GE-KPLS in both cases. However, mSANN is slightly worse than GE-KPLS in predicting C_m near $\alpha = 6°$ for the first two airfoils. Indeed, for NACA 4412 for example, we observe an error of 0.4% and 0.6% with mSANN versus an error of 0.07% and 0.1% with GE-KPLS at $\alpha = 5.55$ and 5.91, respectively.

3.5. Performance of mSANN in transonic flow regime

The distribution of the $\epsilon^i_{L_1}$ and ϵ_{L_2} values are given in Figure 6. The L_2-norm of the relative errors of the model increases slightly compared to the subsonic results given in Figure 4. However, the accuracy of the mSANN model is within the same order of magnitude for both subsonic and transonic regimes.

In addition, the results of the test and validation in terms of ϵ_{L_2} are approximately the same: the L_2-norm of the relative errors on the test samples is slightly larger than the error on the validation samples (with a difference of 0.07% for both C_d and C_m). On the other hand, the means of the L_1-norm of the relative errors are very similar in this case. In sum, our approach generalizes well in both subsonic and transonic regimes.

Figure 7 compares GE-KPLS and mSANN for three transonic NASA airfoils: SC(2)-0710 (max thickness 10% at 37% chord and max camber 2.1% at 81% chord), SC(2)-0606 (max thickness 6% at 34% chord and max camber 1.3% at 81% chord), and SC(2)-0404 (max thickness 4% at 37% chord and max camber 0.7% at 81% chord). The mSANN model outperforms GE-KPLS in all cases, especially in predicting C_m, where GE-KPLS mismatches most of the CFD simulations. In addition, mSANN provides smoother curves, especially when we compare mSANN to GE-KPLS for C_m with SC(2)-0710.

These results show the ability of our approach to approximate two different design spaces (subsonic and transonic) simultaneously.

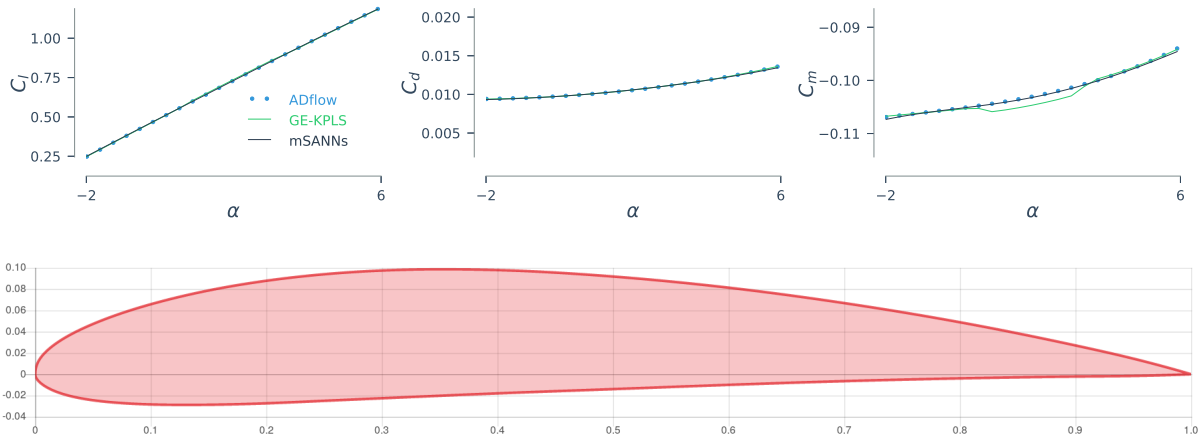
4. Airfoil shape optimization

Using the mSANN model trained as described in Section 3.2, we perform two airfoil shape optimizations in the subsonic and transonic flow regimes. We couple the mSANN model to SNOPT (Sparse Nonlinear OPTimizer), a gradient-based optimizer that implements sequential quadratic programming and solves large-scale constrained optimization problems efficiently [20]. To verify the performance of our optimization results, we compare the solutions provided by the mSANN-based optimization to optimizations using direct the CFD evaluations.

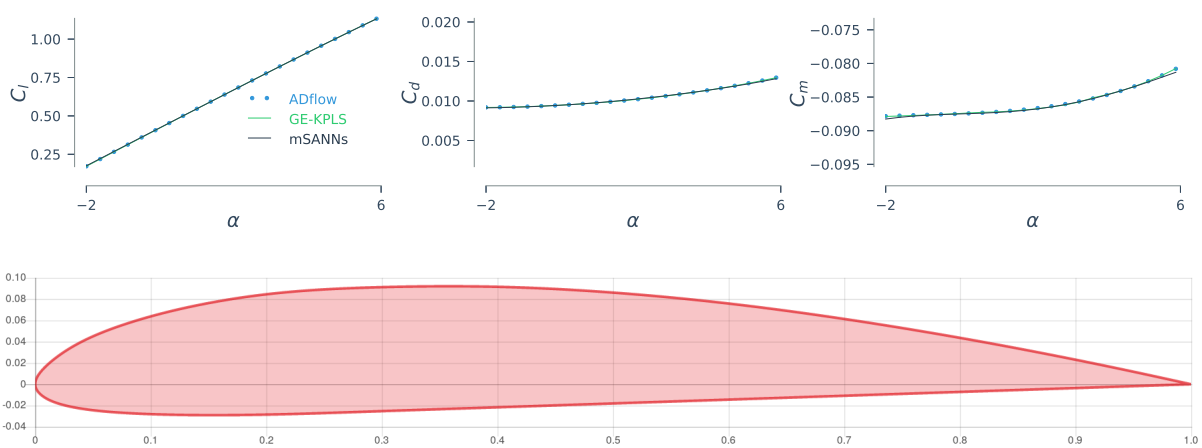
4.1. Subsonic airfoil optimization

In the subsonic case, we minimize the drag coefficient with respect to 14 modes and angle of attack (fixing $M = 0.45$) subject to a constraint $C_l = 0.5$, starting from the NACA 0012 airfoil shape. To avoid abnormal airfoils, we constraint the upper bound of the first thickness mode to be no more than 90% of the NACA 0012 first thickness mode. The remaining 13 mode shape bounds are computed as described by Li et al. [35]. Therefore, the total number of constraints is 15 (14 constraints on the mode coefficients and 1 constraint on C_l).

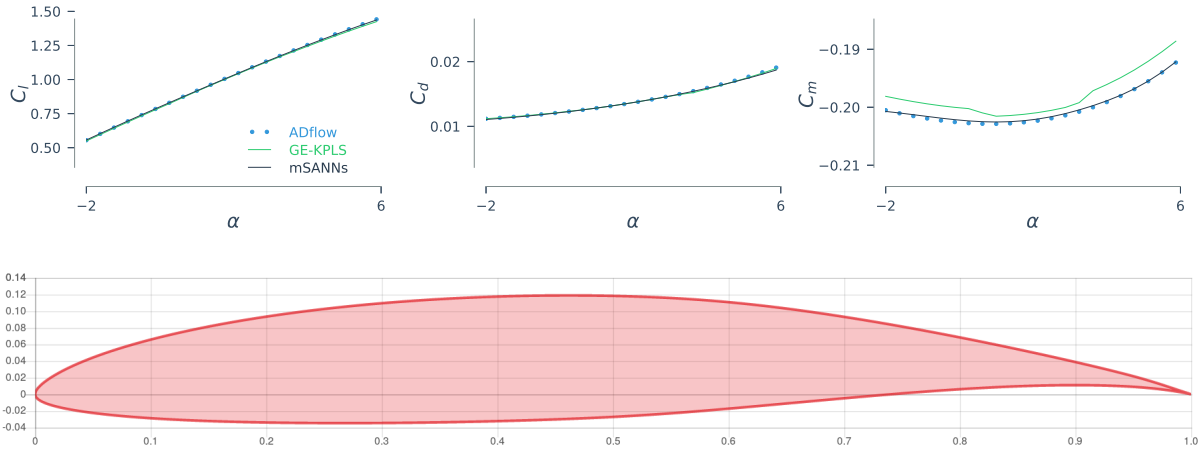
Figure 8 compares the solution found by mSANN with the CFD reference solution. The grey area shows the design space range. In this case, the optimized airfoils with



(a) NACA 4412 airfoil



(b) CLARK Y airfoil



(c) NLF(1)-1015 airfoil

Figure 5: Comparison of aerodynamic coefficient curves between mSANN and GE-KPLS in subsonic ($M = 0.45$) for three airfoils. The results of mSANN and GE-KPLS models are similar and match the CFD simulations, except for C_m, where mSANN performs better overall.

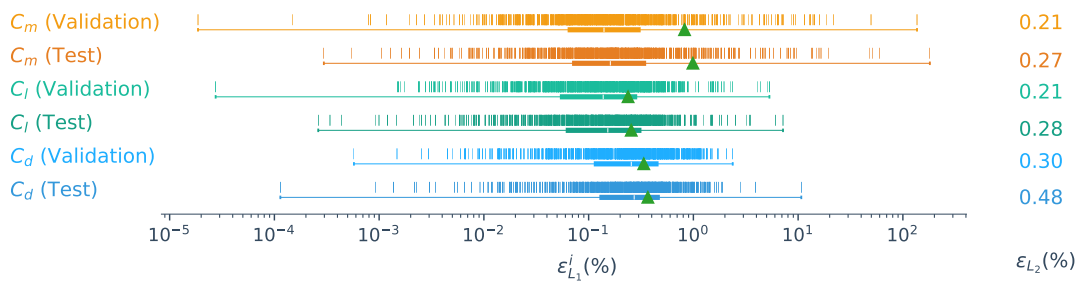


Figure 6: Results of the L_1-norm of the relative errors and the L_2-norm of the relative errors for validation and test samples in transonic. The mean of the L_1-norm of the relative errors are shown by the triangles. The white line in the box represents the median of the L_1-norm of the relative errors.

mSANN and CFD are almost identical. When comparing the CFD-computed coefficients, C_d differs by only 0.01 counts and C_l by 0.001.

4.2. Transonic airfoil optimization

In the transonic case, we minimize the C_d coefficient with respect to eight mode shapes and angle of attack at $M = 0.72$ subject to a lift constraint set to $C_l = 0.82$. Similar to the subsonic case, we constrain the upper bound of the first thickness mode to be no more than 90% of the SC(2)-0710 first thickness mode. The remaining seven mode shape bounds are computed in a similar way to the subsonic case. Therefore, the total number of constraints is 9 (8 constraints on the mode coefficients and 1 constraint on C_l).

Figure 9 compares the two optimal airfoils. Transonic airfoil shape optimization is more challenging than subsonic airfoil shape optimization due to the much more nonlinear flow physics. Nevertheless, the mSANN-based optimization achieves an airfoil similar to the reference one. The C_d differs by only 0.12 counts and the difference in C_l is only 0.002.

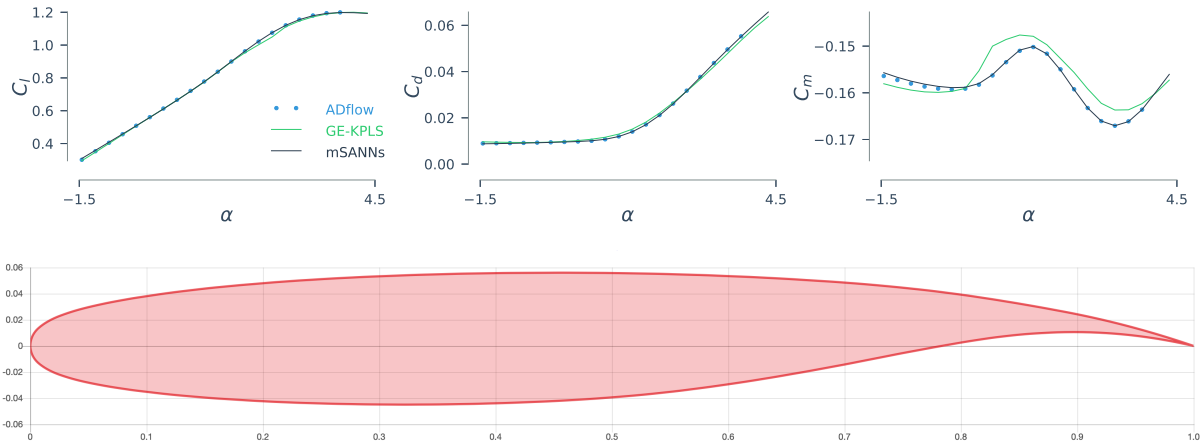
4.3. Efficiency of the mSANN-based optimization algorithm

The cost of an ADflow RANS simulation is between 2 and 5 minutes. Computing a large number of those simulations is expensive, so the training set used for training mSANN is between 58 and 145 days for the subsonic regime and between 5 and 14 days for the transonic one. We reduce this cost using parallel evaluations, resulting in a training cost for mSANN between 8 to 10 hours. The cost of a mSANN-based optimization is only a few seconds.

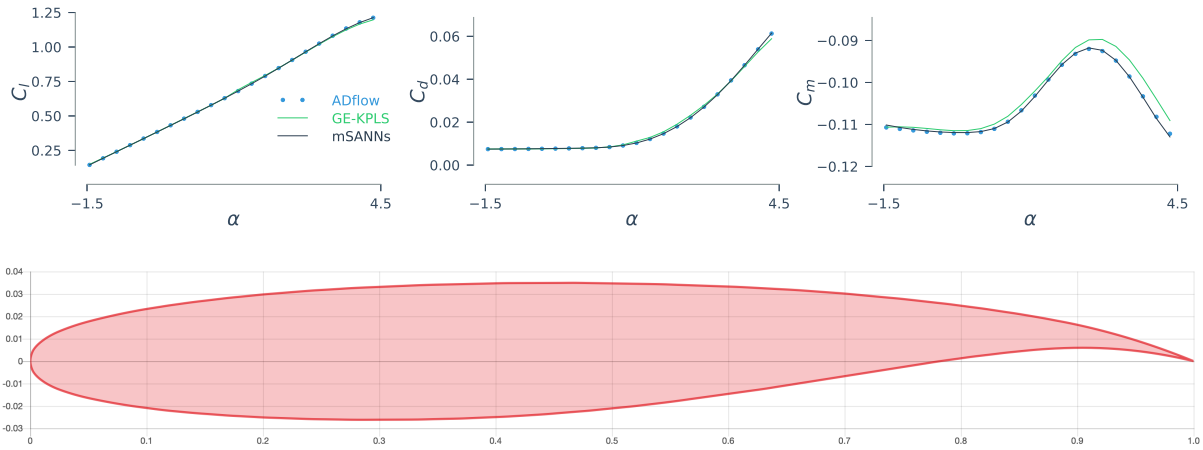
An optimization based on direct calls to ADflow costs between 20 and 60 minutes. This means that the total cost of a few mSANN-based optimizations is much more expensive that the ADflow-based ones when we take into account the training cost, this large cost can be amortized as the number of optimizations increases beyond 100 or so.

Finally, the mSANN-based optimization approach made it possible to create the optimization capability in Webfoil [6], which provides an interactive tool for airfoil design that would not be possible with direct calls to ADflow.

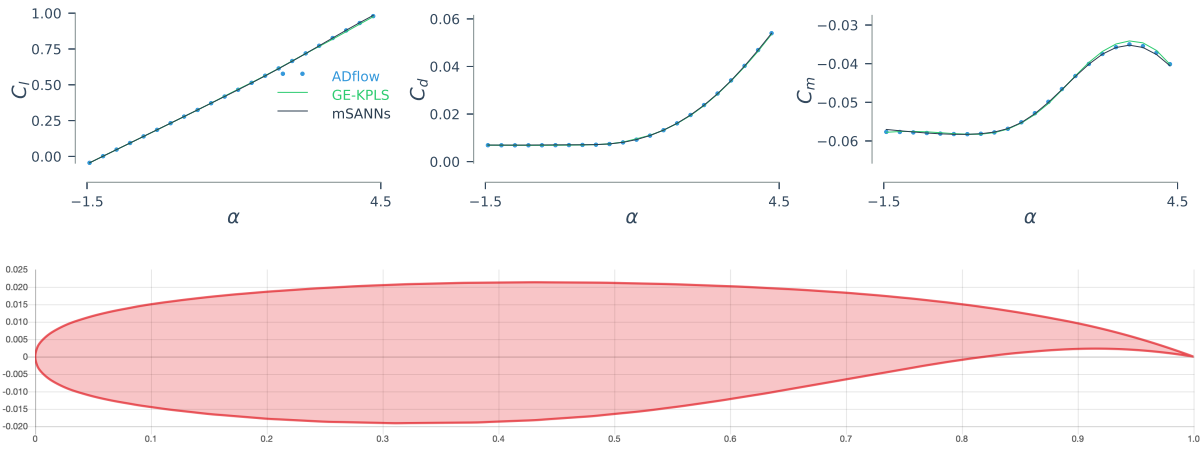
[6] http://webfoil.engin.umich.edu



(a) NASA SC(2)-0710 airfoil



(b) NASA SC(2)-0606 airfoil



(c) NASA SC(2)-0404 airfoil

Figure 7: Comparison between mSANN and GE-KPLS models in the transonic regime ($M = 0.73$) for three airfoils.

	C_d (count)		C_l		α
	ADflow	mSANN	ADflow	mSANN	
*NACA*0012	97.21	97.17	0.500	0.491	3.986
Opt_{ADflow}	92.17	92.36	0.499	0.499	1.863
Opt_{mSANN}	92.18	92.36	0.500	0.500	1.864

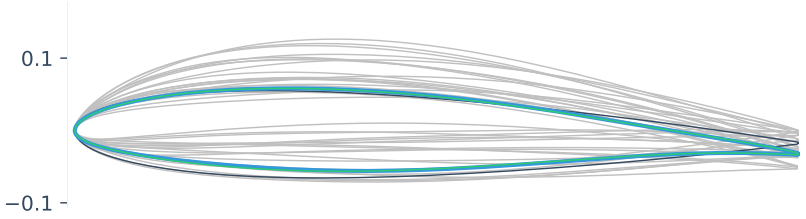


Figure 8: C_d minimization at $C_l = 0.5$ and $M = 0.45$. The optimized solutions given by ADflow and mSANN are in blue and green, respectively. The grey area shows the design space range. The optimal airfoil found using mSANN is indistinguishable to the one found using CFD.

	C_d (count)		C_l		α
	ADflow	mSANN	ADflow	mSANN	
$SC(2) - 0710$	114.44	1143.87	0.819	0.817	1.343
Opt_{ADflow}	101.31	101.36	0.820	0.818	0.848
Opt_{mSANN}	101.43	101.47	0.822	0.820	0.864

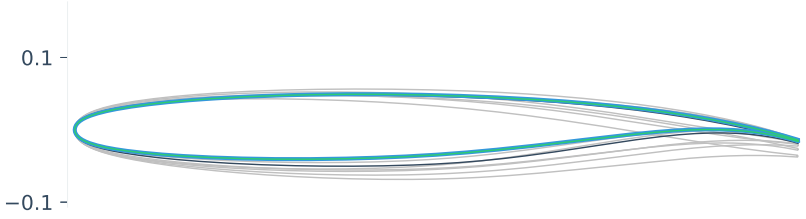


Figure 9: C_d minimization at $C_l = 0.82$ and $M = 0.72$. The optimized solutions given by ADflow and mSANN are in blue and green, respectively. The gray area shows the design space range. The optimal airfoil found using mSANN is indistinguishable to the one found using CFD.

5. Replication results

The codes needed for reproducing the results in this paper are available online under open-source licenses. After installing TensorFlow, the scripts to generate the results presented in this paper are available on https://data.mendeley.com/datasets/ngpd634smf/1. The Rosenbrock function test results given in Section 2.2 could be reproduced using files available in the repository "Rosenbrock". The scripts constructing the mSANN and predicting the aerodynamic coefficients given in Section 3.2 are available in the repository "Analysis". The optimization results given in Section 4 could be reproduced using the script files hosted in the repository "Optimization". In addition, ADflow is available on https://github.com/mdolab/adflow for reproducing ADflow results.

6. Conclusions

We developed an ANN methodology (mSANN) for fast and accurate airfoil design applications. The ANN model was enhanced with gradient information to increase the model accuracy for a given number of training samples. mSANN is based on gradually introducing the gradient information during the learning process via a dynamic parameter λ. We validated mSANN on an academic test case, the Rosenbrock function, and showed that it outperforms the standard ANN model and SANN model with an improvement of about 3% in terms of the L_2-norm of relative error. This approach is applicable in a large design space containing over one thousand airfoil shapes of various types at flow conditions ranging from subsonic to the transonic regime.

In previous work, we had developed a gradient-enhanced Kriging surrogate model (GE-KPLS) for the same purpose. However, that approach required separate models for the subsonic and transonic regimes. With mSANN, we were able to create a single-mode covering both regimes that is even more accurate. In subsonic, we obtained an error of 0.29% in terms of L_2-norm of relative errors compared to an error of 2.75% using GE-KPLS for C_d. The L_2-norm of relative errors for C_d in the transonic regime is 0.48% compared to the CFD results.

We performed two airfoil shape optimizations: one for the subsonic flow regime and another for the transonic regime. Compared to a CFD-based optimization, mSANN found the reference solutions with a difference of 0.01 counts and 0.12 counts for C_d in subsonic and transonic regimes, respectively.

These optimization problems are solved in less than four seconds instead of the hour required using the conventional approach that calls CFD directly, which opens the door to interactive airfoil design. This approach is now accessible through Webfoil, a web-based airfoil design tool and database. [7]

Acknowledgments

This work was partially supported by the Air Force Office of Scientific Research (AFOSR) MURI on "Managing multiple information sources of multi-physics systems", Program Officer Jean-–Luc Cambier, Award Number FA9550-15-1-0038.

[7] http://webfoil.engin.umich.edu

References

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

[2] A. Angelova, A. Krizhevsky, and V. Vanhoucke. Pedestrian detection with a large-field-of-view deep network. In *Proceedings of ICRA 2015*, 2015.

[3] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention, 2014. URL http://arxiv.org/abs/1412.7755. cite arxiv:1412.7755.

[4] D. Bettebghor, N. Bartoli, S. Grihon, J. Morlier, and M. Samuelides. Surrogate modeling approximation using a mixture of experts based on em joint estimation. *Structural and Multidisciplinary Optimization*, 43(2):243–259, 2011. 10.1007/s00158-010-0554-2.

[5] S. Bhattrai, J. H. de Baar, and A. J. Neely. Efficient uncertainty quantification for a hypersonic trailing-edge flap, using gradient-enhanced kriging. *Aerospace Science and Technology*, 80:261 – 268, 2018. ISSN 1270-9638. doi: 10.1016/j.ast.2018.06.036.

[6] N. P. Bons, X. He, C. A. Mader, and J. R. R. A. Martins. Multimodality in aerodynamic wing design optimization. *AIAA Journal*, 57(3):1004–1018, March 2019. doi: 10.2514/1.J057294.

[7] M. A. Bouhlel and J. R. R. A. Martins. Gradient-enhanced kriging for high-dimensional problems. *Engineering with Computers*, 1(35):157–173, January 2019. doi: 10.1007/s00366-018-0590-x.

[8] M. A. Bouhlel, N. Bartoli, J. Morlier, and A. Otsmane. An improved approach for estimating the hyperparameters of the kriging model for high-dimensional problems through the partial least squares method. *Mathematical Problems in Engineering*, 2016. doi: 10.1155/2016/6723410. Article ID 6723410.

[9] M. A. Bouhlel, N. Bartoli, A. Otsmane, and J. Morlier. Improving kriging surrogates of high-dimensional design models by partial least squares dimension reduction. *Structural and Multidisciplinary Optimization*, 53(5):935–952, 2016. doi: 10.1007/s00158-015-1395-9.

[10] M. A. Bouhlel, J. T. Hwang, N. Bartoli, R. Lafage, J. Morlier, and J. R. R. A. Martins. A Python surrogate modeling framework with derivatives. *Advances in Engineering Software*, 135:102662, September 2019. doi: 10.1016/j.advengsoft.2019.03.005.

[11] T. R. Brooks, G. K. W. Kenway, and J. R. R. A. Martins. Benchmark aerostructural models for the study of transonic aircraft wings. *AIAA Journal*, 56(7):2840–2855, July 2018. doi: 10.2514/1.J056603.

[12] T. R. Brooks, J. R. R. A. Martins, and G. J. Kennedy. High-fidelity aerostructural optimization of tow-steered composite wings. *Journal of Fluids and Structures*, 88: 122–147, July 2019. doi: 10.1016/j.jfluidstructs.2019.04.005.

[13] M. D. Buhmann and M. D. Buhmann. *Radial Basis Functions*. Cambridge University Press, New York, NY, USA, 2003. ISBN 0521633389.

[14] S. Chen, Z. Lyu, G. K. W. Kenway, and J. R. R. A. Martins. Aerodynamic shape optimization of the Common Research Model wing-body-tail configuration. *Journal of Aircraft*, 53(1):276–293, January 2016. doi: 10.2514/1.C033328.

[15] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu. Sobolev training for neural networks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4281–4290, 2017. URL http://papers.nips.cc/paper/7015-sobolev-training-for-neural-networks.

[16] J. H. S. de Baar, T. P. Scholcz, and R. P. Dwight. Exploiting adjoint derivatives in high-dimensional metamodels. *AIAA Journal*, 53(5):1391–1395, May 2015. doi: 10.2514/1.j053678.

[17] A. Dumont, J.-L. Hantrais-Gervois, P.-Y. Passaggia, J. Peter, I. Salah el Din, and É. Savin. *Ordinary Kriging Surrogates in Aerodynamics*, pages 229–245. Springer International Publishing, 2019. ISBN 978-3-319-77767-2. doi: 10.1007/978-3-319-77767-2_14.

[18] N. Garg, B. W. Pearce, P. A. Brandner, A. W. Phillips, J. R. R. A. Martins, and Y. L. Young. Experimental investigation of a hydrofoil designed via hydrostructural optimization. *Journal of Fluids and Structures*, 84:243–262, January 2019. doi: 10.1016/j.jfluidstructs.2018.10.010.

[19] K. Giannakoglou, D. Papadimitriou, and I. Kampolis. Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels. *Computer Methods in Applied Mechanics and Engineering*, 195(44):6312 – 6329, 2006. ISSN 0045-7825. doi: 10.1016/j.cma.2005.12.008.

[20] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005. doi: 10.1137/S0036144504446096.

[21] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL http://www.deeplearningbook.org.

[22] J. S. Gray and J. R. R. A. Martins. Coupled aeropropulsive design optimization of a boundary-layer ingestion propulsor. *The Aeronautical Journal*, 123(1259):121–137, January 2019. doi: 10.1017/aer.2018.120.

[23] K. Gurney. *An Introduction to Neural Networks*. Taylor & Francis, Inc., Bristol, PA, USA, 1997. ISBN 1857286731.

[24] L. Haitao, W. Xiaofang, and X. Shengli. Generalized radial basis function-based high-dimensional model representation handling existing random data. *Journal of Mechanical Design*, 139(MD-16-1098):13, 2016. ISSN 1050-0472. doi: 10.1115/1.4034835.

[25] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[26] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998. ISBN 0132733501.

[27] P. He, C. A. Mader, J. R. R. A. Martins, and K. J. Maki. An aerodynamic design optimization framework using a discrete adjoint approach with OpenFOAM. *Computers & Fluids*, 168:285–303, May 2018. doi: 10.1016/j.compfluid.2018.04.012.

[28] S. He, E. Jonsson, C. A. Mader, and J. R. R. A. Martins. Aerodynamic shape optimization with time spectral flutter adjoint. In *2019 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, San Diego, CA, January 2019. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2019-0697.

[29] X. He, J. Li, C. A. Mader, A. Yildirim, and J. R. R. A. Martins. Robust aerodynamic shape optimization—from a circle to an airfoil. *Aerospace Science and Technology*, 87:48–61, April 2019. doi: 10.1016/j.ast.2019.01.051.

[30] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean. Multilingual acoustic models using distributed deep neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, CA, 2013. URL https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/40807.pdf.

[31] I. C. Kampolis, E. I. Karangelos, and K. C. Giannakoglou. Gradient-assisted radial basis function networks: theory and applications. *Applied Mathematical Modelling*, 28(2):197 – 209, 2004. ISSN 0307-904X. doi: 10.1016/j.apm.2003.08.002.

[32] K. Kang, C. Qin, B. Lee, and I. Lee. Modified screening-based kriging method with cross validation and application to engineering design. *Applied Mathematical Modelling*, 70:626 – 642, 2019. ISSN 0307-904X. doi: 10.1016/j.apm.2019.01.030.

[33] G. K. W. Kenway, C. A. Mader, P. He, and J. R. R. A. Martins. Effective adjoint approaches for computational fluid dynamics. *Progress in Aerospace Sciences*, 110:100542, October 2019. doi: 10.1016/j.paerosci.2019.05.002.

[34] L. Laurent, R. Le Riche, B. Soulier, and P.-A. Boucard. An overview of gradient-enhanced metamodels with applications. *Archives of Computational Methods in Engineering*, 26(1):61–106, January 2019. ISSN 1886-1784. doi: 10.1007/s11831-017-9226-3.

[35] J. Li, M. A. Bouhlel, and J. R. R. A. Martins. Data-based approach for fast airfoil analysis and optimization. *AIAA Journal*, 57(2):581–596, February 2019. doi: 10.2514/1.J057129.

[36] R. P. Liem, G. K. W. Kenway, and J. R. R. A. Martins. Multimission aircraft fuel burn minimization via multipoint aerostructural optimization. *AIAA Journal*, 53 (1):104–122, January 2015. doi: 10.2514/1.J052940.

[37] R. P. Liem, C. A. Mader, and J. R. R. A. Martins. Surrogate models and mixtures of experts in aerodynamic performance prediction for aircraft mission analysis. *Aerospace Science and Technology*, 43:126–151, June 2015. doi: 10.1016/j.ast.2015.02.019.

[38] W. Liu and S. Batill. Gradient-enhanced neural network response surface approximations. *8th Symposium on Multidisciplinary Analysis and Optimization, Multidisciplinary Analysis Optimization Conferences, AIAA*, 2019. doi: doi:10.2514/6.2000-4923.

[39] Z. Lyu, G. K. W. Kenway, and J. R. R. A. Martins. Aerodynamic shape optimization investigations of the Common Research Model wing benchmark. *AIAA Journal*, 53 (4):968–985, April 2015. doi: 10.2514/1.J053318.

[40] M. Mangano and J. R. R. A. Martins. Multipoint aerodynamic shape optimization for subsonic and supersonic regimes. In *57th AIAA Aerospace Sciences Meeting, AIAA SciTech Forum, 2019*, San Diego, CA, January 2019. doi: 10.2514/6.2019-0696.

[41] J. R. R. A. Martins and J. T. Hwang. Review and unification of methods for computing derivatives of multidisciplinary computational models. *AIAA Journal*, 51(11): 2582–2599, November 2013. doi: 10.2514/1.J052184.

[42] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, September 2003. doi: 10.1145/838250.838251.

[43] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943. ISSN 1522-9602. doi: 10.1007/BF02478259.

[44] S. Pan and K. Duraisamy. Long-time predictive modeling of nonlinear dynamical systems using neural networks. *Complexity*, 2018:1–26, December 2018. doi: 10.1155/2018/4801012.

[45] N. Papila, W. Shyy, L. Griffin, and D. Dorney. Shape optimization of supersonic turbines using response surface and neural network methods. In *39th Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, January 2001. doi: 10.2514/6.2001-1065.

[46] M. J. D. Powell. *The Theory of Radial Basis Function Approximation in 1990*, pages 105–210. Oxford University Press, May 1992.

[47] M. M. Rai and N. K. Madavan. Aerodynamic design using neural networks. *AIAA Journal*, 38(1):173–182, January 2000. doi: 10.2514/2.938.

[48] J. Sacks, S. B. Schiller, and W. J. Welch. Designs for computer experiments. *Technometrics*, 31(1):41–47, 1989.

[49] N. R. Secco and J. R. R. A. Martins. RANS-based aerodynamic shape optimization of a strut-braced wing with overset meshes. *Journal of Aircraft*, 56(1):217–227, January 2019. doi: 10.2514/1.C034934.

[50] N. R. Secco, J. P. Jasa, G. K. W. Kenway, and J. R. R. A. Martins. Component-based geometry manipulation for aerodynamic shape optimization with overset meshes. *AIAA Journal*, 56(9):3667–3679, September 2018. doi: 10.2514/1.J056550.

[51] W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris. Screening, predicting, and computer experiments. *Technometrics*, 34(1):15–25, 1992. doi: 10.1080/00401706.1992.10485229.

[52] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12(6):620–631, 1998. doi: 10.1063/1.168744.

[53] Z. Xu, J. H. Saleh, and V. Yang. Optimization of supercritical airfoil design with buffet effect. *AIAA Journal*, pages 1–11, January 2019. doi: 10.2514/1.j057573.

[54] A. Yildirim, G. K. W. Kenway, C. A. Mader, and J. R. R. A. Martins. A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations. *Journal of Computational Physics*, 397:108741, Nov. 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2019.06.018.

[55] Y. Yu, Z. Lyu, Z. Xu, and J. R. R. A. Martins. On the influence of optimization algorithm and starting design on wing aerodynamic shape optimization. *Aerospace Science and Technology*, 75:183–199, April 2018. doi: 10.1016/j.ast.2018.01.016.

[56] Y. Zhang, W. J. Sung, and D. N. Mavris. Application of convolutional neural network to predict airfoil lift coefficient. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, 2018. doi: 10.2514/6.2018-1903.

[57] W. Zongmin. Hermite-birkhoff interpolation of scattered data by radial basis functions. *Approximation Theory and its Applications*, 8(2):1–10, Jun 1992. ISSN 1573-8175. doi: 10.1007/BF02836101.