Proceedings of the ASME 2021 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2021
August 17-20, 2021, Online, Virtual

IDETC2021-70502

GRADIENT-ENHANCED MULTIFIDELITY NEURAL NETWORKS FOR HIGH-DIMENSIONAL FUNCTION APPROXIMATION

Jethro Nagawkar
Department of Aerospace Engineering
Iowa State University
Ames, Iowa, 50011
Email: jethro@iastate.edu

Leifur Leifsson*
Department of Aerospace Engineering
Iowa State University
Ames, Iowa, 50011
Email: leifur@iastate.edu

ABSTRACT

In this work, a novel multifidelity machine learning (ML) algorithm, the gradient-enhanced multifidelity neural networks (GEMFNN) algorithm, is proposed. This is a multifidelity extension of the gradient-enhanced neural networks (GENN) algorithm as it uses both function and gradient information available at multiple levels of fidelity to make function approximations. Its construction is similar to the multifidelity neural networks (MFNN) algorithm. The proposed algorithm is tested on three analytical functions, a one, two, and a 20 variable function. Its performance is compared to the performance of neural networks (NN), GENN, and MFNN, in terms of the number of samples required to reach a global accuracy of 0.99 of the coefficient of determination (R^2). The results showed that GEMFNN required 18, 120, and 600 high-fidelity samples for the one, two, and 20 dimensional cases, respectively, to meet the target accuracy. NN performed best on the one variable case, requiring only ten samples, while GENN worked best on the two variable case, requiring 120 samples. GEMFNN worked best for the 20 variable case, while requiring nearly eight times fewer samples than its nearest competitor, GENN. For this case, NN and MFNN did not reach the target global accuracy even after using 10,000 high-fidelity samples. This work demonstrates the benefits of using gradient as well as multifidelity information in NN for high-dimensional problems.

*Address all correspondence to this author.

INTRODUCTION

Surrogate modeling methods are useful for reducing the computational cost in various problems involving optimum design [1, 2], uncertainty quantification [3, 4], and global sensitivity analysis [5, 6]. In these methods, a computationally efficient surrogate model replaces an expensive physics-based model, reducing the overall cost involved in solving such problems.

Surrogate modeling methods can be broadly classified as either being data-fit methods [7] or multifidelity methods [8]. In data-fit methods, a response surface is fitted through evaluated single-fidelity sample points. While in multifidelity methods, low-fidelity data is used to augment the predictive capabilities of surrogate models constructed from a limited number of high-fidelity data. High-fidelity models are models that solve the task at hand with a desired accuracy. Low-fidelity models solve the same task but at lower cost and with lower accuracy.

A variety data-fit methods exist in literature such as Kriging [9] (also known as Gaussian process regression) and its variants [10–12], polynomial chaos expansions [13], and support vector machines [14]. Of these methods, Kriging is the most widely used method in various engineering analysis and design tasks [15]. Kriging, however, suffers from a variety of issues such as being poor at approximating discontinuous functions [16], difficulty in handling high-dimensional problems [17], costly to use in the presence of a large number of data samples [15], and being difficult to implement [18]. Several methods have been introduced to handle these issues such as the use of gradient information [15, 19], and the partial least-squares correlation func-

tions [11, 12]. While improvements have been reported, several of the key issues still remain [18, 20].

Multifidelity methods have been introduced as a way of reducing the overall cost involved in engineering design and analysis tasks [8, 15]. The key benefit is the use of low-fidelity data, along with a limited number of high-fidelity data, reducing the overall cost in acquiring the data to construct the surrogate model. Cokriging [21], the multifidelity version of Kriging, and its variants [22, 23], while becoming popular in design and analysis tasks, still suffers from the issues associated with Kriging [18].

The use of neural networks (NN) [24] in engineering design and analysis problems is becoming more prevalent [18, 20]. NN overcome many of the major challenges in Kriging models such as the ability to handle high-dimensional datasets [20], scalability with the number of data [24], easier to implement [18], as well as being good at approximating discontinuous data [18]. The major drawback of NN is that they require a large number of samples to make accurate predictions [24], especially for high-dimensional problems [20]. To overcome this challenge, different NN variants, such as gradient-enhanced NN (GENN) [20,25], and multifidelity NN (MFNN) [18], have been recently introduced.

In this work, a multifidelity variant of GENN [25], gradient-enhanced MFNN (GEMFNN) is introduced and demonstrated on three different analytical problems, involving one, two, and 20 variables. The proposed approach is compared to NN [24], GENN [25], and MFNN [18] for these problems. GEMFNN are constructed in a similar fashion as MFNN by leveraging both function and gradient information available from low- and high-fidelity models to yield accurate function approximations. To the author's knowledge, the proposed GEMFNN is a novel machine learning (ML) modeling algorithm.

The remainder of this paper is organized in the following way. The next section describes the methods used to construct the GEMFNN ML algorithm. In the following section, the GEMFNN algorithm is demonstrated on three analytical benchmark problems. This paper then ends with the conclusion and future work.

METHODS

This section describes the construction of the GEMFNN ML algorithm. An outline of the GEMFNN-based analysis is first introduced, followed by the sampling plan used to generate the data, which is needed in order to train and test this ML algorithm. The construction methodology of the GEMFNN algorithm is discussed in the following section, followed by the validation metric used to quantify its global accuracy. Finally, this algorithm can be used for further analysis, such as optimal design and uncertainty quantification.

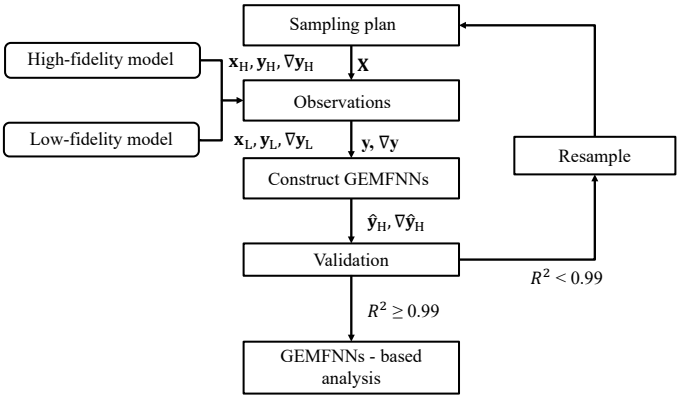

FIGURE 1. FLOWCHART OF THE GRADIENT-ENHANCED MULTIFIDELITY NEURAL NETWORK CONSTRUCTION.

Outline of the GEMFNN Construction

A flowchart of the GEMFNN construction is shown in Fig. 1. It begins by sampling the input design space, $\mathbf{X} \in \mathbb{R}^{m\times n}$, first, in order to generate the data required to both train and test the ML algorithm. m is the number of samples and n is the number of input variables. The training data consists of two different sample sets, one for evaluating the high-fidelity model, $\mathbf{x}_H \in \mathbb{R}^{m_H\times n}$, and the other for evaluating the low-fidelity model, $\mathbf{x}_L \in \mathbb{R}^{m_L\times n}$. m_H and m_L are the number of high- and low-fidelity samples, respectively. Both the function and its gradients need to be evaluated at the high- ($\mathbf{y}_H \in \mathbb{R}^{m_H\times 1}$ and $\nabla\mathbf{y}_H \in \mathbb{R}^{m_H\times n}$) and low-fidelity models ($\mathbf{y}_L \in \mathbb{R}^{m_L\times 1}$ and $\nabla\mathbf{y}_L \in \mathbb{R}^{m_L\times n}$), respectively. $\mathbf{y}_L$ and $\mathbf{y}_H$ together represent the combined observation $\mathbf{y} \in \mathbb{R}^{m\times 1}$, while $\nabla\mathbf{y}_L$ and $\nabla\mathbf{y}_H$ together represent the combined observation $\nabla\mathbf{y} \in \mathbb{R}^{m\times n}$. A separate testing set is created by evaluating only the high-fidelity model's function and its gradients. $\hat{\mathbf{y}}_H \in \mathbb{R}^{m_H\times 1}$ and $\nabla\hat{\mathbf{y}}_H \in \mathbb{R}^{m_H\times n}$ are the high-fidelity function and gradient predictions, respectively, from the GEMFNN. The accuracy of these predictions are then measured using the coefficient of determination (R^2) error metric. The above process is repeated several times, each with an increasing training sample size, until terminating on the validation criteria. On terminating, GEMFNN can be used for further analysis in engineering design and analysis.

Sampling Plan

Sampling is the first process involved in constructing the GEMFNN. It is the process of selecting discrete samples in the variable space [15]. In this study , both the full factorial sampling plan [15], as well as the Latin Hypercube sampling (LHS) [26] plan are used to generate both the training and testing data. The choice of the sampling plan used is case dependent and is discussed in their corresponding sections.

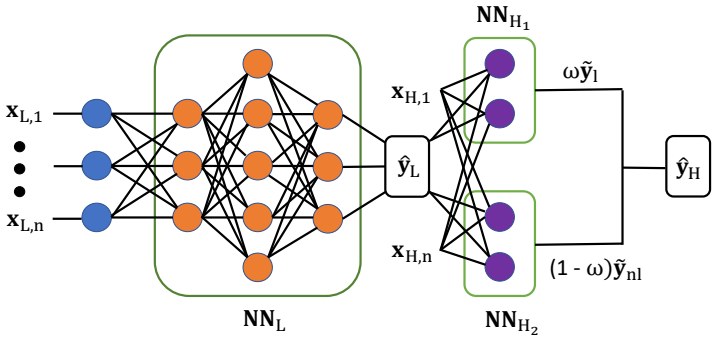

FIGURE 2. MULTIFIDELITY NEURAL NETWORK ARCHITECTURE (ADAPTED FROM [18]).

Gradient-Enhanced Multifidelity Neural Networks

NN are universal function approximators [24], where a hierarchy of features, known as layers, is used to approximated any given function. The layers in-between the input and output layers are called hidden layers. The output and hidden layers contain neurons, which are a fundamental unit of computation and contain an activation function [24]. In NN, an unconstrained optimization problem is solved, where the parameters of the NN are tuned using the Adaptive Moments (ADAM) [27] gradient-based optimizer [24], where the backpropagation algorithm [28] is used to compute the gradients. In this study, the mean squared error (MSE) is used as the loss function in the optimization problem is given by

$$\mathscr{L}_{NN} = \frac{\sum_{l=1}^{N}(\hat{y}_{\mathrm{H}}^{(l)} - y_{\mathrm{H}}^{(l)})^2}{N}, \tag{1}$$

where N the number of samples in a subset of the training data, called mini-batch [24], is used to minimize the mismatch between the high-fidelity training data observations, y_{H}, and the predicted values, $\hat{y}_{\mathrm{H}}$, of the NN.

GENN [20] modify the loss function in (1) by adding the mismatch match between the high-fidelity training data gradient, ∇y_{H}, and the predicted gradient of the NN, $\nabla\hat{y}_{\mathrm{H}}$, to it, and is given by

$$\mathscr{L}_{GENN} = \mathscr{L}_{NN} + \frac{\sum_{l=1}^{N}\sum_{k=1}^{D}(\nabla\hat{y}_{\mathrm{H},k}^{(l)} - \nabla y_{\mathrm{H},k}^{(l)})^2}{N}, \tag{2}$$

where D is the dimension of the input variable space. This loss function ensures a reduction in the mismatch between both the function and its tangent at a given training point to the corresponding true values, respectively.

A schematic of the MFNN [18] architecture is shown in Fig. 2. It contains three NN, NN_L, which is used to approximate the low-fidelity data, the output of which is used as an additional input variable to two other NN, NN_{H_1} and NN_{H_2}. NN_{H_1} and NN_{H_2} are used to capture the linear ($\tilde{y}_l$) and nonlinear correlations ($\tilde{y}_{nl}$) between high- and low-fidelity data. NN_{H_1} contains linear activation functions, while NN_{H_2} contains nonlinear activation functions. The weighted sum of the outputs of the linear and nonlinear layer gives the high-fidelity prediction of the MFNN as

$$\hat{y}_{\mathrm{H}} = \omega\tilde{y}_l + (1-\omega)\tilde{y}_{nl}, \tag{3}$$

where ω is an additional parameter of the MFNN. Note that NN and GENN use only the NN_{H_2} part of MFNN, but do not include $\hat{y}_{\mathrm{L}}$ as an additional input parameter. The loss function of MFNN is given as

$$\mathscr{L}_{MFNN} = \mathscr{L}_{NN} + \frac{\sum_{l=1}^{N}(\hat{y}_{\mathrm{L}}^{(l)} - y_{\mathrm{L}}^{(l)})^2}{N}. \tag{4}$$

The proposed GEMFNN algorithm is a novel multifidelity version of GENN [25] and is constructed similar to MFNN [18]. It uses gradient information available at high and low-fidelity data during training. The new and unique loss function for GEMFNN is taken as

$$\begin{aligned}\mathscr{L}_{GEMFNN} =& \mathscr{L}_{MFNN} + \frac{\sum_{l=1}^{N}\sum_{k=1}^{D}(\nabla\hat{y}_{\mathrm{L},k}^{(l)} - \nabla y_{\mathrm{L},k}^{(l)})^2}{N} \\ &+ \frac{\sum_{l=1}^{N}\sum_{k=1}^{D}(\nabla\hat{y}_{\mathrm{H},k}^{(l)} - \nabla y_{\mathrm{H},k}^{(l)})^2}{N}.\end{aligned} \tag{5}$$

The steps in the GEMFNN algorithm are as follows:

1. Normalize the input, output and gradient of output with respect the inputs for all the data.
2. Perform forward propagation through NN_L to get $\hat{y}_{\mathrm{L}}$. Then do the same through the MFNN to get $\hat{y}_{\mathrm{H}}$.
3. Use reverse mode automatic differentiation [29] to calculate both $\nabla\hat{y}_{\mathrm{L}}$ and $\nabla\hat{y}_{\mathrm{H}}$.
4. Calculate $\mathscr{L}_{GEMFNN}$ using (5).
5. Use backpropagation [28] to calculate the gradient of $\mathscr{L}_{GEMFNN}$ with respect to all the parameters in GEMFNN (θ), given by $\nabla\mathscr{L}_{GEMFNN}$.
6. Update the parameters:

$$\theta \leftarrow \theta - \alpha\nabla\mathscr{L}_{GEMFNN}, \tag{6}$$

where α is the learning rate hyperparameter.

7. Iterate over steps $2-6$ till all the mini-batches present in one epoch is used. One epoch refers to one iteration over an entire training dataset [24].
8. Repeat steps $2-7$ for all the epochs.
9. GEMFNN is now trained and ready to be used for function approximation.

Validation

In this work, the coefficient of determination is used to measure the global accuracy of the ML algorithms, R^2, given as

$$R^2 = 1 - \frac{\sum_{j=1}^{N_t}(y_t^{(j)} - \hat{y}_t^{(j)})^2}{\sum_{j=1}^{N_t}(y_t^{(j)} - \bar{y}_t)^2}, \tag{7}$$

where N_t is the total number of testing data samples in one dataset, $\hat{y}_t^{(j)}$ and $y_t^{(j)}$ are the ML algorithm estimation and high-fidelity observation of the j^{th} testing point, respectively, and $\bar{y}_t$ is the mean of $y_t^{(j)}$, given by

$$\bar{y}_t = \frac{\sum_{j=1}^{N_t} y_t^{(j)}}{N_t}. \tag{8}$$

R^2 is the measure of "Goodness of fit" [15] of an algorithm. When R^2 equals one, the algorithm has approximated the true function perfectly. $R^2 = 0$ implies that the algorithm prediction is the same as the mean of the true function, while R^2 less than zero implies that the mean of the true function is better than the algorithm at predicting the trend of the true function. This makes it easier to chose the global accuracy criterion. In this work, a value of R^2 greater than 0.99 is considered an acceptable global accuracy.

In this work, for each high-fidelity sample size and for each ML algorithm, the mean and the standard deviation of the R^2 metric is plotted using 'n_t' different datasets. This is done in order to account for the variation in the training data used as well as due to the stochastic nature of the ADAM optimizer [27]. The mean of R^2 is

$$\mu_{R^2} = \frac{\sum_{k=1}^{n_t} R_k^2}{n_t}, \tag{9}$$

and the standard deviation is

$$\sigma_{R^2} = \sqrt{\frac{\sum_{k=1}^{n_t}(R_k^2 - \mu_{R^2})^2}{n_t}}. \tag{10}$$

In this work, n_t is set to ten for all the cases.

TABLE 1. ONE DIMENSIONAL FUNCTION MODELING COST.

ML algorithm	Modeling cost
NN	10
GENN	12*
MFNN	12**
GEMFNN	18*,**

*Function plus gradient evaluation cost

**Plus 50 low-fidelity training samples

NUMERICAL EXAMPLES

In this study, the GEMFNN ML algorithm is demonstrated on three different analytical problems. The first is an one variable analytical function. The second, the two variable Rastrigin function [30], and the final a 20 variable analytical function. The GEMFNN ML algorithm is compared to other ML algorithms, namely, NN, GENN, and MFNN.

Case 1: One Dimensional Analytical Function

The one dimensional analytical function used in this study was developed by Forrester et al. [15] and is written as

$$f_{\text{HF}}(\mathbf{x}) = (6x-2)^2 \sin(12x-4), \tag{11}$$

where $x \in [0,1]$. Forrester et al. [15] also introduced a corresponding low-fidelity model given by

$$f_{\text{LF}}(\mathbf{x}) = 0.5 f_{\text{HF}}(\mathbf{x}) + 10(x-0.5) - 5. \tag{12}$$

The corresponding gradient of the high-fidelity model is

$$\begin{aligned}\nabla f_{\text{HF}}(\mathbf{x}) =& 12(6x-2)\sin(12x-4) \\ &+ 12(6x-2)^2\cos(12x-4),\end{aligned} \tag{13}$$

and the low-fidelity model gradient is

$$\nabla f_{\text{LF}}(\mathbf{x}) = 0.5\nabla f_{\text{HF}}(\mathbf{x}) + 10. \tag{14}$$

ML Algorithm Setup In this case, the full factorial sampling plan is used to generate both the high- and low-fidelity

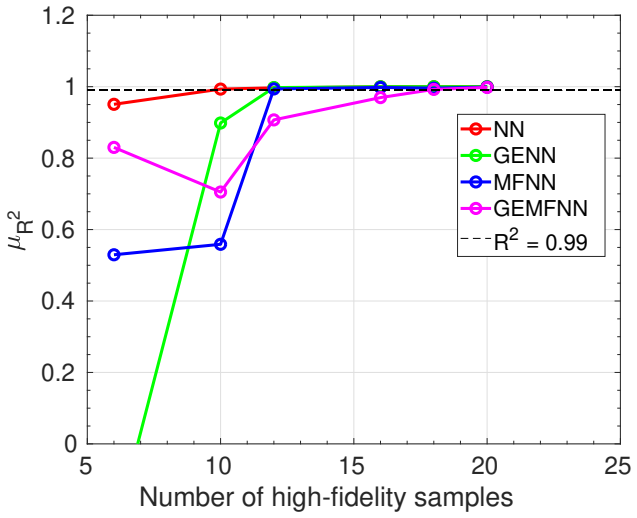

FIGURE 3. MEAN OF R^2 FOR CASE 1.

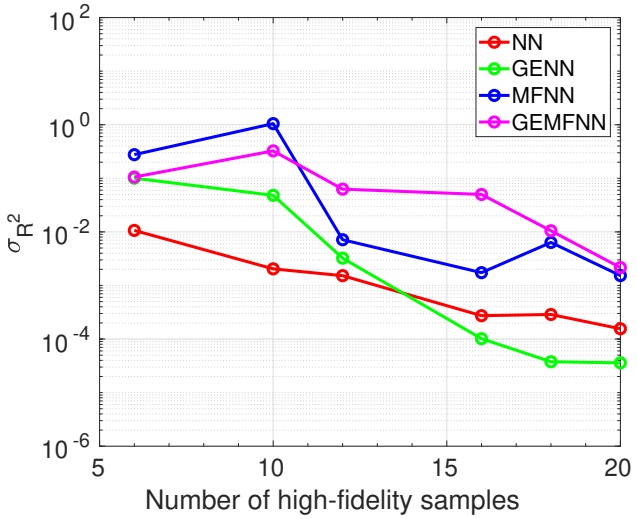

FIGURE 4. STANDARD DEVIATION OF R^2 FOR CASE 1.

training data, as well as the high-fidelity testing data. The testing data contains 1,000 samples. As discussed the in the previous section, NN and GENN use only the NN_{H_2} section of the composite NN, while MFNN and GEMFNN use NN_L, NN_{H_1}, and NN_{H_2}. Note that the various hyperparameters used for this case are kept the same for the different ML algorithms. NN_{H_1} and NN_{H_2} for this case is set to include only one hidden layer, with ten neurons, while NN_L has 20 neurons and one hidden layer. Both NN_L and NN_{H_2} use the tangent hyperbolic activation function, while NN_{H_1} uses a linear activation function. The learning rate is set to 0.001, while the batch size is fixed with a value of 10. The maximum number of epochs used in this case is 15,000. No regularization is used while training the different ML algorithms.

Results Table 1 shows the number of high-fidelity samples required by each ML algorithm to reach the global accuracy of $R^2 = 0.99$. The multifidelity algorithms use an additional of 50 low-fidelity sample points. Note that the number of sampling points for the gradient-enhanced cases accounts for both the cost of evaluating the function and its gradient. Therefore, 200 sampling points for these cases, for example, correspond to 100 function and 100 gradient evaluations. Figure 3 shows that NN outperformed all the other algorithms, requiring only ten high-fidelity samples to reach the global accuracy. GENN and MFNN both required twelve samples each to reach this accuracy, while GEMFNN required 18 samples. The results in Fig. 3 are generated by averaging the outcomes from ten different datasets. The corresponding standard deviation for each algorithm is shown in Fig. 4. Figure 4 shows that the standard deviation decrease with increase number of samples, resulting in algorithms that are less sensitive to the training data, as well as due to the stochastic nature of the ADAM optimizer [27]. For this case, the cost of evaluating the low-fidelity model is neglected. The results for this case imply that there is no benefit of using gradients as well as data from different levels of fidelity in NN when the model is of a low-dimensional input space.

Case 2: Two Dimensional Rastrigin Function

The two dimensional Rastrigin function [30] used in this study is written as

$$f_{\mathrm{HF}}(\mathbf{x}) = 20 + \sum_{i=1}^{2}(x_i^2 - 10\cos(2\pi x_i)), \tag{15}$$

where $\mathbf{x} \in$ [-1,1.5]. For this study the low-fidelity model

$$f_{\mathrm{LF}}(\mathbf{x}) = 0.5 f_{\mathrm{HF}}(\mathbf{x}) + \sum_{i=1}^{2}(x_i - 0.5) \tag{16}$$

is introduced. The corresponding high-fidelity model gradient is

$$\nabla f_{\mathrm{HF},i}(\mathbf{x}) = 2x_i + 20\pi\sin(2\pi x_i), \tag{17}$$

where $i = 1,2$, and the low-fidelity model gradient is

$$\nabla f_{\mathrm{LF},i}(\mathbf{x}) = 0.5\nabla f_{\mathrm{HF},i}(\mathbf{x}) + 1. \tag{18}$$

TABLE 2. RASTRIGIN FUNCTION MODELING COST.

ML algorithm	Modeling cost
NN	150
GENN	120*
MFNN	150**
GEMFNN	120*,**

*Function plus gradient evaluation cost

**Plus 500 low-fidelity training samples

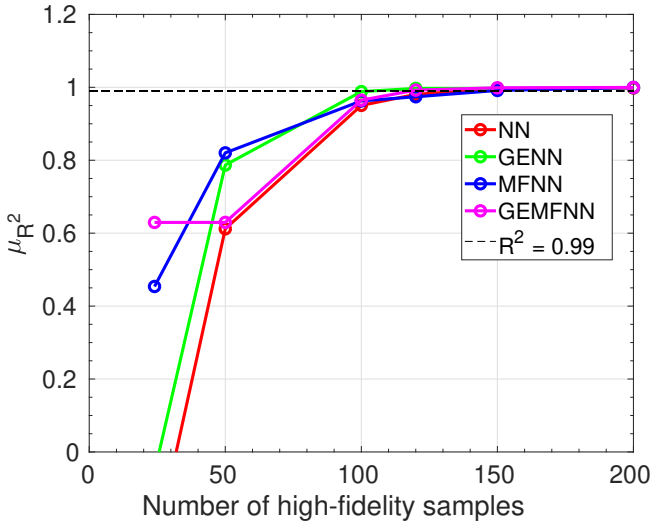

FIGURE 5. MEAN OF R^2 FOR CASE 2.

ML Algorithm Setup The LHS plan is used to generate the low- and high-fidelity training data, while the full factorial sampling plan is used to generate the testing data. The testing dataset consists of 10,000 samples. NN_L and NN_{H_2} use two hidden layers, with 50 neurons each, while NN_{H_1} contains only one hidden layer with ten neurons. Similar to the previous case, NN_L and NN_{H_2} use the tangent hyperbolic activation function, and NN_{H_1} contains a linear activation function. The batch size, the learning rate and total number epochs used in this study are set to 32, 0.001, and 10,000, respectively. No regularization was used.

Results The number of high-fidelity samples required to reach the global accuracy threshold of $R^2 = 0.99$ is shown in Table 2. The multifidelity algorithms also use an additional of 500 low-fidelity samples during its construction. The variation of the R^2 error metric with number of high-fidelity samples is shown in Fig. 5. The results in Fig. 5 are averaged over ten different datasets. Figure 6 shows the corresponding standard deviations for the same datasets. GENN and GEMFNN both require 120 high-fidelity sample points to reach the target threshold. NN and MFNN, other the other hand, require 150 high-fidelity samples to reach the same threshold. Similar to the previous case, the standard deviation of the R^2 metric decrease with increasing sample size, as seen in Fig. 6. For this case, the use of multifidelity data does not add to an improvement in predictive capabilities of the ML algorithms, however, using gradient information does. GENN, hence, work best for modeling the Rastrigin function.

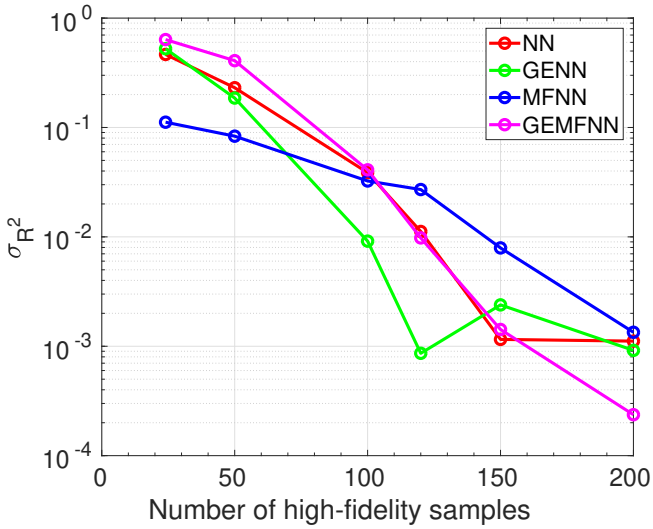

FIGURE 6. STANDARD DEVIATION OF R^2 FOR CASE 2.

Case 3: 20 Dimensional Analytical Function

The high-fidelity model of the 20 dimensional analytical function [18] is written as

$$f_{\mathrm{HF}}(\mathbf{x}) = (x_1 - 1)^2 + \sum_{i=2}^{20} (2x_i^2 - x_{i-1})^2, \tag{19}$$

where $\mathbf{x} \in$ [-3,3]. The corresponding low-fidelity model is [18]

$$f_{\mathrm{LF}}(\mathbf{x}) = 0.8 f_{\mathrm{HF}}(\mathbf{x}) - \sum_{i=1}^{19} 0.4 x_i x_{i+1} - 50. \tag{20}$$

The high-fidelity gradient is

$$\nabla f_{\mathrm{HF},1}(\mathbf{x}) = 2(x_1 - 1) - 2(2x_2^2 - x_1), \tag{21}$$

TABLE 3. 20 DIMENSIONAL FUNCTION MODELING COST.

ML algorithm	HF sample cost
NN	>10,000
GENN	5,000*
MFNN	>10,000**
GEMFNN	600*,**

*Function plus gradient evaluation cost

**Plus 30,000 low-fidelity training samples

$$\nabla f_{\mathrm{HF},i}(\mathbf{x}) = 8x_i(2x_i^2 - x_{i-1}) - 2(2x_{i+1}^2 - x_i), \tag{22}$$

for $i = 2,3,...,19$, and

$$\nabla f_{\mathrm{HF},20}(\mathbf{x}) = 8x_{20}(2x_{20}^2 - x_{19}). \tag{23}$$

The low-fidelity gradient is

$$\nabla f_{\mathrm{LF},1}(\mathbf{x}) = 0.8\nabla f_{\mathrm{HF},1}(\mathbf{x}) - 0.4x_2, \tag{24}$$

$$\nabla f_{\mathrm{LF},i}(\mathbf{x}) = 0.8\nabla f_{\mathrm{HF},i}(\mathbf{x}) - 0.4(x_{i-1} + x_{i+1}), \tag{25}$$

for $i = 2,3,...,19$, and

$$\nabla f_{\mathrm{LF},20}(\mathbf{x}) = 0.8\nabla f_{\mathrm{HF},20}(\mathbf{x}) - 0.4x_{19}. \tag{26}$$

ML Algorithm Setup For this case, the LHS plan was used to generate the training and testing data. The testing data contains 10,000 samples. Similar to the previous two cases, NN_{H_1} contains only one hidden layer with ten neurons. NN_{H_2} contains four hidden layers, with 64 neurons, while NN_L contains six hidden layers with 128 neurons. No regularization was used, while the batch size, learning rate, and number of epochs were set to 64, 0.001, and 10,000, respectively.

Results Table 3 shows the high-fidelity sample cost required by each ML algorithm to reach the target accuracy. GEMFNN far outperforms the other algorithms and requires around eight times fewer samples compared to its nearest competitor, GENN, which requires 5,000 samples. Both NN and MFNN fail to meet the target accuracy, even with 10,000 samples. Figures 7 and 8 show the mean and standard deviations, respectively, of R^2 with respect to the number of high-fidelity samples, performed using ten different datasets. The mean of R^2 increases, while the standard deviation of R^2 decreases with increasing sample sizes. The benefit of both gradient and multifidelity information in training the NN is demonstrated for a high-dimensional problem and is shown in this case.

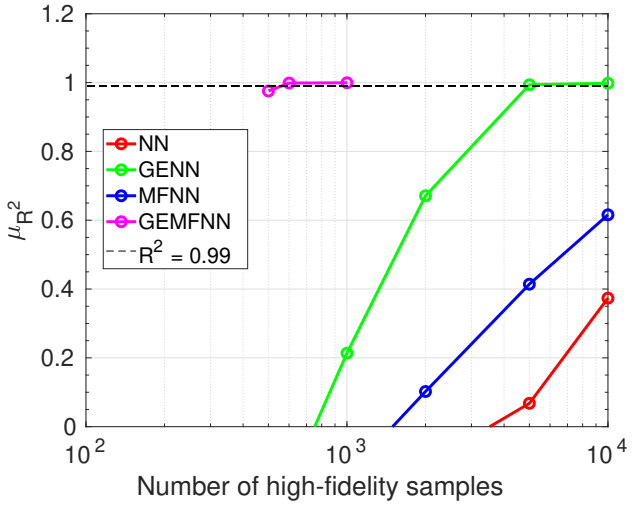

FIGURE 7. MEAN OF R^2 FOR CASE 3.

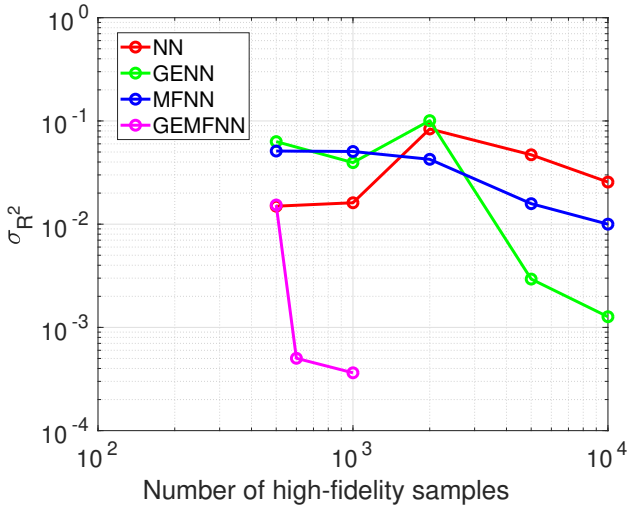

FIGURE 8. STANDARD DEVIATION OF R^2 FOR CASE 3.

CONCLUSION

The GEMFNN ML algorithm is applied to three analytical cases, namely, a one, two and a 20 dimensional variable problem, and is compared to NN, GENN, and MFNN. GEMFNN are a multifidelity version of GENN and its construction is similar to MFNN. GEMFNN consists of three NN, one to approximate the low-fidelity data (NN_L), which is then connected to two other NN, one with linear (NN_{H_1}) and the other with nonlinear (NN_{H_2}) activation functions, in order to capture both linear and nonlinear correlations, respectively, between high- and low-fidelity data. In NN, the loss function used is the MSE between the true and predicted function values. In GENN, this loss function is modified by adding the MSE of the true and predicted gradient values to the original loss function.

NN outperformed all the other algorithms in the one dimensional case, while GENN did the same for the two dimensional case and GEMFNN for the 20 variable case. This study shows the benefit of using both gradient and multifidelity information in training the ML algorithms for high-dimensional cases. It also shows, that for low dimensional cases, using multifidelity information does not improve predictive performance of the ML algorithms.

In this study, for each case, the same hyperparameters are used for all the algorithms. This may not be ideal for individual algorithms as they might perform better with different hyperparameters. Using different hyperparameters such as different activation functions, the addition of regularization as well as the varying the number of hidden layers and the number of neuron in each hidden layers will need to be done.

This study has been conducted using analytical benchmark cases. While they do not represent engineering problems, they are a good starting point in testing ML algorithms. Future applications of the above algorithms will done on problems involving optimum design, uncertainty quantification and global sensitivity analysis.

ACKNOWLEDGMENT

This material is based upon work supported in part by the National Science Foundation under grant number 1846862 and by the Department of Energy under a Laboratory Directed Research and Development grant at Ames Laboratory.

REFERENCES

[1] Li, J., Bouhlel, M. A., and Martins, J. R. R. A., 2019. "Data-based approach for fast airfoil analysis and optimization". *AIAA Journal,* ***57***(2), pp. 581–596.

[2] Nagawkar, J., Ren, J., Du, X., Leifsson, L., and Koziel, S., 2021. "Single- and multipoint aerodynamic shape optimization using multifidelity models and manifold mapping". *Journal of Aircraft, Articles in Advance*, pp. 1–18.

[3] Nagawkar, J., Leifsson, L., and Du, X., 2020. "Applications of polynomial chaos-based cokriging to aerodynamic design optimization benchmark problems". *AIAA Scitech 2020 Forum, 6-10 January, Orlando Florida*.

[4] Du, X., and Leifsson, L., 2019. "Optimum aerodynamic shape design under uncertainty by utility theory and metamodeling". *Aerospace Science and Technology,* ***95***, p. 105464.

[5] Wang, P., Lu, Z., and Tang, Z., 2013. "An application of the kriging method in global sensitivity analysis with parameter uncertainty". *Applied Mathematical Modelling,* ***37***(9), pp. 6543–6555.

[6] Du, X., Leifsson, L., Meeker, W., Gurrala, P., Song, J., and Roberts, R., 2019. "Efficient Model-Assisted Probability of Detection and Sensitivity Analysis for Ultrasonic Testing Simulations Using Stochastic Metamodeling". *Journal of Nondestructive Evaluation, Diagnostics and Prognostics of Engineering Systems,* ***2***(4), p. 041002.

[7] Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., and Tucker, P. K., 2005. "Surrogate-based analysis and optimization". *Progress in Aerospace Sciences,* ***21***(1), pp. 1–28.

[8] Peherstorfer, B., Wilcox, K., and Gunzburger, M., 2018. "Survey of multifidelity methodsin uncertainty propagation, inference, and optimization". *Society for Industrial and Applied Mathematics,* ***60***(3), pp. 550–591.

[9] Krige, D. G., 1951. "Statistical approach to some basic mine valuation problems on the witwatersrand". *Journal of the Chemical, Metallurgical and Mining Engineering Society of South Africa,* ***52***(6), pp. 119–139.

[10] Schobi, R., Sudret, B., and Wairt, J., 2015. "Polynomial-chaos-based kriging". *International Journal of Uncertainty Quantification,* ***5***, pp. 193–206.

[11] Bouhlel, M. A., Bartoli, N., Otsmane, A., and Morlier, J., 2016. "Improving kriging surrogates of high-dimensional design models by partial least squares dimension reduction". *Structural and Multidisciplinary Optimization,* ***53***(5), p. 935–952.

[12] Bouhlel, M. A., Bartoli, N., Otsmane, A., and Morlier, J., 2016. "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***(5).

[13] Blatman, G., 2009. "Adaptive sparse polynomial chaos expansion for uncertainty propagation and sensitivity analysis". PhD Thesis, Blaise Pascal University, France.

[14] Li, D., Wilson, P. A., and Jiong, Z., 2015. "An Improved Support Vector Regression and Its Modelling of Manoeuvring Performance in Multidisciplinary Ship Design Optimization". *International Journal of Modelling and Simulation,* ***35***, pp. 122–128.

[15] Forrester, A. I. J., Sobester, A., and Keane, A. J., 2008.

Engineering design via surrogate modelling: A practical guide. John Wiley and Sons, Ltd, United Kingdom.

[16] Raissi, M., and Karniadakis, G., 2016. Deep multi-fidelity gaussian processes. arXiv:1604.07484.

[17] Perdikaris, P., Raissi, M., Damianou, A., Lawrence, N. D., and Karniadakis, G. E., 2017. "Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling". *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences,* ***473***(2198), p. 20160751.

[18] Meng, X., and Karniadakis, G. E., 2020. "A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems". *Journal of Computational Physics,* ***401***, p. 109020.

[19] Bouhlel, M. A., and Martins, J. R., 2019. "Gradient-enhanced kriging for high-dimensional problems". *Engineering with Computers,* ***35***(1), p. 157–173.

[20] Bouhlel, M. A., He, S., and Martins, J., 2020. "Scalable gradient–enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes". *Structural and Multidisciplinary Optimization,* ***61***(4), p. 1363–1376.

[21] Kennedy, M. C., and O'Hagan, A., 2000. "Predicting the output from a complex computer code when fast approximations are available". *Biometrika Trust,* ***87***(1), pp. 1–13.

[22] Du, X., and Leifsson, L., 2020. "Multifidelity modeling by polynomial chaos-based cokriging to enable efficient model-based reliability analysis of ndt systems". *Journal of Nondestructive Evaluation,* ***39***(1).

[23] Deng, Y., 2020. "Multifidelity data fusion via gradient-enhanced gaussian process regression". *Communications in Computational Physics,* ***28***(5), p. 1812–1837.

[24] Goodfellow, I., Bengio, Y., and Courville, A., 2016. *Deep Learning*. The MIT Press, Cambridge, MA.

[25] Czarnecki, W. M., Osindero, S., Jaderberg, M., Świrszcz, G., and Pascanu, R., 2017. "Sobolev training for neural networks". *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA*.

[26] McKay, M. D., Beckman, R. J., and Conover, W. J., 1979. "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code". *Technometrics,* ***21***(2), pp. 239–245.

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

[28] Chauvin, Y., and Rumelhart, D. E., 1995. *Backpropagation: theory, architectures, and applications*. Psychology press, Hillsdale, NJ.

[29] Rall, L. B., 1981. *Automatic Differentiation: Techniques and Applications. Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Germany.

[30] RASTRIGIN, L. A., 1974. "Systems of extremal control". *Mir Moscow*.