Reply to review

Review from group 29

Summary:

Given an airfoil shape, this team wants to train a neural network to predict a variety of metrics that describe performance. They are trying to predict lift, drag and moment coefficients from data describing the shape of the airfoil, Reynolds number, Mach number, and angle of attack. They use a fully connected neural network for this task.

Strengths:

1) Problem seems interesting - there are many parameters that affect performance and computational fluid dynamics are expensive simulations to perform.
2) Good introduction and problem is well introduced
3) Good introduction and problem is well introduced
4) Did a good job verbally explaining their problem
5) Their explanation of the data is fantastic
6) They did a good job posing themselves a problem and defining inputs, outputs, etc.

Thanks so much for your comments

Questions:

1) What do your results look like with the different Reynolds numbers? You only showed a plot of results given a specific airfoil and 1 value of Reynolds number. How do your results vary across airfoils?
   The results are also well-fitted for different airfoil and reynold number, we could put more results there, but that would be too many plots
2) How would your results change if 1 network is trained to predict all three of your desired parameters?
   This is another way of doingt it, from what we tested, training three outputs seperately gives us more accurate results.

Review from group 69

Definitely an interesting application of NNs for something that's hard to calculate through other means. Clever source of data.

Thanks so much for your comments

I would have likely more discussion around what CFD was used and about the data processing in general. For example, why was each airfoil scaled in size individually rather than with respect to some mean size?

Airfoils vary a lot from one to one, and its very hard find a representive mean size.

On the model itself, I noticed that the output layer had a sigmoid activation function. I feel an unbounded linear output activation would work better, since there are meant to be unbounded outputs. This might explain some of the issues near the top and bottom of your output ranges, since sigmoid will compress these and losses will be small, especially since your loss isn't MSE.

We are using sigmoid because we add non-linearality through the layers. Also, as one airfoil is assumed to be within the same scale (x-axis) the total length is bounded. Therefore, we used sigmoid. We have tried others, but sigmoid seems to provide us with the most desirable results

On presentation, I like the dense loss vs epoch plot, but I'd like to see it in semilog-Y format. The very large noise in the loss might mean too high a learning rate. Try something lower to see how it trains. The plot also shows the 5-layer validation being better than the 11-layer validation, which is unexpected. Maybe go into some more detail on that.

Thanks for your comment. This is also one of the problems we are facing in constructing the model. Mostly, we believe this might due to an insufficient data of airfoil types.

There are also some small inconsistencies: Your presentation says you used the Adam optimizer but the code shows RMSprop. Your loss also seems to be MSE rather than the RMSE you mention in the presentation

We appreciate your comment. In the actual work, we indeed use RMSprop and RMSE. We are sorry for the type during the presentation

Good work!

Review from group 88

Overview: This group first clearly states the background. They evidently shows why they use Neural Network Surrogate Model by comparing it with Dimensionality Reduction Model and Typical Surrogate Model. They use standard Neural Network procedures, which is clear and understandable.

Thanks for your comment.

Improvement:

1) Is the data preprocessing enough? Or is the original dataset completely perfect at all?
   We believe so. We followed the instruction provided by the original paper.
2) Do the features have the same weight and correlation?
   Even though, practically, they weighs different, in our model we treated those features as purely numerical parameters.
3) Did you compare the result with other two models? Otherwise you cannot guarantee that Neural Network has more advantages than the other two models.
   No, we did not compare MNN with the other two models by ourselves. Our conclusion is based on the literature reviews.

Fast Airfoil Aerodynamic Analysis Using Neural Network Surrogate Model

1st Bingran Wang
dept. Aerospace Engineering
University of California, San Diego
La Jolla, CA
b6wang@ucsd.edu

2nd Jiewen Yang
dept. Electrical Engineering
University of California, San Diego
La Jolla, CA
jiy219@ucsd.edu

***Abstract*—Airfoil aerodynamic analysis is a computationally expensive process, but it is vitally important in aircraft designs. In this work, we introduce the method of neural network surrogate modeling. Using the dataset generated by simulations, we trained the neural network surrogate models which have been testified to be capable of accurately predict different airfoils' aerodynamic performance under various conditions.**

***Index Terms*—Airfoil, Neural Network, Surrogate Modeling**

I. Introduction

Airfoil aerodynamic analysis is of great importance on aircraft designs. Airfoils used to be designed exclusively based on wind-tunnel experiments, which are expensive and time consuming. Nowadays, airfoil analysis mostly replies on high-fidelity computational fluid dynamics(CFD) simulations, but they are computationally expensive to run. To address the need of fast airfoil analysis, researchers have focused on two main branches: dimentionality reduction [1], and surrogate modeling [2]. The dimensionality reduction methods, such as principal component analysis and partial least squares, reduce the number of design variables by obtaining representative principal components. However, they lose part of the available information as a trade-off. Surrogate models, on the other hand, provide an affordable alternative to the evaluation of the expensive deterministic functions, popular surrogate models include Gaussian regression process, radial basis function and neural network. Gaussian regression process, radial basis function can greatly reduce the the computational cost but they can hardly deal with large data set. Neural Network [3] surrogate models, on the other hand capture intricate structure of training data and handle large data set via batch optimization. They have been proved to accurately and efficiently represent many physics-based models [4].

In this paper, we present a method of using neural network surrogate model to provide an efficient and accurate way to predict an airfoil's performance analysis.

The rest of the paper is organized as follows. Section II talks about the background of airfoil performance analysis, as well as the reviews of the recently developed methods. Section III-A talks about how the data used to train the neural network is generated, as well as the feature extractions methods we use to process the data. Section IV-A talks about the methods we use to construct and train the neural network. Section V-C talks the results generated from the neural network model. In Section VI, we give conclusions on the effectiveness of this method, and provide insights on the further directions for this method.

II. Related Work

In this Section, we provide some background on airfoil analysis, and then review the recently developed methods to provide a fast analysis of the performance of an airfoil.

A. Airfoil Analysis

Airfoil is an important tool that represents a cross-sectional shape of an aircraft wing. The aerodynamic performance of an airfoil can be characterized by three coefficients, C_l, C_d and C_m. They are the dimensionless coefficients that relate the, respectively, lift, drag and pitching moment generated by the airfoil to the fluid density around the body. They are defined as:

$$C_l = \frac{L}{qS}, \quad C_d = \frac{D}{qS}, \quad C_m \frac{M}{qSc},$$

where L is the lift, M is the pitching moment, D is the drag, q is the dynamic force, S is the wing area, and C is the chord of the airfoil.

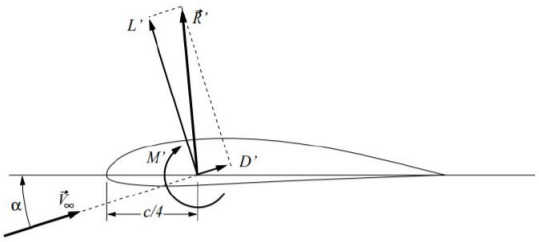

Fig. 1. Airfoil Aerodynamic Characterization.

In aerodynamics, the aerodynamic performance coefficients depend on the Reynold number (Re), angle of attack (α), Mach number (M), and the airfoil shape. Therefore we can characterize an equation for C_l, C_d and C_m as:

$$C_l, C_d, C_m = f(\alpha, Re, M, \textit{Airfoil Shape}). \tag{1}$$

Based on (1), we can construct a surrogate model that quickly analyze C_l, C_d and C_m.

B. Literature Review

Surrogate models provide an affordable alternative to the evaluation of expensive deterministic functions. One typical way to construct a surrogate model is to base on Gaussian regression processes [5]. It is a Bayesian approach which assumes a Gaussian processes prior over functions. This method is easy to implement and very effective, however, the main limitation is that memory requirements and computational demands grow as the square and cube respectively. Another typical way is to use the radial-basis-function [6], this method approximates the functions using radial-basis-function. It is also an effective way to construct the surrogate model but it also suffers the curse of dimensionality.

Neural networks [3], vaguely inspired by the biological neural networks, allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstracts. It motivates breakthrough in large-scale regression problem. Raissi *et al.* [7] proposes the physics-informed neural networks (PINN), which uses neural networks gradient and incorporate useful physics information from governing equations, it has proved that the PINN model can accurately predict the flow field. Du *et al.* [8] proposes a B-Spline-based generative adversarial network (GAN) model for fast interactive airfoil aerodynamic optimization, it uses the GAN model to random generate airfoil shapes, and constructs a multi-layer neural nework (MNN) surrogate model to do an airfoil aerodynamic optimization. The models are proved to be very accurate when predicting aerodynamic coefficients of an airfoil. This project is inspired by [8], we apply a similar method, and try to achieve a promising result.

III. Dataset and Features

In this section, we describe the method to generate the dataset we need, and the feature extractions we can perform to preprocess the dataset.

A. Dataset Generation

The dataset is generated by two steps. At first, 1600 airfoil shape files are downloaded from UIUC Airfoil Coordinate Database [9]. The shape of an airfoil is represented by 251 points, and in each airfoil shape file, we have 251 (x,y) coordinates to represent the 251 points. The snapshot of one airfoil shape file is shown in Figure 2. Once we have the airfoil shape files, the next step is to run CFD solver for each airfoil under different conditions to generate the outputs, i.e. C_l, C_d and C_m. We use ADflow [10], a Reynolds-Averaged Navier-Stokes CFD solver, to generate the output results. For conditions,we vary the angle of attack from -4.25 degrees to 15.75 degrees, Mach number from 0.16 to 0.60, and for Reynold number, we use 3,000,000, 6,000,000, 9,000,000. For each airfoil shape, we generate one file contain different conditions and the corresponding output results, a snapshot of the output file can be found in 3. Putting all the data available, we have the inputs of size $5,000,000 \times 505$, and outouts of size $5,000,000 \times 3$. For inputs, we have 505 columns which are 251 x-coordinates, 251 y-coordinates, angle of attack, mach number, and Reynold number. For outputs, the three columns are C_l, C_d and C_m.

After the data collection, we split the data into training set and testing set, whose proportions were 80% and 20% respectively. Moreover, the 5-fold cross validation was also utilized with 20% of the training data used for validation set. Since we have a large number of features (505 columns) for the dataset, it is necessary for us to reduce the dimensions by some feature extraction methods.

	A	B
1	x	y
2	1	0
3	0.999842	3.47E-05
4	0.999368	0.000138
5	0.998579	0.000311
6	0.997476	0.000551
7	0.996057	0.00086
8	0.994326	0.001238
9	0.992282	0.001682
10	0.989928	0.002193
11	0.987263	0.002771
12	0.984292	0.003415
13	0.981014	0.004124
14	0.977432	0.004897

Fig. 2. Snapshot of one airfoil shape file

cL	cD	cM	Mach	Reynold Number	Alpha
-0.320500	0.142790	-0.043100	0.3	3000000	-13.750000
-0.357500	0.139430	-0.054400	0.3	3000000	-13.500000
-0.348400	0.136740	-0.055800	0.3	3000000	-13.250000
-0.339600	0.133990	-0.057200	0.3	3000000	-13.000000

Fig. 3. Snapshot of the CFD result file for one airfoil shape

B. Feature Extraction

Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. The data input data set has a size of $5,000,000 \times 505$, a total of 505 parameters to be modeled. Two feature extraction methods are implemented for a high efficiency and accurate model, dense and dropout regularizations. Dense layer provides a fully connected neural network system, which condense the dimension of perceptrons by a given weights. It is one of most common and useful hidden layer to be utilized nowadays. Dropout regularization, on the other hand, randomly drops parts of its information to prevent overfitting, especially useful when we dealing with data type with 10^{-4} difference.

IV. Methods

After we generate the dataset, we construct Multi-layer neural network (MNN) surrogate models for C_l, C_d and C_m separately. All the models have similar neural network structures. This section describes the neural network structure of the MNN surrogate model, as well as the verification methods we use to check the accuracy of the model.

A. Multi-Layer Neural Network

Multi-layer neural network (MNN) provides the solution for the classification of input parameters, from which, Coefficient of drag (C_d), lift (C_f), and pitching moment (C_f) can be

extracted. The way neural network gain its capability of distinguish varies parameter inputs is by add sophisticated hidden layers along with proper activation function within its structure. Each hidden layers contribute to the feature extraction. More specifically, on each hidden layer, the MNN will transform data information from previous layers under certain feature extraction methods and then integrate to the next one by applying proper nonlinear function, for example, Rectified Linear Units (ReLU). The process ends when it reaches the output layer. MNN will extract parameter features and each hidden layer and automatically allocate the information within certain output. Fig.4 provides a simple multi-layer neural network with two layers. ReLU is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. So it can be written as

$$f(x) = max(x, 0). \tag{2}$$

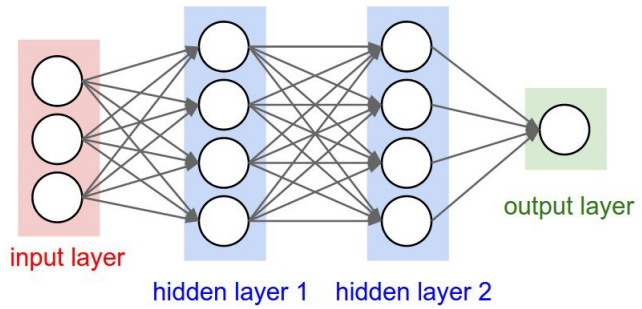

Fig. 4. Neural Network with two hidden layers.

The construction process of MNN model is shown in Figure 5 and described as:

1) Preprocess the input parameters with $MinMaxScaler$.
2) Build up mutiple-hidden-layer neural networks with dense and dropout feature extraction, followed by ReLU activation function (2).
3) Set the cost function to RMSE between training data and MNN predictions.
4) Train MNN model with rmsprop optimizer via batch optimization.
5) Calculate RMSE and relative error of training and validation sets to avoid overfitting

Figure 6 provides a snapshot of MNN model within tf.keras tool.

B. Optimizer

The optimizer is selected to be rmsprop, which deals with problem that gradients may vary widely in magnitude. Rmsprop stands for root mean square prop algorithm, which

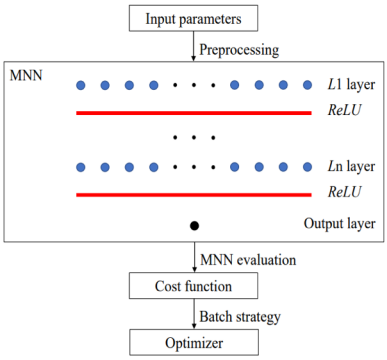

Fig. 5. Construction process of MNN model [8].

```
def root_mean_squared_error(y_true, y_pred):
        return K.square(K.mean(K.square(y_pred - y_true)))
input_dim = 505
def Airfit():
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Dense(100,activation='relu',input_shape=(input_dim,)))
    model.add(tf.keras.layers.Dropout(0.3))
    model.add(tf.keras.layers.Dense(100,activation='relu'))
    model.add(tf.keras.layers.Dropout(0.3))
    model.add(tf.keras.layers.Dense(100,activation='relu'))
    model.add(tf.keras.layers.Dropout(0.3))
    model.add(tf.keras.layers.Dense(100,activation='relu'))
    model.add(tf.keras.layers.Dropout(0.3))
    model.add(tf.keras.layers.Dense(100,activation='relu'))
    model.add(tf.keras.layers.Dropout(0.3))
    model.add(tf.keras.layers.Dense(100,activation='relu'))
    model.add(tf.keras.layers.Dropout(0.3))

    model.add(tf.keras.layers.Dense(1,activation='sigmoid'))
    return model

#Compile and train the model
Air_fit = Airfit()
Air_fit.compile(optimizer = "rmsprop", loss = root_mean_squared_error,
                metrics =["accuracy"])

history_Air_fit = Air_fit.fit(X_train, Y_train, validation_data=(X_val, Y_val), epochs=
```

Fig. 6. Snapshot of MNN model within tf.keras tool

keeps the moving average of the squared gradients for each weight. Mathematical representation can be written as:

$$E[g^2]_t = [g^2]_{t-1} + (1-\beta)(\frac{\partial C}{\partial w})^2, \tag{3}$$

$$w_t = w_{t-q} - \frac{\mu}{\sqrt{E[g^2]_t}}\frac{\partial C}{\partial w}, \tag{4}$$

where E[g] is the moving average of quared gradients, $\frac{\partial C}{\partial w}$ is the gradient of the cost function with respect to the weight. μ is the learning rate, β is the moving average parameter. In python tf.keras toolbox, there is existing function to call RMSprop optimizer. RMSprop is a good, fast stable optimizer.

C. Verification

We select the root mean squared error (RMSE) and relative errors to verify the accuracy of the trained MNN surrogate model. The RMSE represents the square root of the second sample moment of the differences between predicted values and observed values of the quadratic mean of these differences. The RMSE serves to measure the accuracy of the trained neural network model, it is scale dependent, as it compare forecasting errors of different models for a particular dataset. The function of RMSE is defined as,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N_t}(Y_{pred} - Y_{real})^2}{N_{testing}}}, \tag{5}$$

where $N_{testing}$ is the number of testing points, Y_{pred} is from MNN model prediction, Y_{real} is real model observations. The relative error is the ratio of the sum-of-squares error between the test output and predicted output to the sum-of-squares of the test output, it can be expressed as

$$Rel.error = \sum_{i=1}^{N_t}(Y_{pred} - Y_{real})^2 / \sum_{i=1}^{N_t}(Y_{real})^2. \quad (6)$$

Since the outputs of our model, C_l, C_d, and Cm, all have small values between 0 and 1. Large errors are undesirable in our MNN model . RMSE, particularly, weighs more on relatively high errors, which can give a better insights of MNN model performance when comparing with other cost functions such as Mean Absolute Error (MAE) or Mean Squared Error (MSE).

V. Results

A. Convergence Study

We first investigate the depth of neural network structure needed to train a good surrogate model. To train the MNN surrogate model, we use RMSprop optimizaer, RMSE loss function, and we set epochs equal to 70, batch size equal to 100. We compared two MNN structures, the first has 5-hidden-layer neural networks,with dense and dropout feature extraction, followed by ReLU activation function, the second one has 10 layers of neural networks, also with feature extraction and followed by ReLU. The loss vs epoch plots of both structures are shown in Figure 8, from which, we observe that for both MNN strcutures, our models converges with 70 epochs, and there is no over fitting observed. Therefore, we decide to use 5-hidden-layer neural networks for a faster computation time.

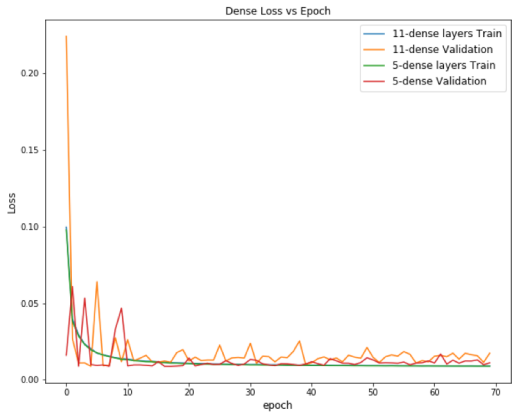

Fig. 7. Calculated model RMSE comparison between validation and training data

B. Accuracy Results

Once we decide on the specific MNN model we use to train the surrogate models, we want to compare the RMSE and relative errors to check the accuracy of our trained models. A proper estimation of of RMSE is within one standard deviation of testing points, which is expected to be about 10%. For relative verification metrics, relative error within 1% is anticipated [8]. The key verification metrices are shown in TableI. The RMSE values are all below 1% meaning they are all good global surrogate models. The relative errors are well-controlled as they are less and equal to 1% .

TABLE I
TABLE TITLE

Model	RMSE	Relative Error
C_l	0.935%	0.147%
C_d	0.000331%	0.287%
C_m	0.00231%	1.00%

C. Visualize the Results

To better visualize the results of our trained model, we generate the C_l vs angle of attack plots for one specific airfoil at Reynold number 3,000,000, and at differnt mach numbers. The plots are shown in Figure V-C. It shows that the predicted C_l fits very well with the real C_l data at vairous conditions.

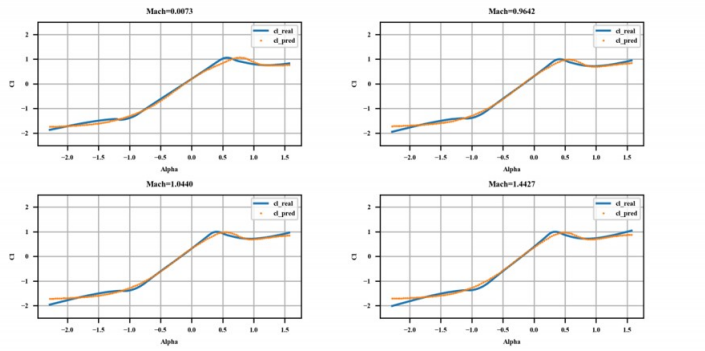

Fig. 8. Airfoil AG34, C_l vs alpha plots

VI. Conclusion

In this project, we construct three MNN surrogate models to predict Cl, C_d and C_m of different airfoils at various conditions. We parametrize the airfoil shape with 251 points, and run CFD simulations of each airfoil under different condtions to generate the dateset. We then train the MNN surrogate model with five hidden-layer neural networks. The surrogate models have been verified to have very small RMSE and relative errors, which proves the accuracy of the surrogate models. With the trained surrogate models, analysis of the aerodynamic performance of an airfoil can be done accurately and efficiently. It enables the potential of optimizing an airfoil shape, which may take a large number of iterations, for specific aerodynamic performance requirement.

VII. Contribution

1) Bingran: Preprocessing data, writing the document.
2) Jiewen: Training neural network model, writing the document.

References

[1] X. Chen, M. Diez, M. Kandasamy, Z. Zhang, E. F. Campana, and F. Stern, "High-fidelity global optimization of shape design by dimensionality reduction, metamodels and deterministic particle swarm," *Engineering Optimization*, vol. 47, no. 4, pp. 473–494, 2015.

[2] B. Peherstorfer, K. Willcox, and M. Gunzburger, "Survey of multifidelity methods in uncertainty propagation, inference, and optimization," *Siam Review*, vol. 60, no. 3, pp. 550–591, 2018.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data," *Journal of Computational Physics*, vol. 394, pp. 56–81, 2019.

[5] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1939–1959, 2005.

[6] J. Park and I. W. Sandberg, "Approximation and radial-basis-function networks," *Neural computation*, vol. 5, no. 2, pp. 305–316, 1993.

[7] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[8] X. Du, P. He, and J. Martins, "A b-spline-based generative adversarial network model for fast interactive airfoil aerodynamic optimization," in *AIAA Scitech 2020 Forum*, 2020, p. 2128.

[9] M. S. Selig, "Uiuc airfoil coordinates database." [Online]. Available: https://m-selig.ae.illinois.edu/ads/coord$_database.html$

[10] M. H. A. Madsen, F. Zahle, N. N. Sørensen, and J. R. Martins, "Multipoint high-fidelity cfd-based aerodynamic shape optimization of a 10 mw wind turbine," *Wind Energy Science*, vol. 4, no. 2, pp. 163–192, 2019.