Received: 9 November 2022 | Accepted: 14 December 2022 | Published on: 4 January 2023

DOI: 10.1002/sam.11610

WILEY

RESEARCH ARTICLE

Robust deep neural network surrogate models with uncertainty quantification via adversarial training

Lixiang Zhang | **Jia Li**

Department of Statistics, The Pennsylvania State University, University Park, Pennsylvania, USA

Correspondence
Lixiang Zhang, Department of Statistics, The Pennsylvania State University, University Park, PA 16802, USA.
Email: lzz46@psu.edu

Funding information
National Science Foundation, Grant/Award Number: DMS-2013905

Abstract

Surrogate models have been used to emulate mathematical simulators of physical or biological processes for computational efficiency. High-speed simulation is crucial for conducting uncertainty quantification (UQ) when the simulation must repeat over many randomly sampled input points (aka the Monte Carlo method). A simulator can be so computationally intensive that UQ is only feasible with a surrogate model. Recently, deep neural network (DNN) surrogate models have gained popularity for their state-of-the-art emulation accuracy. However, it is well-known that DNN is prone to severe errors when input data are perturbed in particular ways, the very phenomenon which has inspired great interest in adversarial training. In the case of surrogate models, the concern is less about a deliberate attack exploiting the vulnerability of a DNN but more of the high sensitivity of its accuracy to input directions, an issue largely ignored by researchers using emulation models. In this paper, we show the severity of this issue through empirical studies and hypothesis testing. Furthermore, we adopt methods in adversarial training to enhance the robustness of DNN surrogate models. Experiments demonstrate that our approaches significantly improve the robustness of the surrogate models without compromising emulation accuracy.

KEYWORDS

adversarial training, robustness, simulator, surrogate model, uncertainty quantification

1 | INTRODUCTION

In science and engineering, many computational models with high complexity have been used to simulate physical or biological processes, perform forecasting, and help with decision-making. Examples include global climate models developed in earth system science to predict future climate [27, 30], infectious disease models, for example, stochastic HIV simulator for studying the effects of partnership concurrency on HIV transmission [1], and stochastic and high-dimensional simulators of physics systems based on stochastic partial differential equation (SPDE) [20, 32]. The accuracy of a simulator is measured by how well it reproduces empirical data, and the simulation result often comes with an assessment of uncertainty. Frequently in practice, the input to the simulator contains many parameters, which are not given precisely. The randomness in the input causes variation in the output. It is thus desirable to quantify the amount of output uncertainty, for example, by providing a prediction interval (PI), and even more comprehensively, to specify a probability density function (PDF) for the output.

The most straightforward approach to quantifying uncertainty is to perform Monte Carlo (MC) simulations

[24]. Specifically, random samples of the input parameters are drawn from the vicinity of a given configuration, and simulation is repeated at each input sample. The collection of simulation results is used for uncertainty analysis, for example, estimating the distribution of the output. The sample size needed for MC to converge can be large, while each round of simulation is often computationally costly. The compounding effect of the two factors may make the MC simulation computationally infeasible. To overcome this hurdle, researchers often adopt a surrogate model, which emulates the original simulator but generates results much faster. In the literature, several different types of surrogate models have been proposed, including Gaussian Process (GP) [5, 6], Polynomial Chaos Expansion (PCE) [25, 33, 34], and Deep Neural Network (DNN) [13, 20, 32, 38]. Among those models, DNN is attracting growing attention for several reasons. First, DNN can effectively handle the high dimensional input usually encountered in modern applications. Second, DNN often yields state-of-the-art accuracy in terms of approximating the simulators. At last, with the rapid advance of computer hardware and optimization techniques, such as Adam [18], the time to train a DNN has been reduced significantly.

It is known in image analysis and computer vision that perturbation on the data along some directions can quickly flip the classification by a DNN, even when the human eye cannot notice any change [16]. This issue has inspired research on the topic of attacks and defenses for DNN [36, 37], for example, to mention just a few, Generative Adversarial Network (GAN) [3, 11, 15] and Adversarial Autoencoder (AAE) [2, 14, 21].

Does the high sensitivity of DNN to small input changes in some directions significantly impact the uncertainty quantification (UQ) of the predictions of surrogate models? After all, inputs to surrogate models are different from images. In this paper, we answer this question by comparing density functions and by hypothesis testing. We also explore whether typical MC simulations are adequate for detecting directional sensitivity. Our study confirms the concern that the emulation error of DNN can increase sharply at a slight change in input depending on the direction of the change and to further complicate the issue, such errors are not easy to unveil through standard MC sampling. These findings call the attention of researchers using surrogate models to an overlooked yet important issue and shed light on the best practice for emulating simulators. Furthermore, we propose a computationally efficient adversarial training method and demonstrate that a DNN trained by this method achieves much better consistency in performance without losing average accuracy. Because the new method employs a revised objective function applicable to any DNN architecture, it can serve as a general framework for developing robust surrogate models.

We refer to the survey [29] for a thorough review of adversarial learning. Based on the information available to the adversary, adversarial attacks fall into three types: white-box, gray-box, and black-box attacks [29]. In white-box attacks, the adversary has full access to the target model, for example, model architecture, the training algorithm, and the gradients of the prediction function. Such attacks often achieve remarkable degradation in performance [7]. In gray-box attacks, an extra training process is evoked, during which the target model is accessible. After that training step, the adversary generates adversarial examples without querying the target model [35]. In the most challenging case of black-box attacks, without any knowledge about the target model, the adversary only uses information about the settings or past inputs to identify the vulnerability [8]. On the other hand, we can also categorize adversarial attacks into targeted versus non-targeted attacks [29]. Targeted attacks attempt to mislead the model to a particular wrong label or direction, while non-targeted attacks aim at causing incorrect prediction.

The rest of the paper is organized as follows: In Section 2, we present an approach to improving the robustness of the DNN surrogate model using adversarial learning. In Section 3, we describe the study on the sensitivity of the DNN surrogate model to perturbation directions and provide experimental results on the performance of our proposed method in terms of both prediction accuracy and UQ. Finally, we conclude in Section 4.

2 | ENHANCE DNN SURROGATE MODELS BY ADVERSARIAL TRAINING

Consider a computational model, referred to as a *simulator*, that characterizes a natural process. We generally view the model as a function $F : \mathcal{X} \to \mathcal{Y}$, where the input $x \in \mathcal{X}$ can contain a variety of quantities, for example, material properties, boundary conditions, and initial conditions. The function F is usually so complex that it requires numerical solutions for sophisticated mathematical systems, for example, PDEs. We call the calculation of $F(x)$ querying the simulator. Given the distribution of input x, in UQ, we aim at obtaining some statistics about the output $F(x)$, such as mean $E[F(x)]$ and variance $Var[F(x)]$, or a more comprehensive description, such as the PDF of $F(x)$. As discussed previously, it is computationally intensive or even infeasible to conduct MC simulations via the original simulator. A fast surrogate model, denoted by $\widehat{F}$, is used to approximate F. The performance of the surrogate model is measured by the closeness of its solution to the simulator's,

which we call *emulation accuracy*. We will study the performance of DNN as a surrogate model, in particular, the directional sensitivity of the emulation accuracy.

2.1 | Identify adversarial directions

Since our purpose is to build a robust DNN surrogate model that can avoid drastic accuracy loss with a small perturbation in the input, in the terminology of adversarial learning, the "attacks" belong to the white-box type, that is, the model is always available when generating a perturbed input. We face the case of non-targeted white-box attacks in a figurative sense. Let $\widehat{F}$ be the trained DNN prediction function and x be the input. Finding the most vulnerable perturbation that changes the prediction is equivalent to generating an adversarial example $x' = x + \delta$ by adding a perturbation δ optimized by the following problem:

$$\arg\min_{\delta} \{\|\delta\| : \widehat{F}(x+\delta) \neq \widehat{F}(x)\}. \tag{1}$$

There are several approaches to the above optimization problem, the most popular being *Fast Gradient Sign Method* (*FGSM*) [16]. FGSM calculates the gradient of the loss function J with respect to the DNN and creates adversarial examples using the following equation:

$$x' = x + \varepsilon \cdot \text{sign}\left[\nabla_x J(\theta, x, y)\right], \tag{2}$$

where ε is the magnitude of the perturbation, θ contains model parameters, and y is the true output. FGSM has several variants, for example, the Target Class Method and the Basic Iterative Method [19]. However, Cheng et al. [9] argue that the sign alone may not produce perturbations most effectively. The reason is that the sign only specifies gradients in terms of $\{0, 1, -1\}$ and a perturbation generated based on the sign can differ considerably from the gradient (the fastest-changing direction). Hence they proposed the *Fast Gradient Non-sign Method* (*FGNM*):

$$x' = x + \varepsilon \cdot \frac{\|\text{sign}\left[\nabla_x J(\theta, x, y)\right]\|}{\|\nabla_x J(\theta, x, y)\|} \cdot \nabla_x J(\theta, x, y), \tag{3}$$

where the perturbation has the same norm as FGSM but follows exactly the direction of the gradient. In our experiments, we use both FGSM and FGNM to generate adversarial samples for the DNN surrogate model and evaluate accuracy on these samples. For commonly used DNN surrogate models, we find that both methods generate adversarial samples with a strong effect on model accuracy. Details are in Section 3. Even though DNN surrogate models perform well on average with randomly drawn input samples, they are susceptible to small changes in some directions.

2.2 | Adversarial training

To enhance robustness against adversarial samples, a widely used approach is to add adversarial examples into the training data, known as *adversarial training* [31]. The addition of adversarial samples is either literal [19] or indirectly done by training with a modified loss function [16]. We adopt the second approach because it is time consuming to generate the adversarial samples—we must query the original simulator with perturbed x' to obtain the output y. In particular, we propose the following adversarial loss function:

$$\widetilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1-\alpha) J(\theta, x + \delta(x), y), \tag{4}$$

where $\delta(x)$ is the perturbation generated by the white-box adversarial attack methods FGSM or FGNM, α is the weight of the original loss (we set it to 0.8 in our experiments). The second term in the adversarial loss function imposes smoothness on the DNN. The rationale for (4) is that with a small amount of perturbation, the new output $F(x + \delta(x))$ should not move far away from the original output y, providing protection for the model under adversarial attacks. We explicitly show the dependence of $\delta(x)$ on x to emphasize that it is not a single direction. Instead, $\delta(x)$ is set along the most sensitive direction at every x. In other words, the use of $\delta(x)$ in (4) does not imply we seek robustness only along one direction given by $\delta(x)$. Also note that since $\delta(x)$ depends on the gradient of the trained DNN, it is updated iteratively as part of the optimization. Consequently, $\delta(x)$ computed from the final DNN based on the loss (4) is different from $\delta(x)$ based on the DNN without adversarial training. To evaluate the robustness of the final DNN, we will use the most sensitive perturbation directions of this DNN in addition to the most sensitive directions of the DNN without adversarial training.

3 | EXPERIMENTS

In this section, we use hypothesis testing to verify that the DNN surrogate models trained with the original loss are susceptible to changes in certain directions. Moreover, we demonstrate that the DNN surrogate models trained with the modified loss can yield higher accuracy on both the original test data and the adversarially perturbed data.

3.1 | Problem setting

We consider the benchmark elliptic partial differential equations (PDEs) on the 2-d unit square domain in Equation (5), which is called the Mindlin–Reissner (RM) plate model [23, 28]. We follow the parameter settings in Luo and Kareem [20]. Let Ω be a smooth domain, $\boldsymbol{C}$ a positive definite tensor, and ε a linear green strain tensor. Use $\boldsymbol{\theta} = \left[\theta_x, \theta_y\right]$ to record the rotations of the surface and ω to specify the transverse displacement in z-direction. Parameter γ is the scaled shear stresses; f is the applied scaled transversal load; and $\lambda = \frac{E\kappa}{2(1+\nu)}$ is the shear modulus, where E is Young's modulus, $\nu = 0.3$ is the Poisson ratio, and $\kappa = \frac{5}{6}$ is the shear correction factor. In short, Equation (5) is a system of second order PDEs that describe a clamped plate bent by a transverse force.

$$\begin{aligned} -\operatorname{div}\boldsymbol{C}\varepsilon(\boldsymbol{\theta}) - \gamma &= 0 \quad \text{in} \quad \Omega \\ -\operatorname{div}\gamma &= f \quad \text{in} \quad \Omega \\ -\lambda t^{-2}(\nabla\omega - \theta) &= \gamma \quad \text{in} \quad \Omega \\ \boldsymbol{\theta} = 0, \omega &= 0 \quad \text{in} \quad \partial\Omega \end{aligned} \tag{5}$$

Here the unknown input is the material field, that is, Young's modulus $E \in \mathbb{R}^{64\times64}$, which is modeled as a log normal random field:

$$\log E(s) \sim GP\left(m(s), k\left(s, s'\right)\right),$$

where $m(s)$ and $k\left(s, s'\right)$ are the mean and covariance functions of the GP. Based on the settings of Luo and Kareem [20], the mean function is zero and the exponential kernel is adopted as the covariance function:

$$k\left(s, s'\right) = \sigma^2 \exp\left(-\frac{\left(s - s'\right)^2}{2l^2}\right),$$

where the correlation length $l = 0.5$. In this paper, we study two datasets generated from the Mindlin–Reissner (RM) plate model, where the inputs are both Young's modulus E and the outputs are two different fields: displacement field $D \in \mathbb{R}^{64\times64}$ and stress field $S \in \mathbb{R}^{64\times64\times3}$. The displacement field dataset is denoted by $\mathbb{D}$, and the stress field dataset is $\mathbb{S}$. The PDE simulator is solved by the finite element method (FEM) [17].

For each dataset, we generate 2024 samples in total and put aside 1000 samples for testing. Following Luo and Kareem [20], for $\mathbb{D}$, we use a surrogate DNN, specifically a convolutional neural network (CNN) containing 21 layers, while for $\mathbb{S}$, we use a 25-layer CNN. The prediction task for both datasets is regression, so the *mean squared error* (MSE) is used as the loss J. L_2 penalty, Mini-batch [10], and Adam optimizer [18] are used for training.

3.2 | Performance in regression

We now evaluate the regression performance of DNN surrogate models trained respectively with the original loss and the adversarial loss on the above two datasets. Denote the DNN surrogate model trained with the original loss (including L_2 penalty) $J(\theta, x, y) + \lambda\|\theta\|_2^2$ by DNN_{ori}, while the one with the adversarial loss $\widetilde{J}(\theta, x, y) + \lambda\|\theta\|_2^2$ by DNN_{adv}. We remind that the perturbation used in $\widetilde{J}$ is generated by FGNM (Equation (3)) with $\varepsilon = 0.1$. On each dataset, the two models are evaluated on three types of test data:

1. Original test data denoted by $\mathbb{D}_{\text{test}}$ and $\mathbb{S}_{\text{test}}$.
2. Test data perturbed using the adversarial directions of DNN_{ori}, denoted by $\mathbb{DP}_{\text{ori}}$ and $\mathbb{SP}_{\text{ori}}$.
3. Test data perturbed using the adversarial directions of DNN_{adv}, denoted by $\mathbb{DP}_{\text{adv}}$ and $\mathbb{SP}_{\text{adv}}$.

The adversarial directions of any trained DNN are computed by FGNM and FGSM (Equation (2)), and the amount of perturbation is specified by ε. Although we only present results at $\varepsilon = 0.1$, we also experimented with different values of ε, for example, 0.01 and 1, and obtained similar results. For any perturbed input point, we compute the corresponding "ground-truth" output using the PDE simulator (Equation (5)). Table 1 shows the regression results in terms of MSE. We can see that the accuracy of DNN_{ori} is highly sensitive to the perturbation direction. When we use the adversarial direction computed by FGNM for both datasets, compared with the non-perturbed data, the MSE of DNN_{ori} increases by nearly one order of magnitude or more. When FGSM is used to generate the adversarial directions, the increase

TABLE 1 Regression performance in MSE achieved by DNN_{ori} and DNN_{adv} on datasets $\mathbb{D}$ and $\mathbb{S}$

	$\mathbb{D}_{\text{test}}$	FGNM-$\mathbb{DP}_{\text{ori}}$ (FGSM-$\mathbb{DP}_{\text{ori}}$)	FGNM-$\mathbb{DP}_{\text{adv}}$ (FGSM-$\mathbb{DP}_{\text{adv}}$)
DNN_{ori}	112.125	1806.347 (1012.510)	104.831 (112.322)
DNN_{adv}	110.107	101.083 (101.404)	115.275 (127.438)
	$\mathbb{S}_{\text{test}}$	FGNM-$\mathbb{SP}_{\text{ori}}$ (FGSM-$\mathbb{SP}_{\text{ori}}$)	FGNM-$\mathbb{SP}_{\text{adv}}$ (FGSM-$\mathbb{SP}_{\text{adv}}$)
DNN_{ori}	855.567	7812.958 (4372.825)	844.360 (899.756)
DNN_{adv}	728.513	835.686 (825.421)	803.132 (855.130)

in MSE is not as dramatic as with FGNM, but still more than five times that of the original data. However, if the perturbation is not targeted at the adversarial directions of DNN_{ori}, specifically, if the adversarial directions of DNN_{adv} are used instead, the accuracy of DNN_{ori} is close to that obtained on the original data. In contrast, the accuracy of DNN_{adv} is much more stable across the original data, data perturbed in its adversarial directions, or the adversarial directions of DNN_{ori}. For both datasets, the worst MSE of DNN_{adv} occurs with perturbation in the adversarial direction computed by FGSM, but the increase in MSE compared with that of the original data is below 18%. It is also interesting that the MSE values achieved by DNN_{adv} for both original datasets are lower than those by DNN_{ori}, indicating that with much-enhanced robustness, the average accuracy of DNN_{adv} is not compromised but improved.

The *squared errors* (SE) on the original test data and the test data perturbed by FGNM are visualized in the heatmap plots in Figure 1. The horizontal and vertical axes of the plots correspond to the two dominant coordinates computed by linear discriminant analysis (LDA) dimension reduction [4]. To apply LDA, we divide the samples into three classes according to their SE values. The SE values are indicated by color. We see that for both datasets, compared with the performance on the original data, DNN_{ori} yields much higher SE on the data perturbed in the adversarial direction calculated from the network itself. For the data perturbed in the adversarial direction calculated

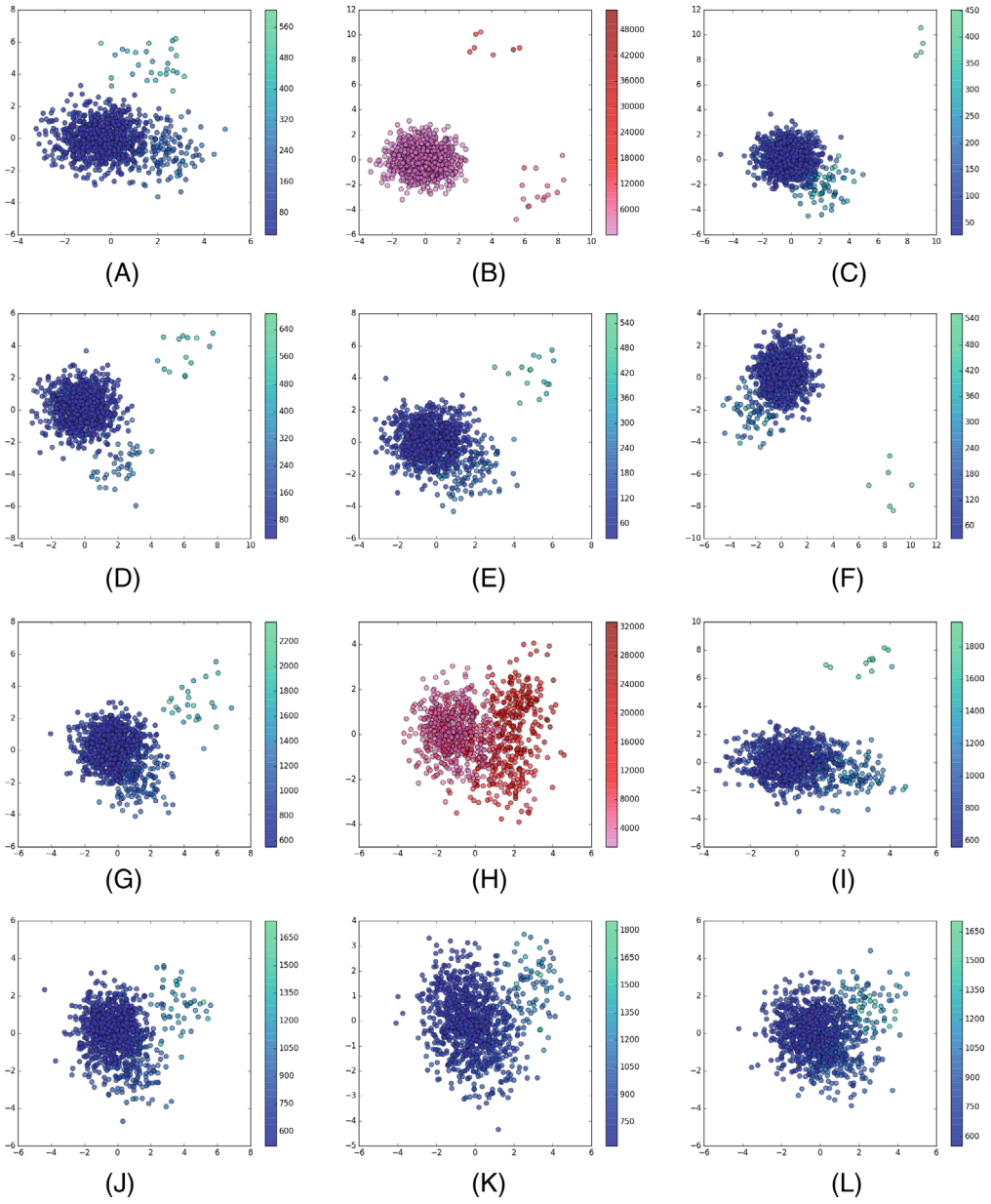

FIGURE 1 SE heatmap plots obtained by DNN_{ori} and DNN_{adv} on dataset $\mathbb{D}$ and $\mathbb{S}$. The horizontal and vertical coordinates are determined by LDA-based dimension reduction. Plots (A)-(F) are for $\mathbb{D}$ and (G)-(L) are for $\mathbb{S}$. (A) and (G): DNN_{ori} on the original test data. (B) and (H): DNN_{ori} on test data perturbed by FGNM based on DNN_{ori}. (C) and (I): DNN_{ori} on test data perturbed by FGNM based on DNN_{adv}. (D) and (J): DNN_{adv} on the original test data. (E) and (K): DNN_{adv} on test data perturbed by FGNM based on DNN_{ori}. (F) and (L): DNN_{adv} on test data perturbed by FGNM based on DNN_{adv}.

from DNN_{adv}, DNN_{ori} performs similarly as with the original data. In contrast, DNN_{adv} achieves consistent accuracy on all the test data, regardless of whether the data are perturbed or in which direction they are perturbed.

3.3 | Effects of adversarial directions

To demonstrate that the accuracy of DNN_{ori} is highly sensitive to perturbation directions but standard MC sampling can hardly detect such sensitivity, we conduct the following experiment. We randomly draw 50 sample points from the 1000 test data. For each point, we generate 100 randomly perturbed points by adding Gaussian noise with a mean of 0 and the same norm as the FGNM perturbation. The Gaussian noise ensures that the perturbation direction is uniformly random across all possible directions. For every perturbed data point, we compute the SE achieved by DNN_{ori} and then fit a 1-D density plot of $\log(SE)$ using the results for the 5000 points. Specifically, the kernel density estimation (KDE) [12, 26] is used. We obtain results at three levels of ϵ for datasets $\mathbb{D}$ and $\mathbb{S}$, respectively. The perturbed versions of the dataset $\mathbb{D}$ at $\epsilon = \epsilon_i$ (0.01,0.1,1) are denoted by $\mathbb{D}_{\epsilon=\epsilon_i}$, and similar notations are used for those of $\mathbb{S}$. Denote by $f_{rand}\left(\cdot|\mathbb{D}_{\epsilon=\epsilon_i}\right)$ the fitted density based on randomly perturbed $\mathbb{D}_{\epsilon=\epsilon_i}$. Likewise, we have $f_{rand}\left(\cdot|\mathbb{S}_{\epsilon=\epsilon_i}\right)$. For comparison, we also fit two density functions for $\log(SE)$ based on the 50 samples perturbed at the same ϵ in the adversarial directions given respectively by FGNM and FGSM. Denote these density functions by $f_{FGNM}\left(\cdot|\mathbb{D}_{\epsilon=\epsilon_i}\right)$ and $f_{FGSM}\left(\cdot|\mathbb{D}_{\epsilon=\epsilon_i}\right)$.

Figure 2 shows the fitted density functions for every $\mathbb{D}_{\epsilon=\epsilon_i}$ and $\mathbb{S}_{\epsilon=\epsilon_i}$. We see that $f_{FGNM}\left(\cdot|\mathbb{D}_{\epsilon=0.01}\right)$ and $f_{FGSM}\left(\cdot|\mathbb{D}_{\epsilon=0.01}\right)$ overlap substantially with $f_{rand}\left(\cdot|\mathbb{D}_{\epsilon=0.01}\right)$ although the average $\log(SE)$ for either of the former two densities is higher than that of the latter (similar results for $\mathbb{S}_{\epsilon=0.01}$). At larger values of ϵ, that is, 0.1 or 1, $f_{FGNM}\left(\cdot|\mathbb{D}_{\epsilon=\epsilon_i}\right)$ and $f_{FGSM}\left(\cdot|\mathbb{D}_{\epsilon=\epsilon_i}\right)$ locate mostly to the right of $f_{rand}\left(\cdot|\mathbb{D}_{\epsilon=\epsilon_i}\right)$. The average of $\log(SE)$ is significantly larger than the maximum value obtained from the samples perturbed in a random direction. This observation shows that evaluations based on the standard MC samples can grossly underestimate the error of the surrogate model in some directions. Techniques in adversarial learning are crucial for revealing this issue.

Next, we conduct hypothesis testing to investigate whether the performance of DNN_{ori} is significantly worse in the adversarial directions compared with a random direction. We generate another test dataset by adding Gaussian noise (zero mean and the same norm as the

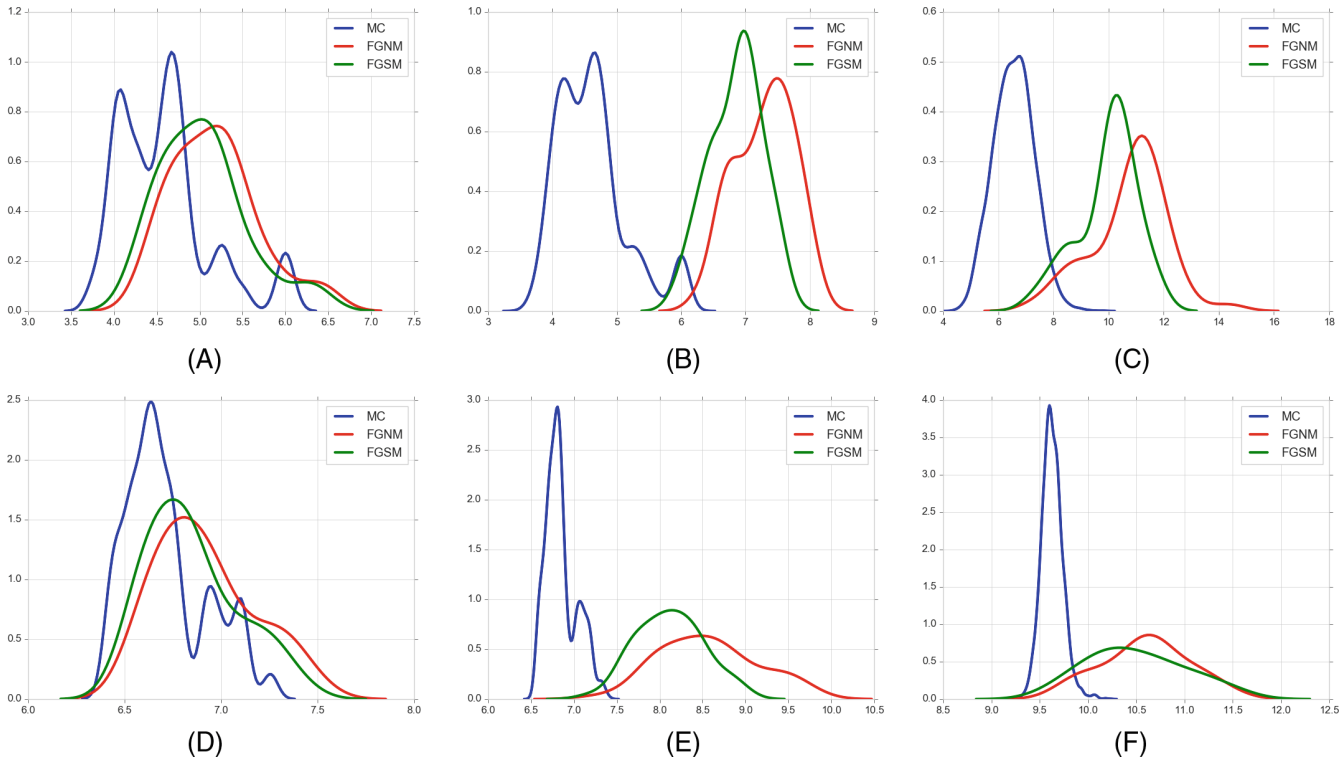

FIGURE 2 Density plots of $\log(SE)$ obtained by DNN_{ori} using MC samples generated by perturbing 50 sample points. The value of ϵ determines the amount of perturbation. In each plot, the red and green lines correspond to the fitted density functions based on the 50 random samples perturbed in the adversarial directions given by FGNM and FGSM, respectively, while the blue line corresponds to the density based on the 50 samples perturbed in 100 randomly directions. (A–C): Dataset $\mathbb{D}$, $\epsilon = 0.01,0.1,1$. (D–F): Dataset $\mathbb{S}$, $\epsilon = 0.01,0.1,1$.

TABLE 2 *p*-values of hypothesis testing on different magnitude of the perturbation.

	$\mathbb{D}_{\epsilon=0.01}$	$\mathbb{D}_{\epsilon=0.1}$	$\mathbb{D}_{\epsilon=1}$	$\mathbb{S}_{\epsilon=0.01}$	$\mathbb{S}_{\epsilon=0.1}$	$\mathbb{S}_{\epsilon=1}$
DNN_{ori}	0.000	0.000	0.000	0.000	0.000	0.000
DNN_{adv}	0.152	0.000	0.000	0.107	0.353	1.000

FGNM perturbation) to the 1000 test points. We calculate the MSE obtained by DNN_{ori} on these randomly perturbed data and compare the result with that obtained from data perturbed in the adversarial directions. We apply FGNM instead of FGSM to determine the adversarial directions because the former has induced more increase in MSE according to Table 1. Without making parametric assumptions about the distributions of MSE, we use the Mann–Whitney *U* test [22] (a non-parametric rank test) to test whether DNN_{ori} performs equally well on the randomly perturbed data and the data perturbed in adversarial directions. The alternative hypothesis is that DNN_{ori} yields a higher MSE on data perturbed in the adversarial directions. We perform the test on perturbed data generated at $\epsilon = 0.01,0.1,1$ respectively. From Table 2, we see that all the six tests on DNN_{ori} yield p-values lower than 0.001. Thus, at a significance level of 0.05, the null hypothesis of every test is rejected. We conclude that DNN_{ori} yields significantly worse MSE for data perturbed in the adversarial directions than data perturbed in random directions.

We conduct the same hypothesis tests on DNN_{adv} to see whether its accuracy changes significantly depending on the direction of perturbation. The adversarial directions are generated by FGSM instead of FGNM because the former results in a higher MSE for DNN_{adv} according to Table 1. Table 2 shows that for dataset $\mathbb{D}$, we cannot reject the null hypothesis at the significance level $\alpha = 0.05$ when $\epsilon = 0.01$. But there is a significant difference when ϵ is larger. For dataset $\mathbb{S}$, all three tests have *p*-values greater than 0.05, which indicates that the MSE achieved by DNN_{adv} has no significant difference between adversarial directions and randomly selected directions. Compared with DNN_{ori}, DNN_{adv} is more robust to perturbation in adversarial directions.

3.4 | Performance in uncertainty quantification

Depending on the application, we may be interested in different kinds of UQ. We consider two practices here. First, treating the input as random variables, we regard the output variables as random. Statistics of each output variable, often, its first and second-order moments are computed. These statistics inform us of the expected value and the expected amount of variation at the output when the input is sampled from a given distribution. We anticipate that an accurate surrogate model will produce similar UQ statistics as the original simulator. We hereby calculate the first and second-order moments of each output variable using the PDE simulator, DNN_{ori}, and DNN_{adv}. For both $\mathbb{D}$ and $\mathbb{S}$, the two surrogate models are evaluated on three types of test data (ϵ is set to 0.1):

1. Original test data denoted by $\mathbb{D}_{test}$ and $\mathbb{S}_{test}$.
2. Test data perturbed using random directions, denoted by $\mathbb{DP}_{rand}$ and $\mathbb{SP}_{rand}$.
3. Test data perturbed using the adversarial directions of the corresponding model, denoted by $\mathbb{DP}_{fg}$ and $\mathbb{SP}_{fg}$. Specifically, for DNN_{ori}, FGNM is used to decide the adversarial direction, and for DNN_{adv}, FGSM is used. For clarity of the presentation, we only report results obtained by FGNM perturbation for DNN_{ori} because the experiments show that the drop in accuracy by FGSM perturbation is less severe. For DNN_{adv}, the opposite is true—FGSM perturbation causes more degradation in accuracy. Since we are examining the robustness of these networks, we present the results for the case of the stronger adversarial effect.

As the output for the two datasets is either a 2D or 3D array, we obtain an array of the first- or second-order moments for any model. Denote the array of a moment

TABLE 3 Uncertainty Quantification by the first and second-order moment of every output variable. Relative errors (REs) are listed to measure the difference between a surrogate model and the simulator.

	$\mathbb{D}_{test}$ **1st Moment (2nd Moment)**	$\mathbb{DP}_{rand}$ **1st Moment (2nd Moment)**	$\mathbb{DP}_{fg}$ **1st Moment (2nd Moment)**
DNN_{ori}	0.001 (0.001)	0.001 (0.001)	0.004 (0.007)
DNN_{adv}	0.000 (0.001)	0.000 (0.001)	0.000 (0.001)
	$\mathbb{S}_{test}$ **1st Moment (2nd Moment)**	$\mathbb{SP}_{rand}$ **1st Moment (2nd Moment)**	$\mathbb{SP}_{fg}$ **1st Moment (2nd Moment)**
DNN_{ori}	0.001 (0.001)	0.001 (0.001)	0.009 (0.009)
DNN_{adv}	0.001 (0.001)	0.001 (0.001)	0.001 (0.001)

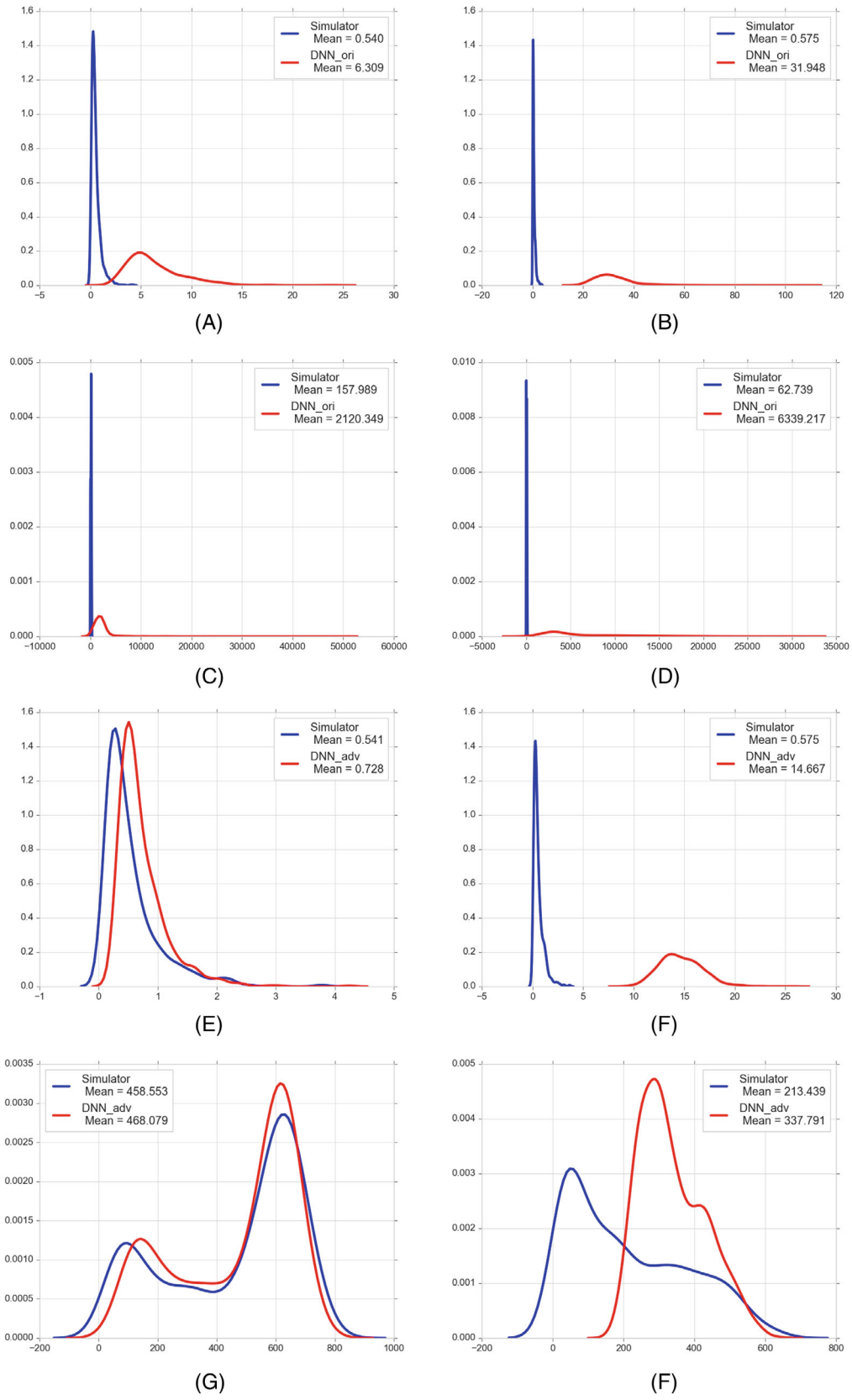

FIGURE 3 Density plots of *SE* between the outputs of the original data and perturbed data obtained from two datasets $\mathbb{D}$ and $\mathbb{S}$. In each plot, the blue line shows the density of *SE* when the simulator is used, while the red line shows the density when DNN_{ori} or DNN_{adv} is used. The mean SE of each distribution is given in the plots. These plots are obtained by different choices of datasets, ways of perturbing the input, and the surrogate model used. Plots in the left column (A, C, E, G) are for $\mathbb{D}$ and those in the right column (B, D, F, H) are for $\mathbb{S}$. (A) and (B): Data perturbed in random directions, comparison between the simulator and DNN_{ori}. (C) and (D): Data perturbed in the adversarial directions, comparison between the simulator and DNN_{ori}. (E) and (F): Data perturbed in random directions, comparison between the simulator and DNN_{adv}. (G) and (H): Data perturbed in the adversarial directions, comparison between the simulator and DNN_{adv}.

obtained by the simulator by m_{sim} and that by a DNN surrogate model by m_{DNN}. The overall disparity in the first- or second-order moment between the surrogate model and the simulator is measured by the *relative error* (*RE*):

$$\mathcal{E} = \|m_{\text{DNN}} - m_{\text{sim}}\|_F / \|m_{\text{sim}}\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Results are provided in Table 3. We see that DNN_{adv} outperforms DNN_{ori}, as shown by smaller RE values across all datasets. Moreover, for data perturbed in the adversarial directions, the RE obtained by DNN_{ori} increases sharply while that by DNN_{adv} stays the same as with randomly perturbed data.

Secondly, we consider the uncertainty caused by imprecise input. A natural question is how much the output would change when the input deviates from the truth to a certain extent. To estimate the variation, outputs at perturbed inputs are computed by the simulator. The SE between the outputs with or without input perturbation is then calculated. We can also fit a density for the SE values based on many perturbed inputs. For the sake of UQ, we would like to have a surrogate model that generates similar distributions of SE. For both datasets $\mathbb{D}$ and $\mathbb{S}$, we evaluate DNN_{ori} and DNN_{adv} by calculating SE values in two cases (ϵ is set to 0.1):

1. Between the outputs of any original input and a randomly perturbed input.
2. Between the outputs of any original input and its perturbation in the adversarial direction.

Figure 3 shows comparisons of the distributions of SE between each surrogate model and the simulator. We see that regardless of whether the direction of perturbation is random or adversarial, the distributions of SE differ significantly between DNN_{ori} and the simulator, while DNN_{adv} has yielded more similar density functions as the simulator. For DNN_{ori} versus the simulator, even the support ranges of the distributions barely overlap. For $\mathbb{D}$, DNN_{adv} yields distributions similar in both shape and the range of support to those by the simulator. In summary, for both tasks of UQ, DNN_{adv} outperforms DNN_{ori}.

4 | CONCLUSIONS

In this paper, we raise awareness of a significant drawback of DNN surrogate models—the emulation accuracy can drop significantly when the input is perturbed slightly in a direction determined by the gradient of the network. This trait of DNNs has motivated active research on adversarial training but has not attracted due attention from researchers using surrogate models. We demonstrate the severity of this problem using hypothesis testing. We also show that this problem is not easily detectable based on standard MC sampling. By exploiting techniques in adversarial training, we have developed an approach to improve the robustness of DNN surrogate models. Experiments show that the DNNs trained by the new method perform substantially better with adversarial samples in terms of both emulation accuracy and UQ. Furthermore, the new method achieves slightly better average emulation accuracy.

ACKNOWLEDGMENTS

The research is supported by the National Science Foundation under grant DMS-2013905.

REFERENCES

1. I. Andrianakis, I. R. Vernon, N. McCreesh, T. J. McKinley, J. E. Oakley, R. N. Nsubuga, M. Goldstein, and R. G. White, *Bayesian history matching of complex infectious disease models using emulation: A tutorial and a case study on hiv in Uganda*, PLoS Comput. Biol. 11 (2015), no. 1, e1003968.
2. A. Creswell and A. A. Bharath, *Denoising adversarial autoencoders*, IEEE Trans. Neural Netw. Learn. Syst. 30 (2018a), no. 4, 968–984.
3. A. Creswell and A. A. Bharath, *Inverting the generator of a generative adversarial network*, IEEE Trans. Neural Netw. Learn. Syst. 30 (2018b), no. 7, 1967–1974.
4. S. Balakrishnama and A. Ganapathiraju, *Linear discriminant analysis—A brief tutorial*, Institut. Signal Inf. Process. 18 (1998), no. 1998, 1–8.
5. I. Bilionis and N. Zabaras, *Multi-output local gaussian process regression: Applications to uncertainty quantification*, J. Comput. Phys. 231 (2012), no. 17, 5718–5746.
6. I. Bilionis, N. Zabaras, B. A. Konomi, and G. Lin, *Multi-output separable gaussian process: Towards an efficient, fully bayesian paradigm for uncertainty quantification*, J. Comput. Phys. 241 (2013), 212–239.
7. N. Carlini and D. Wagner, "*Towards evaluating the robustness of neural networks*," *2017 IEEE symposium on security and privacy*, IEEE, Los Alamitos, CA, 2017, pp. 39–57.
8. A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, D. Mukhopadhyay. Adversarial attacks and defences: A survey. *ArXiv Preprint ArXiv:1810.00069*, 2018.
9. Y. Cheng, X. Zhu, Q. Zhang, L. Gao, J. Song. Fast gradient non-sign methods. *ArXiv Preprint ArXiv:2110.12734*, 2021.
10. A. Cotter, O. Shamir, N. Srebro, and K. Sridharan, "*Better mini-batch algorithms via accelerated gradient methods*," *Advances in neural information processing systems*, Curran Associates Inc., Red Hook, NY, 2011, pp. 1647–1655.
11. A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, *Generative adversarial networks: An overview*, IEEE Signal Process. Mag. 35 (2018), no. 1, 53–65.
12. R. A. Davis, K.-S. Lii, and D. N. Politis, "*Remarks on some nonparametric estimates of a density function*," *Selected works of Murray Rosenblatt*, Springer, New York, 2011, pp. 95–100.
13. H. M. Dipu Kabir, A. Khosravi, M. A. Hosen, and S. Nahavandi, *Neural network-based uncertainty quantification: A survey of methodologies and applications.* IEEE, Access 6 (2018), 36218–36234.
14. P. Ge, C.-X. Ren, D.-Q. Dai, J. Feng, and S. Yan, *Dual adversarial autoencoders for clustering*, IEEE Trans. Neural Netw. Learn. Syst. 31 (2019), no. 4, 1417–1424.
15. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio,

Generative adversarial nets, Adv. Neural Inf. Proces. Syst. 27 (2014), 2672–2680.
16. I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *ArXiv Preprint ArXiv:1412.6572*, 2014.
17. T. J. R. Hughes, *The finite element method: Linear static and dynamic finite element analysis*, Courier Corporation, North Chelmsford, MA, 2012.
18. D. P. Kingma, J. Ba. Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*, 2014.
19. A. Kurakin, I. Goodfellow, S. Bengio. Adversarial machine learning at scale. *ArXiv Preprint ArXiv:1611.01236*, 2016.
20. X. Luo and A. Kareem, *Deep convolutional neural networks for uncertainty propagation in random fields*, Comput. Aided Civil Infrastruct. Eng. 34 (2019), no. 12, 1043–1054.
21. A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey. Adversarial autoencoders. *ArXiv Preprint ArXiv:1511.05644*, 2015.
22. H. B. Mann and D. R. Whitney, *On a test of whether one of two random variables is stochastically larger than the other*, Ann. Math. Stat. 18 (1947), 50–60.
23. RD Mindlin. *Influence of rotatory inertia and shear on flexural motions of isotropic, elastic plates*, American Society of Mechanical Engineers (ASME), New York, 1951.
24. C. Z. Mooney, *Monte Carlo simulation*. Number 116, Sage, Newbury Park, CA, 1997.
25. H. N. Najm, *Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics*, Annu. Rev. Fluid Mech. 41 (2009), 35–52.
26. E. Parzen, *On estimation of a probability density function and mode*, Ann. Math. Stat. 33 (1962), no. 3, 1065–1076.
27. Y. Qian, C. Jackson, F. Giorgi, B. Booth, Q. Duan, C. Forest, Z. Dave Higdon, J. Hou, and G. Huerta, *Uncertainty quantification in climate modeling and projection*, Bull. Am. Meteorol. Soc. 97 (2016), no. 5, 821–824.
28. Eric Reissner. *The effect of transverse shear deformation on the bending of elastic plates*, American Society of Mechanical Engineers (ASME), New York, 1945.
29. K. Ren, T. Zheng, Z. Qin, and X. Liu, *Adversarial attacks and defenses in deep learning*, Engineering 6 (2020), no. 3, 346–360.
30. M. A. Sunyer, H. Madsen, D. Rosbjerg, and K. Arnbjerg-Nielsen, *A bayesian approach for uncertainty quantification of extreme precipitation projections including climate model interdependency and nonstationary bias*, J. Clim. 27 (2014), no. 18, 7113–7132.
31. F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, P. McDaniel. Ensemble adversarial training: Attacks and defenses. *ArXiv Preprint ArXiv:1705.07204*, 2017.
32. R. K. Tripathy and I. Bilionis, *Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification*, J. Comput. Phys. 375 (2018), 565–588.
33. D. Xiu and G. E. Karniadakis, *Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos*, Comput. Methods Appl. Mech. Eng. 191 (2002), no. 43, 4927–4948.
34. D. Xiu and G. E. Karniadakis, *Modeling uncertainty in flow simulations via generalized polynomial chaos*, J. Comput. Phys. 187 (2003), no. 1, 137–167.
35. Y. Xu, X. Zhong, A. J. Yepes, and J. H. Lau. Grey-box adversarial attack and defence for sentiment classification. *ArXiv Preprint ArXiv:2103.11576*, 2021.
36. X. Yuan, P. He, Q. Zhu, and X. Li, *Adversarial examples: Attacks and defenses for deep learning*, IEEE Trans. Neural Netw. Learn. Syst. 30 (2019), no. 9, 2805–2824.
37. J. Zhang and C. Li, *Adversarial examples: Opportunities and challenges*, IEEE Trans. Neural Netw. Learn. Syst. 31 (2019), no. 7, 2578–2593.
38. Y. Zhu and N. Zabaras, *Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification*, J. Comput. Phys. 366 (2018), 415–447.

How to cite this article: L. Zhang, and J. Li, *Robust deep neural network surrogate models with uncertainty quantification via adversarial training*, Stat. Anal. Data Min.: ASA Data Sci. J. **16** (2023), 295–304. https://doi.org/10.1002/sam.11610