14th World Congress in Computational Mechanics (WCCM)
ECCOMAS Congress 2020
Virtual Congress: 11 – 15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

AN ADAPTIVE SAMPLING PROCEDURE FOR TRAINING A NEURAL NETWORK BASED ON A GAUSSIAN PROCESS

NAHLA ALHAZMI[1], YOUSEF GHAZI[2], RADEK TEZAUR[3] AND CHARBEL FARHAT[4]

[1] King Abdulaziz City for Science and Technology
Riyadh, 11442, Saudi Arabia
nalhazmi@kacst.edu.sa

[2] King Abdulaziz City for Science and Technology
Riyadh, 11442, Saudi Arabia
yghazi@kacst.edu.sa

[3] Department of Aeronautics and Astronautics, Stanford University
Stanford, CA 94305-4035, USA
rtezaur@stanford.edu

[4] Department of Aeronautics and Astronautics, Department of Mechanical Engineering
and Institute for Computational and Mathematical Engineering, Stanford University
Stanford, CA 94305-4035, USA
cfarhat@stanford.edu

Key words: Acquisition Function, Adaptive Sampling, Gaussian Process, Machine Learning, Neural Network, Regression.

Abstract. A novel approach is presented for efficiently training a neural network (NN)-based surrogate model when the training data set is to be generated using a computationally intensive high-fidelity computational model. The approach consists in using a Gaussian Process (GP), and more specifically, its acquisition function, to adaptively sample the parameter space of interest and generate the minimum amount of training data needed to achieve the desired level of approximation accuracy. The overall approach is explained and illustrated with numerical experiments associated with the prediction of the lift-over-drag ratio for a NACA airfoil in a large, two-dimensional parameter space of free-stream Mach number and free-stream angle of attack. The obtained numerical results demonstrate the superior accuracy delivered by the proposed training over standard trainings using uniform and random samplings.

1 INTRODUCTION

Classical machine learning algorithms use computational methods to “learn” information directly from data, without relying on a predetermined equation as a model. Originally, their development was mostly driven by problems for which no predetermined equation is available.

As such, machine learning algorithms can also be described as algorithms for building data-driven models. For example, a neural network (NN) regression is a supervised machine learning algorithm that can be used to build a data-driven model. Given a sufficiently large amount of data, it typically partitions the data into two subsets, uses one for training, and the other for testing.

Even for problems where predetermined equations are available to build parametric computational models – for example, high-dimensional structural dynamics (CSD) and computational fluid dynamics (CFD) models (HDMs) – NNs are currently in vogue for building alternative, low-dimensional, surrogate models, at least for a small number of scalar quantities of interest (QoIs), that are easier to handle and faster to process. In this case however, the data is not available, but needs to be generated using: the parametric HDMs; and an approach for sampling the parameter space of interest – for example, a design space or, for applications in aeronautics and astronautics, a space of flight conditions.

Parameter sampling algorithms come in two flavors: a priori sampling; and adaptive sampling. A priori sampling algorithms sample points in the parameter space either randomly, or according to a non-adaptive, predesigned scheme. Examples include the full factorial sampling [1], random sampling, and Latin hypercube sampling [2] algorithms. Such algorithms are not optimal because they have no explicit awareness of where the regression will be inaccurate. For this reason, they may require a larger than necessary number of samples in order to deliver the expected accuracy at regression time. Consequently, they may lead to unaffordable training costs for NNs, particularly when the parameter space is high-dimensional. On the other hand, adaptive sampling schemes sample points in regions of the parameter space where the regression can be deemed to be inaccurate, and therefore avoid over-sampling. Hence, they can mitigate the curse of dimensionality, but require the availability of an error indicator. Unfortunately, when an NN-based surrogate model is to be built for a quantity of interest, finding a suitable error indicator can be challenging. This is in contrast with projection-based model reduction approaches for building low-dimensional surrogate computational models [3], where residual-based error indicators are common for guiding adaptive sampling procedures known as “greedy procedures” [4].

This paper presents a new idea for addressing the above dilemma. It consists in training an NN-based surrogate model (or regression) using a parametric HDM of interest and a Gaussian Process (GP) [5]. Indeed, a GP models distributions over functions and generates a stochastic model that can be utilized to construct an acquisition function for guiding an adaptive sampling of the parameter space of interest. However, given a parametric computational model, a GP and its acquisition function are typically used to construct a computationally leaner surrogate model in the form or a regression model. Hence, one may ask why using a GP to construct an NN-based surrogate model of a given parametric computational model, when the GP can also be used to build a low-dimensional surrogate of that same HDM? The answer lies in the superior approximation properties of NNs and in particular deep NNs. Using as a backdrop the problem of constructing for an airfoil surrogate models of aerodynamic QoIs such as the lift-over-drag ratio in the two-dimensional parameter space (free-stream Mach number M_∞, free-stream angle of attack α_∞), it is shown in this paper that: a GP can be used to construct and train NN-based surrogate models; that such models outperform accuracy-wise counterparts directly constructed using the GP itself; and that they also outperform NN-based surrogate models trained using uniform grid sampling and random sampling. The surrogate models constructed and trained using the proposed approach are particularly interesting for time-critical applications such as design optimization and uncertainty quantification.

2 BASICS OF GAUSSIAN PROCESS REGRESSION

Formally, a GP is a stochastic process – that is, a collection of random variables $Y(x)$, indexed by an n-dimensional vector x that often represents a time or parameter space such that every finite collection of the random variables has a multivariate normal distribution. GPs have recently found their place in machine learning (ML) as a popular tool for classification. They have also been used for regression for quite some time under the term "krieging". As a collection of normal random variables, a GP can be characterized by its mean value function, $m(x) = E[Y(x)]$, where E is the mathematical expectancy, and its covariance function

$$k(x,x') = cov(x,x') = E[(Y(x) - m(x))(Y(x) - m(x))] \tag{1}$$

that is often also called the kernel.

A choice of the kernel can be tuned to a particular data set to represent varying degrees of smoothness or other prior properties known or assumed about the data. Matérn, exponential rational quadratic, and piecewise polynomial are examples of such kernels [5]. However, the most well-known is the squared exponential kernel (also known as the radial basis function (RBF) kernel)

$$\mathrm{k}(x,x') = \sigma^2\, exp\, \frac{-|x-x'|^2}{2\lambda^2} \tag{2}$$

where the hyperparameter σ controls the variance and λ is known as the length scale.

For regression purposes, the training data – either directly available or generated using a parametric computational model that is possibly an HDM – is used as a prior and Bayesian posterior predictions can be shown again to form a Gaussian process whose mean and covariance function can be easily computed [5]. The resulting Gaussian process model is often called nonparametric because it depends only on the training data, the choice of the kernel, and its hyperparameters. To estimate the values of the hyperparameters, cross-validation or maximizing the marginal (log) likelihood [5] are the most popular techniques.

3 BASICS OF NEURAL NETWORKS

Artificial NNs are another popular ML tool. Modelled after biological neurons and their synaptic connections, the nodes of such a network attach to each other by passing information through links among them [6]. Various architectures can be formed, of which a feed-forward network is one of the most common. Its nodes are divided into layers (see Figure 1): an input layer; single or multiple hidden layers; and an output layer. A node in a layer takes input from nodes in the previous layer, processes it, and passes it to the nodes in the next layer. The propagation function for one such node can be written as

$$\boldsymbol{\Phi}\left(\sum_k \boldsymbol{w_k}\ \boldsymbol{a_k} + \boldsymbol{b}\right) \tag{3}$$

where $\boldsymbol{a_k}$ are the inputs a node receives from counterparts of the previous layer, $\boldsymbol{w_k}$ and $\boldsymbol{b}$ are the weights and the bias of the affine function that is applied to the inputs, respectively, and $\boldsymbol{\Phi}$ is typically a nonlinear function known as the activation function.

An NN is usually trained iteratively by optimizing its hyperparameters – which are the weights and biases of each of its nodes – so that a cost (loss) function of interest is minimized. In a supervised learning framework, the training data comes in the form of input-output pairs and the loss function measures the discrepancy between the output of the NN given an input in the training set, and the corresponding output in the training set. Classical, gradient-based, optimization algorithms have been adapted and tuned for performance to NN applications using a technique called back-propagation that makes it possible to efficiently compute the gradient of the cost function with respect to the parameters of the network.

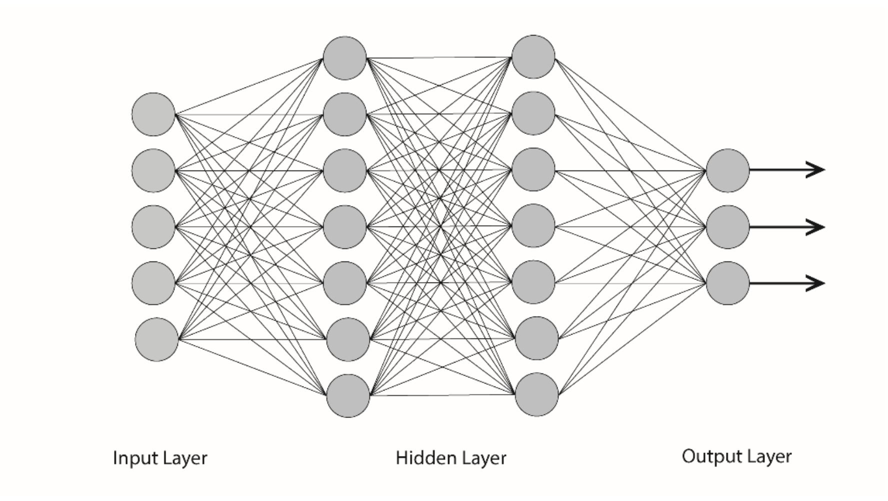

Figure 1: Typical architecture of an artificial NN.

When using an NN as a data fitting tool, mathematical results such as the universal approximation theorem [7] provide confidence that any input-output relationship can be approximated. For example, a single hidden layer network with a large enough number of nodes can approximate any continuous function. However, such results are neither unique to NNs, nor necessarily constructive: they do not give guidance on how many nodes, or layers need to be used to achieve the desired results.

4 ACTIVE LEARNING USING A GAUSSIAN PROCESS REGRESSION

Unlike in a typical application of ML where vast amounts of paid-for data are often available, data in engineering sciences can be scarce, expensive to measure experimentally, or time-consuming to compute using HDMs such as those arising in CSD and CFD applications. In this context, it is desirable to be able to train an ML algorithm with as little data as possible and by inference, use the most useful (in some sense) data points.

Given a parametric computational model, Gaussian process regression (GPR) already provides, as stated in Section 1, a surrogate model that can be exploited to perform a function exploration – that is, to learn a function as well and as quickly as possible. To find the next point to sample in a given parameter space, it relies on an acquisition function that measures the utility of the input data points. Given the current set of sampled data points and their function values, the variance of the posterior GP can be used as an acquisition function. Selecting the argument of the maximum of the acquisition function then amounts to greedily picking a parameter point where the function value is most uncertain with respect to the current model. Such a greedy algorithm only focuses on minimizing the current uncertainty rather than looking farther into the future. However, it can be shown that the next point to sample that it picks is close to optimal in some sense [5].

In this paper, it is proposed to train an artificial NN using the GP-driven greedy algorithm summarized below and referred to as Algorithm 1.

Algorithm 1: *Greedy algorithm for function exploration using a Gaussian process*

```
1. Set $i = 1$ and initialize the sampling set: $\boldsymbol{S} = \{x^1\}$.
2. Evaluate the function on the sampling set, i.e. $y(\boldsymbol{S}) = \{y(x^1), \dots, y(x^i)\}$ .
3. Optimize the hyperparameters of the kernel of the Gaussian process given the sampling
   set $\boldsymbol{S}$ and the function values on the sampling set, $y(\boldsymbol{S})$).
4. Find the critical point $x_{cr}$ where the predicted standard deviation of the Gaussian process
   is maximized.
5. If a stopping criterion is not satisfied, add $x_{cr}$ to $\boldsymbol{S}$, set $i := i + 1$, and go to 2.
```

GPR is most useful for constructing surrogate models for a small number of independent, scalar QoIs. However, in many applications, there are many QoIs or the QoIs are related: for example, the, lift, drag, and moment coefficients in aerodynamics are related. This is where NN-based regressions are preferred over GPRs. In the absence of an error indicator for an NN-based regression, the main idea here is to efficiently train an NN-based surrogate model for a QoI (or a set of related QoIs) by using GPR as a surrogate error indicator and its acquisition function as the algorithm for sampling the next data point to train the NN.

5 SURROGATE MODELING OF AERODYNAMIC QUANTITIES OF INTEREST AND PERFORMANCE ASSESSMENTS

Here, **Algorithm 1** is applied to efficiently construct NN-based surrogate models for predicting aerodynamic QoIs for a NACA airfoil over a range of flight conditions. To this end, the HDMs are constructed using the massively parallel, three-dimensional, compressible flow solver AERO-F, which was validated for many flow problems including aircraft flow problems [9, 10]. Specifically, **Algorithm 1** is applied to adaptively sample the training parameter points as the critical points of the GP equipped with the maximum predicted variance as an acquisition function. The accuracy of the constructed NN-based surrogate model is compared to that of the GPR obtained as a byproduct of the process. It is also contrasted with the performance of two related NN-based surrogate models trained using uniform grid sampling and random sampling. A flowchart of the numerical experiments is graphically depicted in Figure 2.

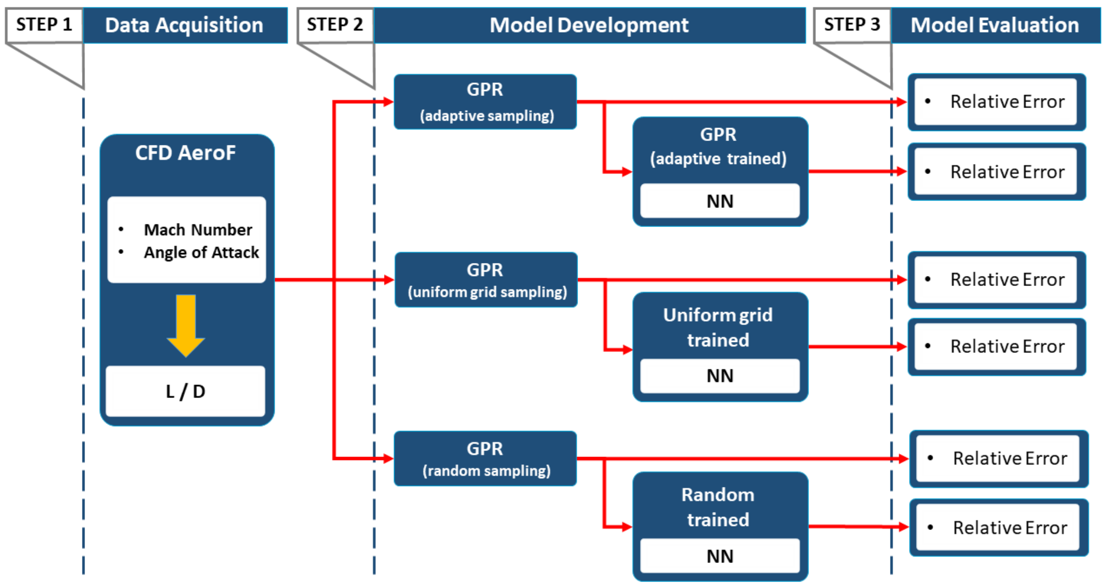

Figure 2: Flowchart for the evaluation of an adaptive approach for training an NN-based surrogate model.

5.1 Parametric Predictions of the Lift-over-drag Ratio for a NACA0012 Airfoil

The objective is set to predicting in real-time the ratio L/D, where L and D denote the lift and drag generated by a NACA0012 airfoil, respectively, in the two-dimensional parameter space of flight conditions represented by M_∞ and α_∞. Air is modelled as a perfect gas. The flow is assumed to be inviscid and modelled using the compressible Euler equations.

5.2 CFD-based Computational Model

The computational fluid domain chosen for computing all flows around the NACA0012 airfoil reported herein is a circular domain extruded in the *z* direction, because AERO-F is a three-dimensional flow solver (see Figure 3 (left)). The airfoil chord is 2.24 m and the diameter of the circle defining the computational fluid domain is 28 m. The computational fluid domain is discretized by 45,000 tetrahedral elements. Figure 3 (right) highlights the presence of local refinement near the wall boundary. Symmetry boundary conditions are applied on the circular lateral boundaries of the computational fluid domain and slip conditions are prescribed on the wall boundary.

5.3 Generation of Reference/test Data

To facilitate error measurements for the ML algorithms under consideration, a steady-state solution of the flow is computed for a range of free-stream flow conditions and postprocessed to obtain the lift-over-drag ratio L/D. Specifically, the Mach number M_∞ is varied from 0.2 to 0.5 in increments of 0.01, and the angle of attack α_∞ is varied from 0° to 5° in increments of 0.25°. The data set thus obtained is referred to below as the reference/test data set. A contour plot of the pressure solution near the airfoil for $M_\infty = 0.4$ and $\alpha_\infty = 5^\circ$ is shown in Figure 3 (right).

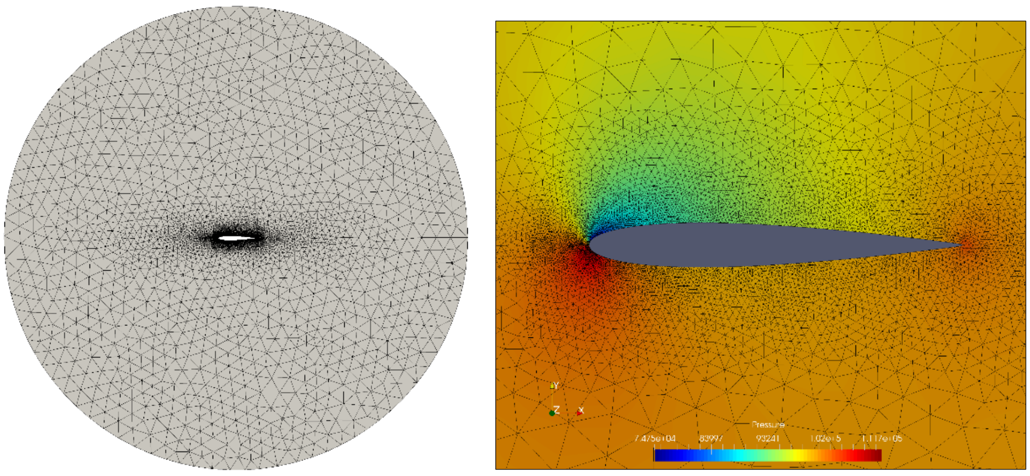

Figure 3: Parmetric flows over a NACA0012 airfoil – Computational fluid domain (left); and pressure contours at the parameter point ($M_\infty = 0.4$, $\alpha_\infty = 5^\circ$) (right).

5.4 Generation of a Training Data Set Using the Acquisition Function of a Gaussian Process Regression

Algorithm 1 (see Section 4) is applied to adaptively approximate the lift-over-drag ratio L/D in the parameter space $(M_\infty, \alpha_\infty) = ([0.2, 0.5] \times [0^\circ, 5^\circ])$. Starting from two corners of this rectangular space, additional data points are sampled by the GP and its acquisition function. Since the lift-over-drag ratio is expected to vary smoothly, an RBF kernel is employed. The `Python` package `Scikit-learn` [11] is used to evaluate the GP model and optimize the hyperparameters of the RBF kernel. To monitor convergence of the function exploration, the following heuristic error indicator is adopted

$$c_{GP} = \frac{\max(L/D)_{std}}{max(L/D)_{pred}} \tag{4}$$

where $(L/D)_{mean}$ and $(L/D)_{std}$ are the mean and standard deviation predicted by the posterior Gaussian process on a test set that is chosen here to coincide with the reference set described above. Alternative choices include a relative decrease of the maximum standard deviation, or a fixed number of iterations that is considered practically affordable (which is not the case here because a large reference set has been precomputed enable the evaluation of the errors of the procedure).

Figure 4 (left) shows the convergence of the error indicator c_{GP} and the relative error e_{GP} defined as

$$e = \frac{\|(L/D)_{pred} - (L/D)_{ref}\|}{\|(L/D)_{ref}\|} \tag{5}$$

It can be seen that both the relative error and relative error indicator drop below 1% after 18 training points have been sampled. In order to enable a comparison with training on a uniform sampling of 4 × 4 points however, only the first 16 training points selected by the greedy algorithm are retained in the remainder of this paper. Figure 4 (right) shows a good agreement between the reference ratio L/D_{ref} and its counterpart L/D_{pred} predicted by the GP using the 16 training points. The relative error using the adaptive sampling is 1.2%. For comparison, the relative error obtained by sampling on a 4 × 4 uniform grid is 7.8% and on a random sample of 16 points, it is 2.2%.

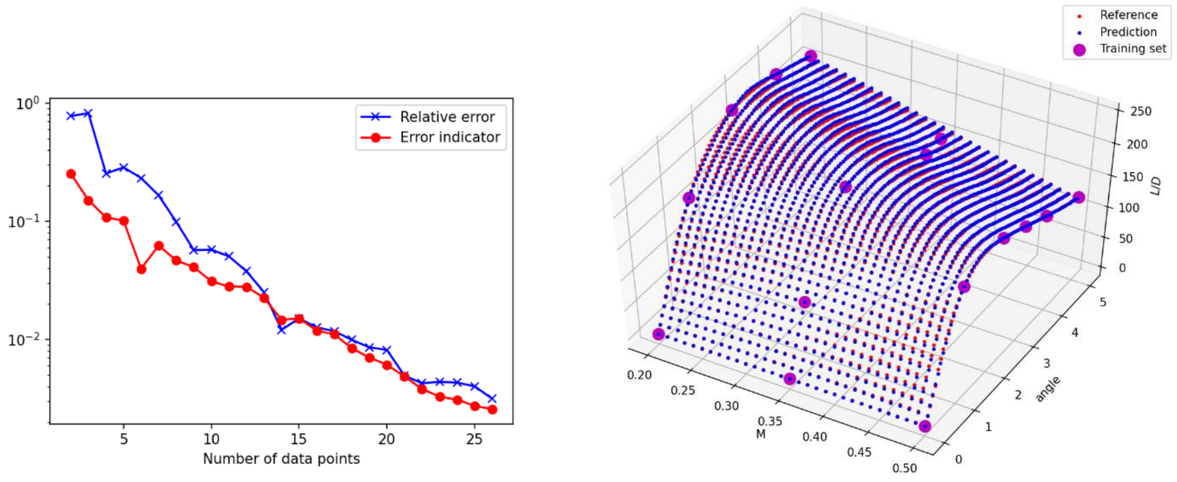

Figure 4: Parametric predictions for the NACA0012 airfoil – Convergence of GPR as a function of the number of adaptively selected training points (left); and reference as well as GPR solutions using 16 adaptively selected training points (right).

5.5 Optimal Configuration of a Neural Network for a Given Training Data Set

Using the training set adaptively determined in Section 5.1.1 via GPR, an NN-based surrogate model is constructed here. First, the performance of such a model in terms of accuracy (measured with respect to the reference training data) is discussed as a function of the chosen activation function, number of hidden layers, dimensions of the hidden layers, specific optimizer, and specific learning rate.

To this end, the NN is constructed using the `pyTorch` software package. Many activation functions (e.g., *sigmoid*, *tanh*, *ReLU*, …) are considered and best accuracy (measured as in (5)) is obtained using the $log\, softmax$ activation function. Similarly, a variety of optimizers are considered and the `Adam` optimizer [12] – a stochastic, first-order gradient-based optimization algorithm – configured with the learning rate of $4{\times}10^{-3}$ is found to deliver the best accuracy. The sensitivities of the accuracy of the NN-based surrogate model to the number of hidden layers and number or nodes in the hidden layers are reported in Table 1. They indicate that while achieving a small relative error on the training set is a necessary condition for achieving a small relative error on the test set, it is not, however, by itself a guarantee for obtaining good accuracy on the test set. The best results are obtained with an NN configured with three layers of dimensions 10, 40, and 40, respectively. This network is illustrated in Figure 5.

Table 1: Parametric predictions for the NACA0012 airfoil – Accuracy of the NN-based surrogate model trained via GPR sampling, as a function of the number of hidden layers and nodes.

H1	H2	H3	Relative error on the training set	Relative error on the test set
10	-	-	33.712%	31.403%
40	-	-	33.716%	31.458%
10	10	-	0.585%	9.308%
40	40	-	0.389%	1.675%
40	10	-	0.650%	2.099%
10	40	-	0.246%	19.322%
10	10	10	0.005%	0.940%
40	40	40	0.004%	1.095%
40	10	10	0.005%	5.434%
10	40	10	0.004%	0.911%
10	10	40	0.005%	1.034%
40	40	10	0.005%	1.486%
40	10	40	0.005%	1.464%
10	40	40	0.017%	0.572%

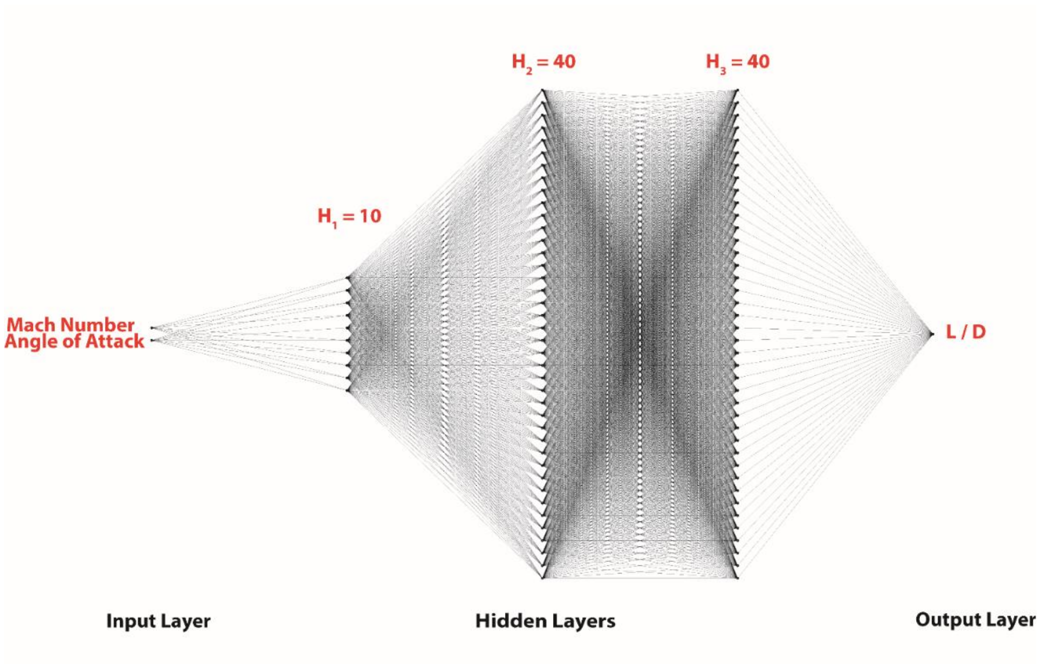

Figure 5: Parametric predictions for the NACA0012 airfoil – NN with three hidden layers of dimensions 10, 40, and 40, respectively.

5.6 Performance Comparisons

Here, the accuracy of the predictions made using the NN-based surrogate model shown in Figure 5 and trained at 16 parameter points determined via GPR sampling is compared to:

- The accuracy of two counterpart NN-based surrogate models where:
  - The best topology for the NN is determined as in Section 5.1.4, by scanning depth and the dimensions.
  - Training is performed using uniform (4×4) and random samplings, but the same total number of sampled parameter points (16).
- The accuracy of the GPR itself when constructed using any of the three aforementioned samplings.

The comparisons are performed in Table 2, where the relative errors are measured as in (5). The following observations are noteworthy:

- The accuracy obtained by any (tuned) NN-based surrogate model exceeds that of the GPR-based surrogate model trained using the same sampled parameter points. This is expected considering the superior approximation properties of deep NNs and their larger number of hyperparameters.
- The proposed adaptive sampling approach leads to the best accuracy. Specifically, it leads to relative errors that are about 7 times and 2 times smaller than those associated with uniform and random samplings, respectively.

Table 2: Parametric predictions for the NACA0012 airfoil – Evaluation of the constructed surrogate models.

Surrogate model	Relative error (training set)	Relative error (test set)
GPR (adaptive sampling)	NA	1.1%
GPR (uniform grid sampling)	NA	7.8%
GPR (random sampling)	NA	2.2%
NN (adaptive sampling)	0.017%	0.57%
NN (uniform grid sampling)	0.004%	1.54%
NN (random sampling)	0.437%	0.70%

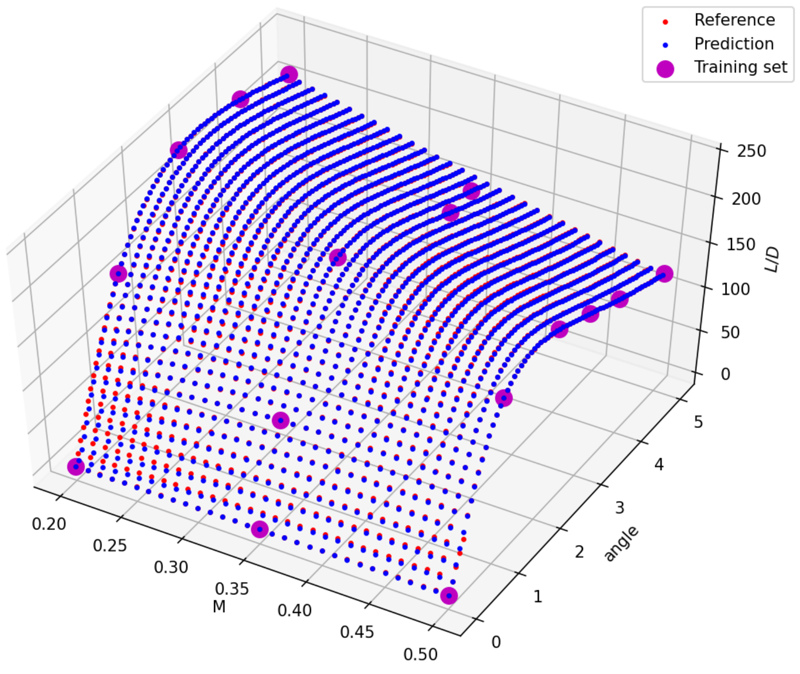

Figure 6: Parametric predictions for the NACA0012 airfoil – Predictions obtained using an NN-based surrogate model trained via GPR sampling.

6 CONCLUSIONS

This paper presents a novel approach for adaptively training a neural network (NN)-based surrogate model based on the acquisition function of a Gaussian Process (GP) regression (GPR). The approach reduces the amount of training needed to achieve a certain accuracy and therefore is particularly interesting in the context of time-critical applications such as design optimization and uncertainty quantification. Numerical experiments associated with the prediction for the NACA0012 airfoil of the lift-over-drag ratio in a large two-dimensional space of free-stream Mach number and angle of attack demonstrate the superior accuracy delivered by the proposed training method over training using uniform and random samplings.

7 ACKNOWLEDGEMENTS

The authors acknowledge support by a research grant from the King Abdulaziz City for Science and Technology (KACST). This document does not necessarily reflect the position of this institution and no official endorsement should be inferred.

REFERENCES

[1] Box J. F., R. A. Fisher and the Design of Experiments, 1922–1926, The American Statistician (1980) 34:1-7.

[2] McKay M. D., Beckman R.J., Conover W. J., A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics, American Statistical Association (1979) 21: 239-245.
[3] Carlberg K., Bou-Mosleh C., Farhat C., Efficient nonlinear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations, International Journal for Numerical Methods in Engineering (2011) 86: 155-181.
[4] Farhat C., Tezaur R., Chapman T., Avery P., Soize C., A feasible probabilistic learning method for model-form uncertainty quantification in vibration analysis, AIAA Journal (2019) 57:1-14.
[5] Rasmussen C. E., Gaussian processes in machine learning, Advanced Lectures on Machine Learning, Springer (2004) 63-71.
[6] Stulp F., Sigaud O., Many regression algorithms, one unified model: A review, Neural Networks (2015) 69:60-79.
[7] Csáji B. C., Approximation with artificial neural networks, Faculty of Sciences (2001), Eötvös Loránd University, Hungary.
[8] Krause A., Singh A., Guestrin, C. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies, Journal of Machine Learning Research (2008) 9:235-284.
[9] Geuzaine P., Brown G., Harris C., Farhat C., Aeroelastic dynamic analysis of a full F-16 configuration for various flight conditions, AIAA Journal (2003) 41:363–371.
[10] Farhat C., Geuzaine P., Brown G., Application of a three-field nonlinear fluid-structure formulation to the prediction of the aeroelastic parameters of an F-16 fighter, Computers and Fluids (2003) 32:3–29.
[11] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss J., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E., Scikit-learn: Machine learning in Python, Jounal Machine Learning Research (2011) 12:2825-2830.
[12] Kingma D. P., Ba J., Adam: A method for stochastic optimization, 3rd International Conference for Learning Representations, San Diego (2015).