Bilateral Sensitivity Analysis for Understandable Neural Networks and its application to Reservoir Engineering

Jian Wang, Huaqing Zhang, Kai Zhang, and Nikhil R. Pal, *Fellow, IEEE*

***Abstract*—In this paper, a model-independent sensitivity analysis for (deep) neural network, *Bilateral Sensitivity Analysis (BiSA)*, is proposed to measure the relationship between neurons and layers. Both the *BiSA* between pair of layers and the *BiSA* between any pair neurons in different layers are defined for (deep) neural networks. This sensitivity can measure the influence or contribution from any layer to another layer behind this layer in the (deep) neural networks. It provides a helpful tool to interpret the learned model. The *BiSA* can also measure the influence or contribution from any neuron to another neuron in a subsequent layer and is critical to analyze the relationship between neurons in different layers. Then the *BiSA* from any input to any output of a network is easily defined to assess the connections between the inputs and outputs. The proposed *BiSA* of (deep) neural networks is then applied to characterize the well connectivity in reservoir engineering. Given a network trained by Water Injection Rates (WIRs) and Liquid Production Rates (LPRs) data, the well connectivity can be efficiently described through *BiSA*. The empirical results verify the effectiveness of the proposed method. The comparisons with the exiting methods demonstrate the robustness and the superior performance of the proposed method.**

***Index Terms*—Sensitivity analysis, bilateral sensitivity, neural network, understandable, interpetable, well connectivity.**

I. Introduction

(Deep) neural networks have shown powerful capabilities and effectiveness in realizing intelligent systems [1]–[3]. Despite this, the "black-box" nature still limits its applications and becomes the main reason of the refusal by many application areas [4]. In reservoir engineering, for instance, the researchers question the reliability of neural networks in real oil field handling, which has a direct effect on the cost of production and productivity. Researchers need machine learning models that they can understand and interpret. Besides, they are eager for meaningful interactions with the models. Recently, several attempts have been made to develop understandable machine learning models that can explain the insights of the models to improve model transparency and obtain thorough insights into what is learned by the models [5]. This issue has been addressed by various means such as extracting symbolic rules [6], [7], visualization [8], and quantitative analysis [19]. However, explaining neural networks, especially deep neural networks, is a hard issue, even simple properties of the models are difficult to describe using strictly theoretical inference. Hence, more efforts are needed to clarify the internal mechanism of the mighty (deep) neural networks.

Sensitivity analysis (SA) is a process that determines the impact of an input variable on the output of a model [9], [10]. The most widely adopted definition of SA is the one introduced by Saltelli *et al.* [11] as follows: the study of how uncertainty in the output of a model can be apportioned to different sources of uncertainty in the model input. Sensitivity in a learning system refers to the change in the output of the network corresponding to a small change in the parameter(s) under study. Sensitivity analysis, as a fundamental part of complex system analysis, has been proven to be an efficient tool to interpret the awareness of neural networks [12]–[14]. It opens the "black-box" by investigating the first-hand impacts of the parameter(s) on the outputs for classification or regression tasks, and helps to understand the learned relationship between the input parameters and the output variables. SA has been used to analyze the adaptive linear element (Adaline) network as early as 1962 [15]. After that, a series of works have been developed to discover the characteristics of sensitivity for various neural networks, such as Adaline-based networks [16], multi-layer perceptron (MLP) neural networks [17], [18], and deep neural networks [19]. In [20], the sensitivity analysis of ANNs is described as a partial derivative of the output to the input of the network. Some other literatures focus on the variance of the output error under certain assumptions [17], [18]. Most works have implemented sensitivity analysis either on input or weight perturbations for a single neuron taken from an MLP network. In [17], sensitivity analysis is performed on the output layer's error, sequentially computing the sensitivity neuron by neuron from the first to the last layer.

The neural network sensitivity analysis methods are developed in two major streams: *partial derivative SA (PD-SA)* and *stochastic SA (ST-SA)*. The PD-SA method [20]–[26] primarily quantifies the significance of the input parameters to the model output based on the differentiation of the input

This work was supported in part by the National Natural Science Foundation of China (No. 51722406, 61573018, 51874335 and 61305075), the Natural Science Foundation of Shandong Province (No. ZR2015AL014, ZR201709220208, JQ201808) and the Fundamental Research Funds for the Central Universities (No. 15CX08011A, 18CX02036A, 18CX02097A), the Major Scientific and Technological Projects of CNPC under Grant (No. ZD2019-183-008), the National Science and Technology Major Project of China under Grant (No. 2016ZX05025001-006). (Corresponding authors: Jian Wang, Kai Zhang)

J. Wang is with the College of Science, China University of Petroleum, Qingdao, 266580, China (email:wangjiannl@upc.edu.cn).

H. Zhang is with the College of Control Science and Engineering, China University of Petroleum, Qingdao, 266580, China (email:zhhq@upc.edu.cn).

K. Zhang is with the Department of Reservoir Engineering, China University of Petroleum, Qingdao, Shandong 266580, China (e-mail:zhangkai@upc.edu.cn).

N. R. Pal is with the Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta 700108, India (e-mail: nikhil@isical.ac.in).

variables from the output variables. A standard approach in sensitivity analysis is to compute the partial derivative of outputs with respect to its input vector via the chain rule of the derivatives, computing one layer at a time (Hashem [20] and Fu and Chen [21]). On the other hand, the PD-SA computes the sensitivity of the MLP neural network output according to input perturbation which approaches zero [23], [27]. It requires an assumption that the input perturbations are random variables with the maximum absolute value less than or equal to a calculated Q-value. This is not applicable to common models owing to the difficulty of computing the Q-value automatically. In [23], a PD-SA is applied to quantify the relevance of input and hidden neurons of feedforward neural networks. A variance-based pruning heuristic is proposed to determine which neurons to remove. The partial derivative sensitivity analysis computes the changes in the network output with respect to perturbations of the parameters.

The PD-SA is often done based on the approximation of the changes in the outputs with respect to the input or weight perturbations. For the m-th input of the j-th sample with perturbations, the Taylor expansion of the output changes is as follows [15]:

$$\begin{aligned}&\hat{f}_w\left(X^{(j)}+\triangle X_m^{(j)}\right)-\hat{f}_w\left(X^{(j)}\right)\\&=\hat{f}'_w\left(X^{(j)}\right)\triangle x_m+\frac{1}{2}\hat{f}''_w\left(X^{(j)}\right)(\triangle x_m)^2+\cdots\\&\approx\hat{f}'_w\left(X^{(j)}\right)\triangle x_m,\end{aligned}\tag{1}$$

where $\triangle X_m=(0,\cdots,\triangle x_m,\cdots,0)$, $\triangle x_m>0$ is a very small fixed number for each feature. Then the PD-SA is defined as the first order derivative, $\hat{f}'_w\left(X^{(j)}\right)$, based on the assumption that the part $\frac{1}{2}\hat{f}''_w\left(X^{(j)}\right)(\triangle x_m)^2+\cdots$ is very small. PD-SA is sometimes represented as $\hat{f}''_w\left(X^{(j)}\right)$ if $\hat{f}'_w\left(X^{(j)}\right)\approx 0$. In [10], the PD-SA for radial basis function neural network (RBFNN) with regard to the m-th input of the j-th sample is defined as

$$\begin{aligned}SA_m^j&=\hat{f}'_w\left(X^{(j)}\right)=\frac{\partial\hat{f}_w\left(X^{(j)}\right)}{\partial x_m}\\&=\sum_{i=1}^{Q}\left[w_i\mathbf{exp}\left(\frac{\|X^{(j)}-U_i\|}{-2v_i^2}\right)\frac{x_m^{(j)}-u_{im}}{-v_i^2}\right],\end{aligned}\tag{2}$$

where $X^{(j)}$ represents the j-th sample, x_m is the m-th input, w_i is the weight connecting the i-th hidden node, U_i and v_i are the center and width of the i-th basis function, respectively. This kind of method has two main weaknesses. First, it cannot deal with networks with non-differentiable activation functions, and second, it does not consider the magnitude of the parameters.

The stochastic sensitivity analysis (ST-SA) uses the magnitudes of the output perturbations between the original samples and perturbed samples, in a statistical sense. Because of its high computational efficiency, and as long as the function expression between the output variable and the input parameter is given, the sensitivity of the output variable in different input parameters can be effectively analyzed. So it is widely used in sensitivity analysis. The ST-SA for neural network is defined by the magnitudes of the change in the output with respect to weight or input perturbations. Compared to PD-SA, where the rate of output change is used to approximate the sensitivity corresponding to parameter perturbations, the ST-SA possesses several advantages. First, the ST-SA does not evaluate each training sample one by one and constrains less on the used perturbations. It measures the differences between the original outputs and the perturbed outputs based on the analytical formulas for the neural network. Besides, ST-SA is more interpretable, and it is more meaningful for feature or instance selection. Finally, ST-SA is a straightforward manner reflects the generalization error because of the usage of magnitudes of output error produced by the parameter perturbations, which is a direct reflection of the generalization. Hence, the sensitivity analysis for neural networks in this paper is defined in a stochastic manner.

In this paper, we first propose a *Bilateral Sensitivity Analysis (BiSA)* scheme for the (deep) neural networks, then apply it to an important issue of characterizing the well connectivity in reservoir engineering.

The main **contributions** of this paper are:

(i) The model-independent *BiSA* between any pair of layers in a neural network is defined;

(ii) The model-independent *BiSA* between any pair of neurons where the members of the pair are from different layers in a neural network is defined;

(iii) The *BiSA* from an input to the output of a neural network is given;

(iv) The proposed *BiSA* of neural networks is applied to characterize the well connectivity in reservoir engineering.

The remainder of this paper is organized as follows. Some related works on stochastic sensitivity analysis and inferring well connectivity are investigated in Section II. In Section III, the *Bilateral Sensitivity* is proposed for both between a pair of laye rs and between as pair of neurons from different layers. The reservoir connectivity is characterized using the *Bilateral Sensitivity* in Section IV. Some useful conclusions of this work are given in Section V.

II. Related works

In this section, first, a brief survey of stochastic sensitivity of neural networks is investigated to make clear the diverse definitions. Then some basic knowledge of well connectivity in reservoir engineering is given to make readers ready for a specific application of the proposed *BiSA* method.

A. The stochastic sensitivity analysis of neural networks

To the best of our knowledge, the sentivity of neural networks was first discussed by Hoff in [28] to analyze the output changes of Adalines generated by weight perturbations and later it was extended in [15], [16], [30]. Stevenson *et al.* [16] studied the sensitivity of multi-layered networks (Madaline) to weight errors, input errors, and the combination of input and weight errors. For instance, the probability of decision error was expressed in term of the ratio for weight error. Similarly the probability of a decision error was approximated by a

simple expression involving the weight and input perturbation ratios due to the combined effect of weight and input perturbations.

In [16], [18] and [29], the statistical sensitivity to weight perturbations $\triangle W$ is defined as

$$S^j(W) := \lim_{\sigma \to 0} \frac{\sqrt{\mathrm{var}[\triangle y]}}{\sigma}, \tag{3}$$

where $\triangle y$ is the output changes due to weight perturbations, σ is the standard deviation of each component of $\triangle W$, and $\mathrm{var}[\cdot]$ is the variance of $[\cdot]$. Piché [30] introduced the direct statistical analysis to determine the sensitivity of a neural network to perturbations of weights. It provided understandable consequence of neurons by the variance of the output error for Madalines according to weight perturbations. This is a fundamental work for ST-SA, where the output error of the n-th Adaline node in layer l caused by the input and weight perturbations is defined as

$$\triangle y_l^n = f\left(\left(X_l + \triangle X_l\right)^T \left(W_l + \triangle W_l^n\right)\right) - f\left(X_l^T W_l^n\right), \tag{4}$$

where $\triangle X_l$ is the perturbation on the input to the l-th layer, X_l, and $\triangle W_l^p$ is the perturbations to the weight W_l associated with the n-th Adaline node in layer l. Then the sensitivity of the l-th layer output to perturbations in the weights is represented by the noise-to-signal ratio (NSR), which is defined as the ratio of the variance of the l-th layer's output error to the variance of the output of the l-th layer:

$$NSR_l = \frac{\sigma^2_{\triangle y_l}}{\sigma^2_{y_l}}. \tag{5}$$

In [31], Alippi *et al.* introduced a sensitivity to errors in neural networks using probability as follows:

$$p_l\left(\delta y_l\right) = \sum_{\delta X_l} p_{\delta y_l} | \delta X_l \cdot p(\delta X_l), \tag{6}$$

where $p_{\delta y_l}|\delta X_l$ is the conditional probability that a relative output error δy_l is generated whenever the input relative error is δX_l.

One important definition of stochastic sensitivity is proposed by Sobol [36] considering the variance with respect to the parameters and the total variance of the output. Sobol defined the global sensitivity indices as as follows:

$$S_{f_I}(x_I) = \frac{V(f_I)}{V(f)} = \frac{V(f_I)}{\sum_I V(f_I)}, \tag{7}$$

where $V(f_I)$ is the variance corresponding to the indices I, $V(f)$ is the sum of all the first-order and higher-order terms added up to the total unconditional variance. Saltelli *et al.* [11], [32], [33] improved the estimation procedure for computing variances based on the Fourier amplitude sensitivity test (FAST), where the sensitivity is computed through

$$S_h = \frac{\mathrm{Var}_{X_h}[E(Y|X_h)]}{\mathrm{Var}(Y)}, \tag{8}$$

where $E(Y|X_h)$ represents the expectation of Y conditional on $X = x_h$, Var_{X_h} denotes , Y is the output, and x_h is the h-th input factor. Fock [34] did further improvements on the extended FAST method. Furthermore, a global sensitivity analysis based on the ANOVA decomposition was introduced in [35] as an alternative way of measuring the sensitivity indices of Sobol decomposition [36] for neural network classifiers. However, two conditions, limited variable range and square-integrable classification function, are required to be imposed on the classification function.

Stevenson *et al.* [16] used the probability of decision error to describe sensitivity but the magnitude of the error was not considered. This was enhanced by Cheng and Yeung [37] in neocognitron model but still could be directly used in MLP. With antisymmetric squashing activation function, Yeung *et al.* [38] extended Piche's method [30] and removed the independently identically distributed (i.i.d) restriction on the input and output errors. In [17], sensitivity is defined as the expectation of the output errors with respect to the input and weight perturbations in a continuous interval. Both the sensitivity for a single neuron and of the entire multi-layer perceptron network are discussed. However, the computation is highly complex, especially for the tasks with high dimensional data. Shi *et al.* [39] generalized this method to radial basis function (RBF) networks, later Yeung *et al.* [40] introduced a localized generalization error model (L-GEM) within a so-called Q-neighborhood of the training samples. However, this method does not calculate the sensitivity for individual instances. In [41] another Q-neighborhood based sensitivity was introduced to assess sensitivity for individual instances of the imbalanced classification problem. Recently, a stochastic sensitivity [42] based on L-GEM was introduced to provide a straightforward measure on an MLP's output fluctuations.

Ng *et al.* [43] utilized a localized generalization error model to create a novel hybrid filter-wrapper-type feature selection methodology. This solution gives the possibility of removing a large percentage of features causing a statistically insignificant loss in the accuracy. The work presented in [44] defined the stochastic sensitivity as the expected value of square of the changes in the classifier output with respect to feature perturbations. This, in turn, allows the authors to crate a radial basis function neural network that can be trained by minimizing the defined sensitivity.

The expectation based methods are also combined with the variance to achieve a better performance. For the j-th instance $X^{(j)}$ with the perturbation on the m-th feature, $\triangle X_m^{(j)} = \left(0, \cdots, \triangle x_m^{(j)}, \cdots, 0\right)$, $\triangle x_m^{(j)} \in \mathrm{R}$, the sensitivity of this feature is given based on the difference of the network output

$$\triangle y^{(j)} = \hat{f}_{(W+\triangle W)}\left(X^{(j)} + \triangle X_m^{(j)}\right) - \hat{f}_W\left(X^{(j)}\right), \tag{9}$$

where $\triangle y^{(j)}$ is the output changes for the j-th sample, $\triangle W$ is the weight perturbation on weight W. For instance, the ST-SA for RBFNN in [15] is defined as the combination of expectation and variance:

$$SA = |E(\triangle y)| + c\sqrt{Var(\triangle y)}, \tag{10}$$

where $\triangle y$ is the vector of output changes for all samples, c is a positive constant, $E(\triangle y)$ is the expectation and $Var(\triangle y)$ is the variance of the output changes.

Recently, Karmakar *et al.* [45] exploited the statistical sensitivity in [18] as a penalty in the objective function of an MLP neural network to enable the network to say "Don't know". Xiang *et al.* [46] introduced the maximum sensitivity by a bounded disturbance on the nominal input to measure the maximum deviation of outputs. Li *et al.* [47] studied the deviation of functions represented by DNN from their typical mean field solutions by the large deviation theory and path integral analysis, where the commonly used weight sparsification and binarization in model simplification were investigated under parameter perturbations. In [48], a provable Sensitivity-informed Provable Pruning (SiPPing) method of neural networks was suggested based on a ST-SA of measuring the importance of each weight for one layer. The sensitivity of a weight was defined as the proportion of the signal delivered by this weight to the whole signal of that layer. Motivated by information geometry, Shu *et al.* [49] introduced a stochastic sensitivity analysis for DNN classifiers using a perturbation manifold. To interpret and to enhance the adversarial robustness of DNNs, a sensitivity measure of a neuron was defined by measuring the intensity of variation of neuron's behavior against benign and adversarial examples [50]. Given an example x_j, $(j = 1, 2, \cdots, J)$ and its corresponding adversarial example $x_j^{'}$, the sensitivity for the m-th neuron in the lth layer was computed as

$$S_l^m(x_j, x_j^{'}) = \frac{1}{J}\sum_{j=1}^{J} \|F_l^m(x_j) - F_l^m(x_j^{'})\|_1, \tag{11}$$

where $F_l^m(x_j)$ and $F_l^m(x_j^{'})$ are the corresponding outputs of neuron F_l^m. These sensitivity methods provide a deeper understanding of the (deep) neural networks. Nevertheless, it is necessary to study the sensitivity analysis from the perspective of a general view and provide the sensitivity analysis for both the bilateral neurons and layers. Hence, this paper focuses on contributing a model-independent bilateral sensitivity to evaluate the contribution from one neuron to another neuron or from one layer to another layer. It is easily extended to the cases from one neuron to one layer or from one layer to a neuron. This is essential to deeply understand and interpret the mechanism of the (deep) neural network.

B. Reservoir connectivity

In this paper, we will test the proposed bilateral sensitivity on the task of inferring well connectivity in reservoir engineering, which is a hard and classic inverse-problem in engineering. Well connectivity remains a vital research topic in the petroleum industry, where waterflood is commonly used as the secondary recovery technique to generate man-made oil-displacing energy. Well connectivity reveals the connectivity between the injectors and producers to guide optimizing operations and maximize oil production. Usually, it is identified by the recorded production and injection data. However, the issue is very tough owing to its non-stationary and non-linear nature [51]. Fortunately, abundance of production data such as injector and producer flow rates can be easily obtained even for marginal oil fields. These production data contain the information of well connectivity. Various methods using production data to infer connectivity have been developed. For instance, the injection and liquid production rates are used in a linear multivariate regression (MLR) model by employing capacitance model to characterize the well connectivity in [52], [53]. Wang *et al.* [54] introduced a method with signal processing techniques to represent connectivity. Recent works try to deal with this problem using machine learning techniques [55], [56], but the potential power of machine learning has not yet been exploited. The lack of interpretability of neural networks also limited the interest of reservoir engineer's to explore further the use of NNs on this task.

The sensitivity analysis has also been applied to the topic of reservoir connectivity inference. In [57], Albertoni *et al.* employed sensitivity analysis to multivariate linear regression (MLR) to infer the interwell connectivity. Demiryurek *et al.* [58] simulated the sensitivity analysis of neural network to analyze the connectivity between injector and producer in fields and similar approach was employed in [56]. However, both works use the identical SA method defined in [18], which is a partial derivative sensitivity analysis. The proposed *BiSA* in this paper is based on a stochastic manner, which is more suitable for well connectivity because the observed production data are not continuous basically.

III. Bilateral Sensitivity Analysis (BiSA) for neural networks

In this section, we define the *Bilateral Sensitivity Analysis* for (deep) MLP neural networks. In fact, it is model-independent and can be regarded as a measure for other kinds of models too. Assume that an MLP has $(L+1)$ layers, as shown in Fig. 1, and the l-th $(0 \le l \le L)$ layer possesses n_l neurons. Especially, n_0 represents the number of features in the input layer when $l = 0$ and n_L indicates the number of outputs in the output layer when $l = L$. For the i-th $(1 \le i \le n_l)$ neuron in layer l, $X^l = (x_1^l, x_2^l, \cdots, x_{n_{l-1}}^l)^T$ is the input vector, $W_i^l = (w_{i1}, w_{i2}, \cdots, w_{in_{l-1}})^T$ is the corresponding connection weights, $\theta^l = (\theta_1^l, \theta_2^l, \cdots, \theta_{n_l}^l)^T$ is the bias vector of the l-th layer, then the output of the l-th layer is

$$\mathbf{y}^l = f_l\left(W^l X^l + \theta^l\right). \tag{12}$$

Here, we define the operator ϕ as

$$\phi_l\left(X^l\right) = f_l\left(W^l X^l + \theta^l\right). \tag{13}$$

Then we have

$$\mathbf{y}^l = \phi_l\left(X^l\right). \tag{14}$$

The input to the l-th layer is exactly the output of the $(l-1)$-th layer, thus for an MLP, the mapping from its beginning input to its final output can be represented as

$$\mathbf{y}^L = \Phi_L\left(X^0\right), \tag{15}$$

where $\Phi_L(\cdot) \triangleq \phi_L \circ \phi_{L-1} \circ \cdots \circ \phi_1(\cdot)$.

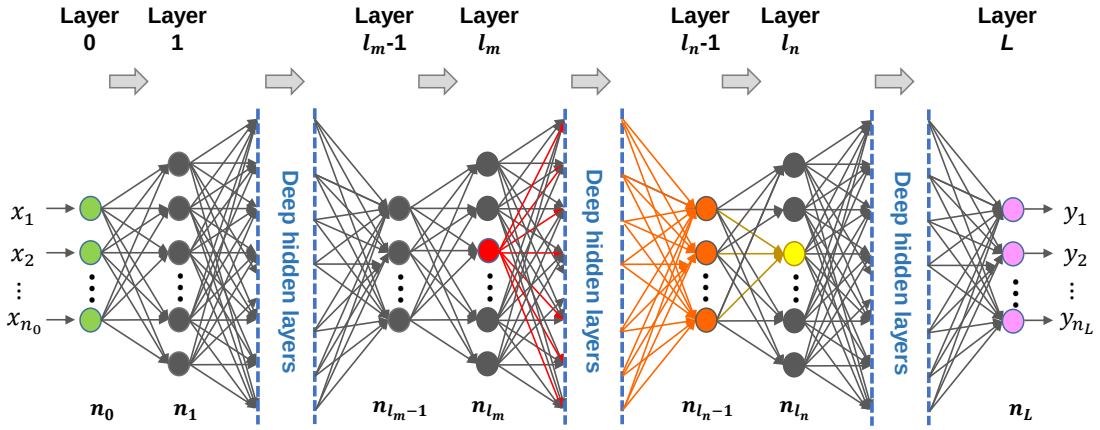

Fig. 1. The structure of a deep MLP network. Layer 0 represents the input layer and layer L represents the output layer of the MLP. The numbers below the network, n_ls $(0 \le l \le L)$, indicate the number of neurons in the corresponding layers. Assume that the red node in the l_m-th $(0 \le l_m < L)$ layer is the i-th neuron in this layer, and the yellow node in the l_n-th $(0 < l_n \le L,\ l_m < l_n)$ layer is the j-th neuron in this layer. The *Bilateral Sensitivity* of the two neurons is investigated in Section III.

A. Definition of Bilateral Sensitivity (BiS)

For the input vector of l_m $(0 \le l_m < L)$ layer, $X^{lm} \in [0,1]^{n_{l_m-1}}$, it is the output of the $(l_m - 1)$ layer, thus

$$X^{lm} = \phi_{l_m-1}\left(X^{l_m-1}\right) = \phi_{l_m-1} \circ \phi_{l_m-2} \circ \cdots \circ \phi_1\left(X^0\right), \tag{16}$$

then the output of the l_m layer is

$$\mathbf{y}^{l_m} = \phi_{l_m}\left(X^{l_m}\right) = \phi_{l_m} \circ \phi_{l_m-1} \circ \cdots \circ \phi_1\left(X^0\right). \tag{17}$$

Similarly, for the input vector of l_n-th $(0 < l_n \le L,\ l_m < l_n)$ layer, $X^{ln} \in [0,1]^{n_{l_n-1}}$ and we have

$$X^{ln} = \phi_{l_n-1}\left(X^{l_n-1}\right) = \phi_{l_n-1} \circ \phi_{l_n-2} \circ \cdots \circ \phi_1\left(X^0\right), \tag{18}$$

then the output of the l_n layer is

$$\begin{aligned}\mathbf{y}^{l_n} &= \phi_{l_n}\left(X^{l_n}\right) = \phi_{l_n} \circ \phi_{l_n-1} \circ \cdots \circ \phi_1\left(X^0\right) \\ &= \phi_{l_n} \circ \phi_{l_n-1} \circ \cdots \circ \phi_{l_m} \circ \phi_{l_m-1} \cdots \circ \phi_1\left(X^0\right) \\ &= \phi_{l_n} \circ \phi_{l_n-1} \circ \cdots \circ \phi_{1_m}\left(X^{l_m}\right).\end{aligned} \tag{19}$$

Based on these, the *Bilateral Sensitivity* of the l_m-th layer to the l_n-th layer can be defined using the perturbations on the inputs of the l_m-th layer as follows.

Definition 1. *Given the perturbations on the input vector of l_m-th layer, $\triangle X^{l_m} = \left(\triangle x_1^{l_m}, \triangle x_2^{l_m}, \cdots, \triangle x_{n_{l_m-1}}^{l_m}\right)^T$, and $\Phi_{l_m}^{l_n}(\cdot) \triangleq \phi_{l_n} \circ \phi_{l_n-1} \circ \cdots \circ \phi_{1_m}(\cdot)$, then the Bilateral Sensitivity of the l_m-th layer to the l_n-th layer $(l_m < l_n)$ is defined as*

$$BiS\left(l_m, l_n\right) = E\left(\left|\Phi_{l_m}^{l_n}\left(X^{l_m} + \triangle X^{l_m}\right) - \Phi_{l_m}^{l_n}\left(X^{l_m}\right)\right|\right). \tag{20}$$

This sensitivity can measure the influence or contribution from any layer to another later layer in a neural network including deep neural networks. It provides a helpful tool to interpret the the learned deep model. Here, the Bilateral Sensitivity $BiS\left(l_m, l_n\right)$ describes the relation between the l_m-th layer and the l_n-th layer. The j-th element in the output of the l_n-th layer is

$$\mathbf{y}_j^{l_n} = \left(\phi_{l_n}\left(X^{l_n}\right)\right)_j = f_{ln}\left(W_j^{ln} X^{ln} + \theta_j^{ln}\right), \tag{21}$$

then the *Bilateral Sensitivity* of the i-th neuron in the l_m-th layer to the j-th neuron in the l_n-th layer can be given through perturbing the i-th neuron in the l_m-th layer as follows.

Definition 2. *Given the perturbation on the i-th input of the l_m-th layer $\triangle X_i^{l_m} = \left(0, \cdots, 0, \triangle x_i^{l_m}, 0, \cdots, 0\right)^T_{1 \times n_{l_m}}$, the Bilateral Sensitivity of the i-th neuron in the l_m-th layer to the j-th neuron in the l_n-th layer is defined as*

$$\begin{aligned}BiS\left(l_m, i, l_n, j\right) = E\Bigg(\Bigg|&f_{l_n}\left(W_j^{l_n}\Phi_{l_m}^{l_n-1}\left(X^{l_m} + \triangle X_i^{l_m}\right) + \theta_j^{l_n}\right) \\ &- f_{l_n}\left(W_j^{l_n}\Phi_{l_m}^{l_n-1}\left(X^{l_m}\right)\right)\Bigg|\Bigg).\end{aligned} \tag{22}$$

This sensitivity can measure the influence or contribution from any neuron to any other neuron in any higher layer. This is critical to analyze the relationship from one neuron to another. The perturbation on the i-th neuron of the l_m-th layer, $\triangle X_i^{l_m}$, affects the whole subsequent network including the j-th neuron in the l_n layer, as shown in Fig. 1. The vector $W_j^{l_n} = \left(W^{l_n}(j,1), W^{l_n}(j,2), \cdots, W^{l_n}(j, n_{l_n-1})\right)$ is the j-th row vector of weight matrix W^{l_n}, i.e. the weight vector connecting all the neurons in the (l_n-1)-th layer to the j-th neuron in the l_n-th layer, as shown by the orange lines in Fig. 1 for example.

Subsequently, the relation between the i-th input and the j-th output for an MLP with L layers, namely the *Bilateral Sensitivity* of the i-th input in the 0-th layer to the j-th output in the L-th layer, is given by the following definition.

Definition 3. *The Bilateral Sensitivity of the i-th input to the j-th output of the MLP is described as*

$$\begin{aligned}BiS_{io}(i,j) &= BiS\left(0, i, L, j\right) \\ &= E\Bigg(\Bigg|f_L\left(W_j^L \Phi_{L-1}\left(X^0 + \triangle X_i^0\right) + \theta_j^L\right) \\ &\quad - f_L\left(W_j^L \Phi_{L-1}\left(X^0\right) + \theta_j^L\right)\Bigg|\Bigg),\end{aligned} \tag{23}$$

where $\triangle X_i^0 = \left(0, \cdots, 0, \triangle x_i^0, 0, \cdots, 0\right)_{1\times n_0}^T$ *is the perturbation on the input layer and* $\Phi_{L-1}(\cdot) \triangleq \phi_{L-1} \circ \cdots \circ \phi_1(\cdot)$.

The proposed *Bilateral Sensitivities* are calculated after the network being trained and are independent of the optimization method used to train the network. Hence, the sensitivity indices are model-independent. They are calculated easily and efficiently using the optimized weights and biases. The calculation is only a chain of l $(0 < l < L)$ operations consisting of multiplication, addition (if bias is used), and activation function. In this paper, the perturbations are generated by Gaussian noise. After training the network, the final weights capture the patterns existing in the training data. In this case, it is easy to compute the *Bilateral Sensitivity* for each pair, which reflects the connectivity between the corresponding injection-production wells through perturbing the water injection rates.

IV. Inferring well connectivity using Bilateral Sensitivity of neural networks

A. Inferring well connectivity

Here we consider an MLP with single hidden layer, as shown in Fig. 2, where no bias is used to simplify the discussion. The input vector, $I = (I_1, I_2, \cdots, I_M)^T$, consists of the injection rates from M water injection wells. The output vector, $P = (P_1, P_2, \cdots, P_N)^T$, involves the liquid production rates from N production wells. We train the network with the reservoir history data, i.e., the water injection rates (WIRs) for the water injectors as inputs and the liquid production rates (LPRs) for the producers as outputs. For a highly connected injector-producer pair, I_m $(1 \le m \le M)$ and P_n $(1 \le n \le N)$ for instance, LPR_n (the liquid production rate of P_n) is expected to vary in accordance with the alteration of WIR_m (the water injection rate of I_m). Nevertheless, the consistency tends to hardly occur for a poorly connected pair.

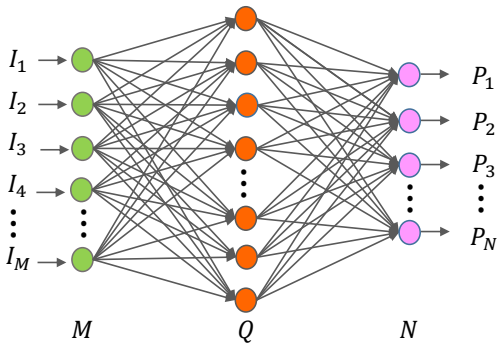

Fig. 2. The structure of an MLP with only one single layer.

Perturbation creation. After training, with the learned knowledge from the training history data, the relationship between input variables and target properties can be analyzed by the *Bilateral Sensitivity Analysis*. For the neural network with only one hidden layer in Fig. 2, the connectivity between the m-th injector and the n-th producer is defined by the *Bilateral Sensitivity* from the m-th input to the n-th output:

$$C(m,n) = E(|g\left(U_n f(V(I+\triangle I))\right) - g\left(U_n f(VI)\right)|), \quad (24)$$

where $\triangle I = (0, \cdots, 0, \triangle I_m, 0, \cdots, 0)_{M\times J}$, the perturbation $\triangle I_m = (\triangle I_{m1}, \triangle I_{m2}, \cdots, \triangle I_{mj}, \cdots, \triangle I_{mJ})_{1\times J}$. Suppose, $I_m = (i_{m1}, i_{m2}, \cdots, i_{mj}, \cdots, i_{mJ})$ is the m-th input vector for all training samples, $std(I_m)$ is its standard deviation, then the j-th element in $\triangle I_m$ is created by

$$\triangle I_{mj} \sim N(0, \gamma * std(I_m)), \quad (25)$$

where $\gamma \in [0, 1]$ represents a percentage to control the difference between standard deviations of the generated perturbation and the original input data. In this paper, we let $\gamma = 10\%$ and 20% respectively to generate noises. We adopt such a strategy because if the variance of an input is very high, then it should be tolerant to larger level of noise perturbation.

Data source. Two synthetic reservoir scenarios in petroleum engineering are used throughout this paper. One is from [55], the other is from a simulation of a complex reservoir scenario. Although both cases have relatively simple permeability, they are typical and universal examples of the real cases in practical applications. The understanding of these characteristics are quite crucial to the design of the water flooding scheme. These scenarios have been built and run by a commercial reservoir simulator, *Eclipse* 2011. These kinds of synthetic models have been proven to be useful in a number of works [55] and are widely used in reservoir engineering. The first case is defined as a *Streak Case*, which represents reservoirs with high-permeability streaks. The second one is defined as *Braided River*, which represents complex reservoirs which contain several high permeability channels. The proposed methods, INNGLP and CINNGLP, are tested on both cases with the network architecture in Fig. 2. We take the water injection rates of each injection as the network input and liquid production rates of each producer as the network output. All the reported results are the average ones obtained by 10 repeats of the experiments.

Global normalization. Normalization plays an important role in the pre-processing of data for a given task. Minmax and z-score normalization are some of the popular techniques used for relevance score normalization. For a good normalization scheme, the estimates of the location and scale parameters of the matching score distribution must be robust and efficient. Robustness refers to insensitivity to the presence of outliers. Efficiency refers to the proximity of the obtained estimate to the optimal estimate when the distribution of the data is known [59]. Here we use z-score normalization on the input data of the neural network, namely the water injection rates, while Min-Max normalization on the output data. Z-score normalization is a strategy of normalizing data that can avoid the outlier issue [60]. The z-score normalization is defined as below:

$$\tilde{X} = \frac{X - \mu}{\sigma}, \quad (26)$$

where μ is the mean value of the original data vector X and σ is the standard deviation of the vector. For a variable, if all the values are equal (i.e., variance is equal to zero) then the normaized value is set to zero. Note that a global z-score, instead of column (or feature-wise) z-score, is implemented here to maintain the difference of the inputs (water injection rates) at different sources. However, z-score normalization

is not suitable for the output data due to use of the tansig activation function at the output layer of the neural network, where the network output is limited from -1 to 1. Hence a global Min-Max normalization is used for the output data, i.e., the liquid production rates.

B. Case Study 1: Streak Case

The *Streak Case* is a public synthetic field case and its detailed description is available in [55], [61]. As shown in Fig. 3, it is a square reservoir made up of 5 vertical injectors and 4 vertical producers, represented by I1, I2, I3, I4, I5 and P1, P2, P3, P4, respectively. The permeability of the two high-permeability streaks I1-P1 (the streak between I1 and P1) and I3-P4 (the streak between I3 and P4) are set to be 1000 md and 500 md, respectively. The permeability for the rest of the reservoir is 5 md. Note that the permeability of the underground is not known in real applications. Both rock and water compressibilities are $1 * 10^{-6}\ psi^{-1}$. The oil compressibility is $5 * 10^{-6}\ psi^{-1}$. The porosity is 0.18 and the total mobility is 0.45 independent of saturation.

After being trained by the water injection rates and liquid production rates, the neural network has learned the knowledge from the input and output patterns. This means a nonlinear mapping from the inputs to the outputs, which reflects the knowledge of the reservoir, is established based on the learned weights. In this case, we have used the multiplicative Gaussian noise to perturb the inputs and obtained the *Bilateral Sensitivity* between each injector and producer pair.

Fig. 4 shows the results of the proposed *BiSA* method with addictive 10% Gaussian noise (with $\sigma = standard\ deviation\ of\ input\ data * 0.1$) perturbation on the water injection rates of the *Streak Case* scenario, where the darker the color the stronger the connectivity. This heat-map clearly demonstrates the two streaks of high connectivity in *Streak Case*, i.e. I1-P1 and I3-P4, and the quite low connectivities for the other pairs. To verify the robustness of the proposed method, other noise levels are also tested in our experiments. Fig. 5 shows the results of the proposed *BiSA* method with multiplicative 20% Gaussian noise perturbation on the water injection rates of the *Streak Case* scenario. With this stronger perturbations, the heat-map more clearly reflects the two streaks of high connectivity, I1-P1 and I3-P4, and the low connectivities for the other pairs. This demonstrates the robustness of using the multiplicative Gaussian noise as the perturbation.

To explain the mechanism explicitly, we analyze it from the training data and the perturbations. Fig. 6 shows curves of the original I1 (the water injection rates for injection 1), 10% Gaussion noised I1, and the 20% Gaussion noised I1. With these perturbations, we turn to the corresponding outputs of the learned model. Fig. 7 shows the curves of the original P1 (the liquid production rates for producer 1), the output for producer 1 according to 10% Gaussion noised I1, and the output for producer 1 with regard to the 20% Gaussion noised I1, respectively. Obvious changes can be observed for the values of this output with both 10% and 20% Gaussion noised I1. This is because the network learned the knowledge of the

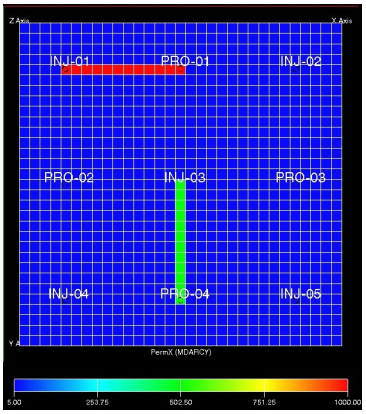

Fig. 3. Scenario 1: The permeability field of the *Streak Case* with 5 injections and 4 producers.

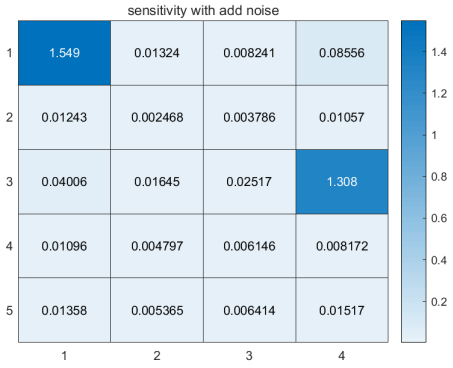

Fig. 4. The results for the *BiSA* method with 10 percent Gaussian noise perturbation on the *Streak Case* scenario.

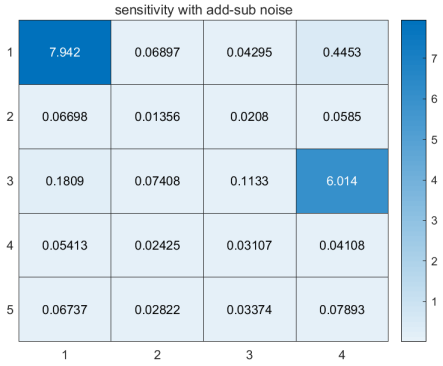

Fig. 5. The results for the *BiSA* method with 20 percent Gaussian multiplicative noise perturbation on the *Streak Case* scenario.

strongly connected streak between I1 and P1, which means the I1 affects largely P1 and the changes of I1 directly lead to the changes in P1. As a result, the computed $BiS(1,1,3,1)$ in (22) is a large value of 0.8551 and 1.72 for 10% and 20% percent Gaussian multiplicative noises, respectively. Here, the values in heat-map indicate the relative magnitudes of connectivities, which means we cannot compare the values in different heat-maps. Note that normalization can be implemented on the final calculated connectivities, if necessary.

Conversely, the network outputs for P2, P3, and P4 are

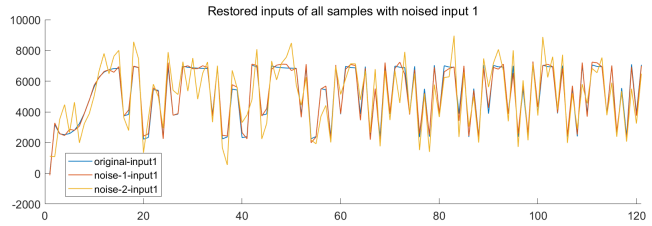

Fig. 6. The results for the *BiSA* method with Gaussian multiplicative noise perturbation on injection 1 in the *Streak Case* scenario.

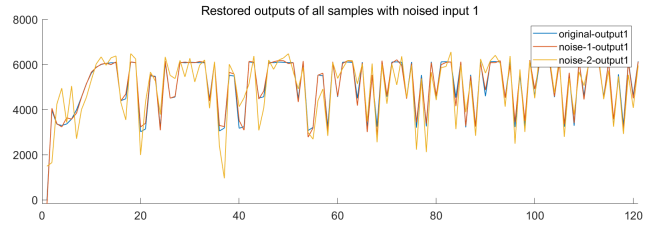

Fig. 7. The original producer 1 and the outputs for the producer 1 of the network with Gaussian noise perturbation on injection 1 in the *Streak Case* scenario for one run.

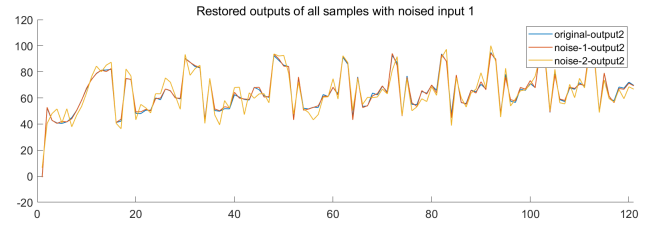

Fig. 8. The original producer 2 and the outputs for the producer 2 of the network with Gaussian noise perturbation on injection 1 in the *Streak Case* scenario for one run.

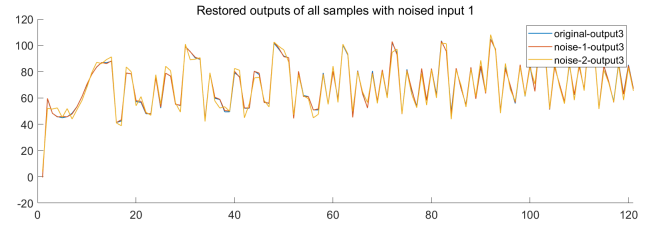

Fig. 9. The original producer 3 and the outputs for the producer 3 of the network with Gaussian noise perturbation on injection 1 in the *Streak Case* scenario for one run.

slightly changed with regard to the perturbations on I1, as shown in Figs. 8, 9, and 10. This is because the network has learned the low dependency of the pairs I1-P2, I1-P3, and I1-P4, which means I1 has little contributions to P2, P3, and P4. Hence, even if we add 20% percent Gaussian noise to I1, the obtained outputs of P2, P3, and P4 are still not perturbed much in the learned model. This leads to the low values of $BiS(1, m, 3, n)$ $(m = 1, 2, \cdots, M$ and $n = 1, 2, \cdots, N)$ in (22), as shown in the figures.

Next we perturb I3 also with respect 10% and 20% percent Gaussian multiplicative noises, as shown in Fig. 11, the graphs

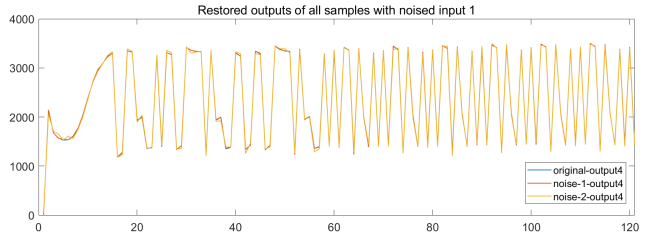

Fig. 10. The original producer 4 and the outputs for the producer 4 of the network with Gaussian noise perturbation on injection 1 in the *Streak Case* scenario for one run.

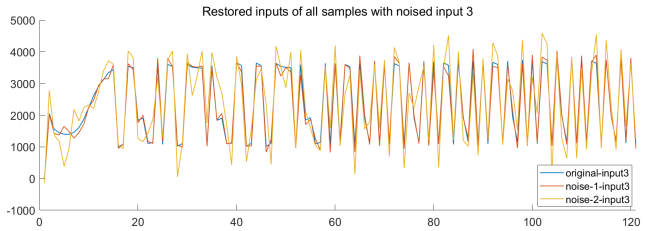

Fig. 11. The results for the *BiSA* method with Gaussian noise perturbation on injection 3 in the *Streak Case* scenario for one run.

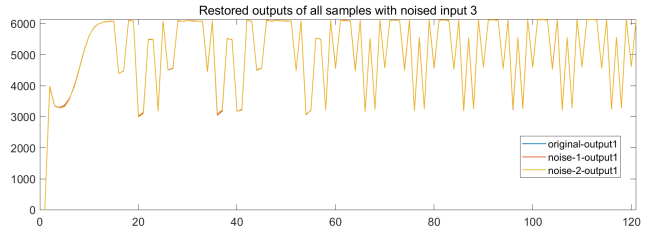

Fig. 12. The original producer 1 and the outputs for the producer 1 of the network with Gaussian noise perturbation on injection 3 in the *Streak Case* scenario for one run.

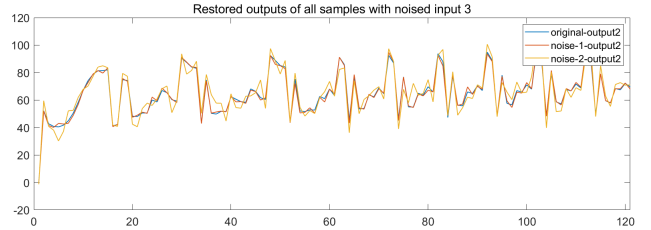

Fig. 13. The original producer 2 and the outputs for the producer 2 of the network with Gaussian noise perturbation on injection 3 in the *Streak Case* scenario for one run.

demonstrate consistent effects for the corresponding network outputs of the producers. Fig. 12 shows the original output of P1, the perturbed P1 with 10% percent noise, and the perturbed P1 with 20% percent noise, respectively. The three curves almost coincided together. This indicates that I3 contributes nearly no energy to P1 and hence, I3-P1 is judged to be a very low connected pair. Consequently, the calculated connection strength is significantly low having a value of 0.01133 and 0.02338 in the two heat-maps, respectively. Both values can even be ignored with comparison to the connectivity of I1-P1.

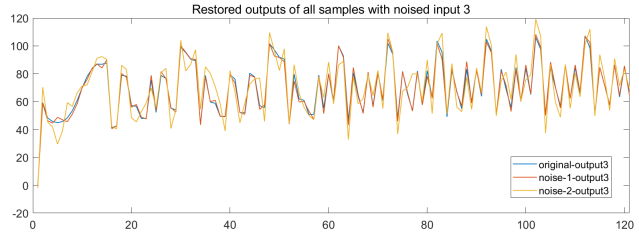

Fig. 14. The original producer 3 and the outputs for the producer 3 of the network with Gaussian noise perturbation on injection 3 in the *Streak Case* scenario for one run.

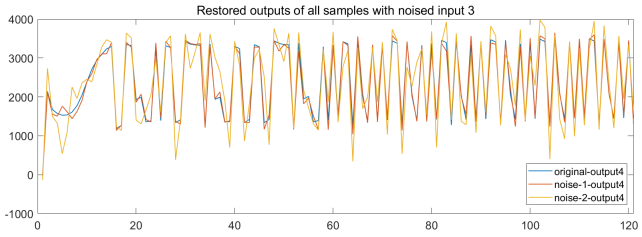

Fig. 15. The original producer 4 and the outputs for the producer 4 of the network with Gaussian noise perturbation on injection 3 in the *Streak Case* scenario for one run.

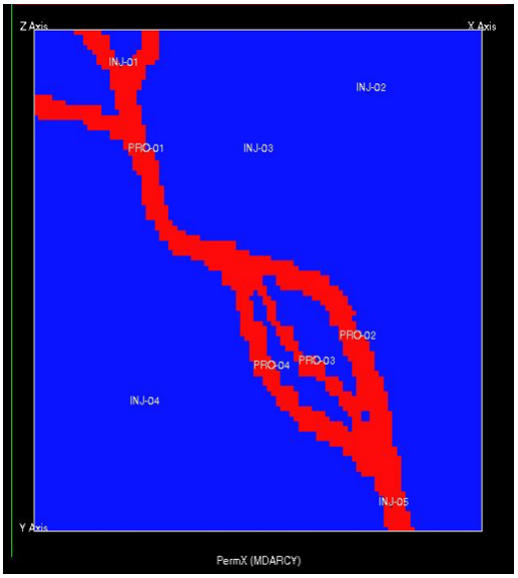

Fig. 16. Scenario 2: The permeability field of the *Braided River* with 4 injections and 4 producers. The permeability varies from 10md to 2080md.

Similar results can be observed for P2 and P3 as shown in Figs. 13 and 14, respectively. However, the obtained outputs of P4 according to the perturbations change a lot to the original output of P4 as shown in Fig. 15. This is due to the learned information in the training patterns that I3 has a high correlation (connectivity) to P4. Consequently, the computed *BiS* is relatively large as 0.3873 and 0.7564 respectively with regard to the two perturbations.

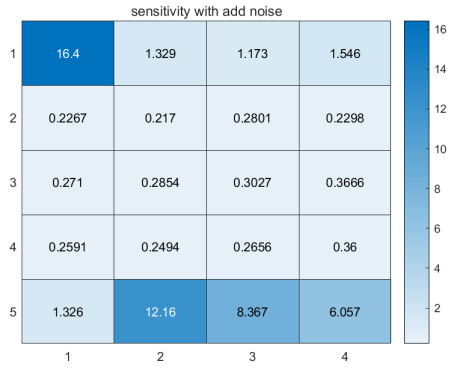

Fig. 17. The results for the *BiSA* method with 10 percent Gaussian noise perturbation on the *Streak Case* scenario.

C. Case Study 2: Braided River

Braided River is a complex scenario with several predominant pathways of water-flooding, 5 injectors (I1, I2, I3, I4, and I5) and 4 producers (P1, P2, P3, and P4). It is a conventional fluvial deposition widely distributed in the continental facies basin. The permeability of the river channel is very high yet those of the other areas are quite low. This case is composed of 100×100 single-layer grids on the horizontal plane, each with a size of $80ft \times 80ft \times 12ft$. The initial oil saturation is set to 0.7 and the porosity is is set to 0.18. In this case, the permeability of the river channels is set to 1000 md while that of other areas is set to 50 md. As shown in Fig. 16, I1 and P1 are located in the river channel on the top left. P2, P3, P4 are situated in three different channels respectively and I5 is located in the channel on the right bottom part.

Fig. 17 shows the results of the proposed *BiSA* method for the *Braided River* scenario with multiplicative 10% Gaussian noise perturbation on the water injection rates, where a darker color reflects a stronger connectivity. From the heat-map, the highly connected well pairs can be clearly observed. The highest one is I1-P1 with the connectivity value of 16.4. This is exactly the reflection of the channel-connected situation of I1 and P1 in the *Braided River* scenario. For pairs I5-P2, I5-P3, and I3-P4, the obtained connectivities are also higher with values 12.16, 8.367, and 6.057, respectively. They are also consistent with the actual situation in Fig. 17, where P2, P3, and P4 are well connected to I5 by three individual channels and I5-P2 is the highest among them. Nevertheless, the connectivities are quite low for the other pairs especially for I2, I3, and I4, caused by that all of these injectors are located in the low permeability area far from the river channels. Fig. 18 shows the results of the proposed *BiSA* method with addictive 20% Gaussian noise perturbation on the water injection rates of the *Streak Case* scenario. With stronger perturbation, the relative magnitude for each injector-producer pair are suitably represented as well. The four highly connected pairs, namely I1-P1, I5-P2, I5-P3, and I5-P4, are prominently characterized by the proposed methods. All these results are consistent with the real situation in *Braided River* scenario. These results clearly demonstrate the utility of the proposed methods.

TABLE I
COMPARISON WITH THE CURRENT METHODS ON THE *Streak Case*.

	ANN [55]				CRM [52]				BiSA-10%				BiSA-20%			
c/f	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4
I1	**0.995**	0.174	0.267	0.081	**1.000**	0.002	0.003	0.005	**1.000**	0.0083	0.0078	0.0712	**1.000**	0.0077	0.0072	0.0676
I2	0.110	0.033	0.001	0.061	0.548	0.003	0.137	0.303	0.0002	0.0001	0.0001	0.0003	0.0002	0.0001	0.0001	0.0003
I3	0.011	0.240	0.339	**1.000**	0.057	0.010	0.033	**0.974**	0.0132	0.0075	0.0071	**0.4529**	0.0135	0.0074	0.0070	**0.4397**
I4	0.273	0.580	0.243	0.473	0.117	0.178	0.001	0.662	0.0001	0.0000	0	0.0003	0.0001	0.0000	0	0.0002
I5	0.289	0.383	0.349	0.390	0.126	0.001	0.106	0.766	0.0001	0.0001	0.0001	0.0003	0.0001	0.0001	0.0001	0.0003

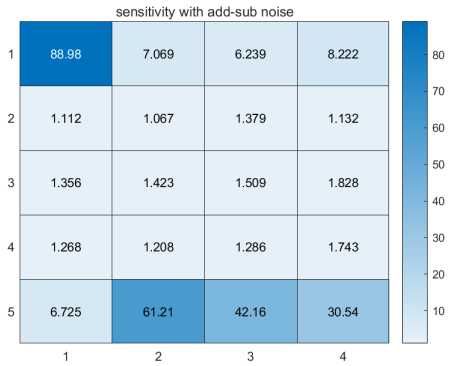

Fig. 18. The results for the *BiSA* method with 20 percent Gaussian noise perturbation on the *Streak Case* scenario.

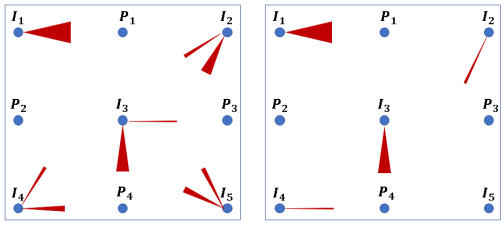

Fig. 19. The connectivity diagrams for the results of ANN (left) and the results by *BiSA* (right).

D. Comparisons with the current methods

Comparisons with the latest works in [55] and [52] are conducted to further validate the effectiveness of *BiSA*. Table I demonstrates the comparison results by the ANN method in [55], the capacitance resistance model (CRM) in [52], *BiSA* with 10% Gaussian noise, and *BiSA* with 20% Gaussian noise on the *Streak Case* scenario in Fig. 3. Although the original values without normalization for *BiSA* provide intuitive description, the used results of the proposed *BiSA* method in Table I are normalized to range [0,1] to make an explicit contrast.

From the table, both ANN and CRM methods can identify the two streaks of high permeability, however, the real characteristics are not appropriately depicted. For CRM method, the obtained connectivity of I3-P4 (0.974) is fairly close to that of I1-P1 (1.0). This is not a proper reflection of the fact that I1-P1 has about double permeability compared to I3-P4. Similarly, for the ANN method, both I1-P1 and I3-P4 have very high and similar values and more surprisingly I1-P1 exhibits a weather connection than I3-P4! Besides, the characterizations for some other inter-well pairs are not accurate as well, as shown by the values marked in red. From the description of the *Streak Case*, we know that except for the pairs I1-P1 and I3-P4, connectivities for the other injector-producer pairs are practically non-existence. However, the acquired connectivities for I3-P3, I4-P2, I4-P4, I5-P2, I5-P3,I5-P4 by the ANN method and those for I2-P1, I2-P4, I4-P4, I5-P4 by the CRM method are noticeably high. These results can mislead the operations in real production. For the ANN method, for instance, the obtained connectivity of I4-P2 is 0.580, which makes an illusion that this pair has a reasonably strong connection between them. However, both the two streaks and the other injector-producer pairs are appropriately characterized by the proposed *BiSA* method. From Table I, we can see that for the proposed method, the connectivity of I1-P1 is the most dominant indicating a high transmissibility path while I3-P4 is the second one having almost the half of the connectivity value of I1-P1. Also, the obtained values for other pairs are very close to zero. This reflects the real situation as in the *Streak Case* scenario. Moreover, the proposed *BiSA* method is more stable and trustable than ANN method in [55], which is greatly affected by the initial weights. Finally, the *BiSA* method is model-independent and quite efficient in computation, compared to the other two methods. All these results demonstrate the effectiveness of the proposed *BiSA* method.

In reservoir engineering, connectivity diagram is a commonly used tool to exhibit the inter-well connectivity. The results of *BiSA* can be easily changed into a diagram as shown in Fig. 19, where the left one shows the results of ANN on the *Streak Case* scenario and the right one shows the connectivity diagram for *BiSA* with 20% Gaussian noise. The performance for these two methods can be easily compared from these views. If a policymaker develops the injection-production scheme depending on the left connectivity diagram, he or she may make unreasonable decisions (undesirable exploration) especially for I4 and I5. Because this diagram shows superior channel connected to both injections. Conversely, the right diagram for *BiSA* is clear and easy to understand. The policymaker can gain a more accurate understanding of the connectivity in this field. These results further demonstrate the effectiveness of the proposed *BiSA* method.

V. CONCLUSION

In this paper, a stochastic sensitivity analysis method for (deep) neural networks, *Bilateral Sensitivity*, is proposed to measure the relationship between layers and neurons. Both the

Bilateral Sensitivity between any layer pair and the *Bilateral Sensitivity* between any neuron pair of different layers in deep neural networks are defined. Then the *Bilateral Sensitivity* from an input to an output of a multi-layer neural network is easily obtained to infer the connections between the inputs and outputs. The proposed method is calculated in a highly efficient way. As long as the function expression between the input parameters and the output variables is given, namely the network is trained, both the *Bilateral Sensitivity* between layers and the *Bilateral Sensitivity* of each output variable with respect to different inputs can be effectively analyzed.

Based on this, the proposed *Bilateral Sensitivity* of neural network is applied to characterize the well connectivity in reservoir engineering. Given a trained network by Water Injection Rates (WIRs) and Liquid Production Rates (LPRs) data, the well connectivity can be efficiently characterized by the measure of *Bilateral Sensitivity*. The empirical results verify the effectiveness of the proposed method and the comparisons with some state-of-the-art methods demonstrate its superior performance. Besides, the proposed *Bilateral Sensitivity* is independent of the model and is easy to be implemented on other neural networks such as RBFs and CNNs.

REFERENCES

[1] W. He, Y. Chen, Z. Yin, "Adaptive Neural Network Control of an Uncertain Robot With Full-State Constraints", *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 620-629, March 2016.

[2] T. Zhang, W. Zheng, Z. Cui, et al., "Spatial-Temporal Recurrent Neural Network for Emotion Recognition", *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 839-847, March 2019.

[3] Z. Zhao, P. Zheng, S. Xu, et al., "Object Detection with Deep Learning: A Review", *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, November 2019.

[4] G. Riccardo, M. Anna, R. Salvatore, et al., "A Survey of Methods for Explaining Black Box Models", *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1-42, June 2018.

[5] R. V. Borges, A. D. Garcez, L. C. Lamb, "Learning and Representing Temporal Knowledge in Recurrent Networks", *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2409-2421, 2011.

[6] S. Y. Wong, K. S. Yap, H. J. Yap, et al., "On Equivalence of FIS and ELM for Interpretable Rule-Based Knowledge Representation", *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 7, pp. 1417-1430, July 2015.

[7] A. Suarez, J. F. Lutsko, "Globally optimal fuzzy decision trees for classification and regression", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 12, pp. 1297-1311, December 1999.

[8] B. Zhou, D. Bau, A. Oliva, et al., "Interpreting Deep Visual Representations via Network Dissection", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, SEPTEMBER 2019.

[9] R. H. Kewley, M. J. Embrechts, C. Breneman, "Data strip mining for the virtual design of pharmaceuticals with neural networks", *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 668-679, May 2000.

[10] N. W. Townsend, L. Tarassenko, "Estimations of error bounds for neural-network function approximators", *IEEE Trans. Neural Netw.*, vol. 10, no. 2, pp. 217-230, March 1999.

[11] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto, "Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models", *Hoboken*, NJ, USA: Wiley, February 2004.

[12] E. Horel, V. Mison, T. Xiong, et al., "Sensitivity based Neural Networks Explanations", *arXiv:1812.01029v1*, pp. 1-9, December 2018.

[13] L. Zhang, X. Sun, Y. Li, et al., "A Noise-Sensitivity-Analysis-Based Test Prioritization Technique for Deep Neural Networks", *arXiv:1901.00054v3, pp. 1-8, January* 2019.

[14] J. D. Rodriguez, A. Perez, J. A. Lozano, "Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 569-575, March 2010.

[15] W. W. Y. Ng, D. S. Yeung, X. Z. Wang, et al., "A study of the difference between partial derivative and stochastic neural network sensitivity analysis for applications in supervised pattern classification problems", *the 3rd Int. Conf. Mach. Learn. Cybern.*, August 2004, pp. 4283-4288.

[16] M. Stevenson, R. Winter, B. Widrow, "Sensitivity of feedforward neural networks to weight errors", *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 71-80, March 1990.

[17] X. Zeng, D. S. Yeung, "Sensitivity Analysis of Multilayer Perceptron to Input and Weight Peaurbations", *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1358-1366, November 2001.

[18] J. Y. Choi, C.H. Choi, "Sensitivity analysis of multilayer perceptron with differentiable activation functions", *IEEE Trans. Neural Netw.*, vol. 3, no. 1, pp. 101-107, January 1992.

[19] W. Samek, A. Binder, et al., "Evaluating the visualization of what a Deep Neural Network has learned", *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, pp. 2660-2673, November 2017.

[20] S. Hashem, "Sensitivity analysis for feedforward artificial neural networks with differentiable activation functions", *IEEE Int. Joint Conf. Neural Netw.* (IJCNN), July 1992, pp. 419-424.

[21] L. Fu, T. Chen, "Sensitivity analysis for input vector in multilayer feedforward neural networks", *IEEE Int. Joint Conf. Neural Netw.* (IJCNN), February 1993, pp. 215-218.

[22] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks", *IEEE Trans. Neural Netw.*, vol. 1, no. 2, June 1990.

[23] A. P. Engelbrecht, "A new pruning heuristic based on variance analysis of sensitivity information", *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1386-1399, November 2001.

[24] M. Kusy, P. A. Kowalski, "Weighted Probabilistic Neural Network", *Inf. Sci.*, vol. 430-431, pp. 65-76, March 2018.

[25] F. Weber, S. Theers, D. Surmann, U. Ligges, and C. Weihs, "Sensitivity analysis of ordinary differential equation models", *Tech. Rep.*, May 2018.

[26] P. A. Kowalski, M. Kusy, "Sensitivity Analysis for Probabilistic Neural Network Structure Reduction", *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1919-1932, May 2018.

[27] J. M. Zurada, A. Malinowski, and S. Usui, "Perturbation method for deleting redundant inputs of perceptron networks", *Neurocomputing*, vol. 14, no. 2, pp. 177193, 1997.

[28] M. E. Hoff, "Leaming phenomena in networks of adaptive switching circuits", Department of Electrical Engineering, Stanford University, July 1962.

[29] P. M. Frank, "Introduction of System Sensitivity Theory", *Academic Press*, New York, 1978.

[30] S. W. Piche, "The selection of weight accuracies for Madalines", *IEEE Trans. Neural Netw.*, vol. 6, no. 2, pp. 432-445, March 1995.

[31] C. Alippi, V. Piuri, M. Sami, "Sensitivity to Errors in Artificial Neural Networks: A Behavioral Approach", *IEEE Trans. Circuits Syst. I*, vol. 42, no. 6, pp. 358-361, June 1995.

[32] A. Saltelli, S. Tarantola, P. S. Chan, "A Quantitative Model-Independent Method for Global Sensitivity Analysis of Model Output", *Technometrics*, vol. 41, no. 1, pp. 39-56, February 1999.

[33] A. Saltelli, S. Tarantola, "On the Relative Importance of Input Factors in Mathematical Models", *J. Am. Stat. Assoc.*, vol. 97, no. 459, pp. 702-709, September 2002.

[34] E. Fock, "Global Sensitivity Analysis Approach for Input Selection and System Identification Purposes-A New Framework for Feedforward Neural Networks", *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 8, August 2014.

[35] F. Fern'andez-Navarro, M. Carbonero-Ruz, et al., "Global Sensitivity Estimates for Neural Network Classifiers", *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, pp. 2592-2604, November 2017.

[36] I. M. Sobol, "Sensitivity estimates for nonlinear mathematical models", *Math. Model. Comp. Exp.*, vol. 1, no. 4, pp. 407-414, 1993.

[37] A. Y. Cheng, D. S. Yeung, "Sensitivity analysis of neocognitron", *IEEE Trans. Syst., Man and Cybern.*, Part C (Applications and Reviews), vol. 29, no. 2, pp. 238-249, May 1999.

[38] D. S. Yeung, X. Sun, "Using function approximation to analyze the sensitivity of MLP with antisymmetric squashing activation function", *IEEE Trans. Neural Netw.*, vol. 13, no. 1, pp. 34-44, January 2002.

[39] D. Shi, D. S. Yeung, J. Gao, "Sensitivity analysis applied to the construction of radial basis function networks", *Neural Networks*, vol. 18, no. 7, pp. 951-957, September 2005.

[40] D. S. Yeung, W. W. Y. Ng, D. Wang, E. C. C. Tsang, and X.-Z. Wang, "Localized generalization error model and its application to architecture selection for radial basis function neural network", *IEEE Trans. Neural Netw.*, vol. 18, no. 5, pp. 12941305, September 2007.

[41] W. W. Y. Ng, J. Hu, D. S. Yeung, et al., "Diversified Sensitivity-Based Undersampling for Imbalance Classification Problems", *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2402-2412, November 2015.

[42] D. S. Yeung, J.-C. Li, W. W. Y. Ng, and P. P. K. Chan, "MLPNN training via a multiobjective optimization of training error and stochastic sensitivity", *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 5, pp. 978992, May 2016.

[43] W. W. Y. Ng, D. S. Yeung, M. Firth, et al., "Feature selection using localized generalization error for supervised classification problems using RBFNN", *Pattern Recognit.*, vol. 41, no. 12, pp. 3706-3719, December 2008.

[44] W. W. Y. Ng, Z.-M. He, D. S. Yeung, and P. P. K. Chan, "Steganalysis classifier training via minimizing sensitivity for different imaging sources", *Inf. Sci.*, vol. 281, pp. 211224, October 2014.

[45] B. Karmakar, N. R. Pal, "How to Make a Neural Network say 'Don't Know'", *Inf. Sci.*, vol. 430-431, pp. 444-466, March 2018.

[46] W. Xiang, H. D. Tran, T. T. Johnson, "Output Reachable Set Estimation and Verification for Multi-Layer Neural Networks", *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, November 2018.

[47] B. Li, D. Saad, "Large Deviation Analysis of Function Sensitivity in Random Deep Neural Networks", *arXiv:1910.05769*, pp. 1-25, October 2019.

[48] C. Baykal, L. Liebenwein, I. Gilitschenski, et al., "SiPPing Neural Networks: Sensitivity-informed Provable Pruning of Neural Networks", *arXiv:1910.05422v1*, pp. 1-18, October 2019.

[49] H. Shu, H. Zhu, "Sensitivity Analysis of Deep Neural Networks", *arXiv:1901.07152v1*, pp. 1-8, January 2019.

[50] C. Zhang, A. Liu, X. Liu, et al. "Interpreting and Improving Adversarial Robustness with Neuron Sensitivity", *arXiv:1909.06978v2*, pp. 1-32, November 2019.

[51] E. Unal, F. Siddiqui, A. Rezaei, et al., "Use of Wavelet Transform and Signal Processing Techniques for Inferring Interwell Connectivity in Waterflooding Operations", *Soc. Petrol. Engrs.*, SPE-196063-MS, October 2019, pp. 1-13.

[52] M. Sayarpour, E. Zuluaga, et al., "The Use of Capacitance-Resistive Models for Rapid Estimation of Waterflood Performance", *Soc. Petrol. Engrs.*, doi:10.2118/110081-MS, November 2007, pp. 1-13.

[53] A. A. Yousef, P. Gentil, J. L. Jensen, et al., "A Capacitance Model To Infer Interwell Connectivity From Production and Injection Rate Fluctuations", *Soc. Petrol. Engrs.*, doi:10.2118/95322-MS, October 2005, pp. 1-19.

[54] Y. Wang, C. S. Kabir, Z. Reza, "Inferring Well Connectivity in Waterfloods Using Novel Signal Processing Techniques", *Soc. Petrol. Engrs.*, doi:10.2118/191643-MS, September 2018, pp. 1-22.

[55] E. Artun, "Characterizing Reservoir Connectivity and Forecasting Waterflood Performance Using Data Driven and Reduced-Physics Models", *Soc. Petrol. Engrs.*, SPE-180488-MS, Western Regional Meeting, May 2016, pp. 1-27.

[56] W. Liu, W. D. Liu, J. Gu, "Reservoir Inter-Well Connectivity Analysis Based on a Data Driven Method", *Soc. Petrol. Engrs.*, doi:10.2118/197654-MS, November 2019, 1-12.

[57] A. Albertoni, L. W. Lake, "Inferring Interwell Connectivity Only From Well-Rate Fluctuations in Waterfloods", *Soc. Petrol. Engrs.*, doi:10.2118/83381-PA, February 2003, pp. 6-16.

[58] U. Demiryurek, F. Banaei-Kashani, C. Shahabi, "Neural-Network based Sensitivity Analysis for Injector-Producer Relationship Identification", *Soc. Petrol. Engrs.*, doi:10.2118/112124-MS, January 2008, pp. 1-10.

[59] J. Quackenbush, "Microarray data normalization and transformation", *Nat. Genet.*, vol. 32, pp. 496-501, January 2003.

[60] A. Jain, K. Nandakumar, A. Ross, "Score normalization in multimodal biometric systems", *Pattern Recognit.*, vol. 38, no. 12, pp. 2270-2285, December 2005.

[61] M. Sayarpour, CRM Generator, *https://www.researchgate.net/publication/280579204_CRM_Generator*, July 2015.