A Quantitative Insight Into the Role of Skip Connections in Deep Neural Networks of Low Complexity: A Case Study Directed at Fluid Flow Modeling

Abouzar Choubineh
Department of Computer Science,
University of Liverpool,
Liverpool L69 7ZX, UK;
Department of Applied Mathematics,
Xi'an Jiaotong-Liverpool University,
Suzhou, 215123, China
e-mails: a.choubineh@liverpool.ac.uk;
a.choubineh20@student.xjtlu.edu.cn

Jie Chen
Department of Applied Mathematics,
Xi'an Jiaotong-Liverpool University,
Suzhou, 215123, China
e-mail: jie.chen01@xjtlu.edu.cn

Frans Coenen
Department of Computer Science,
University of Liverpool,
Liverpool L69 7ZX, UK
e-mail: coenen@liverpool.ac.uk

Fei Ma
Department of Applied Mathematics,
Xi'an Jiaotong-Liverpool University,
Suzhou, 215123, China
e-mail: fei.ma@xjtlu.edu.cn

Deep feed-forward networks, with high complexity, backpropagate the gradient of the loss function from final layers to earlier layers. As a consequence, the "gradient" may descend rapidly toward zero. This is known as the vanishing gradient phenomenon that prevents earlier layers from benefiting from further training. One of the most efficient techniques to solve this problem is using skip connection (shortcut) schemes that enable the gradient to be directly backpropagated to earlier layers. This paper investigates whether skip connections significantly affect the performance of deep neural networks of low complexity or whether their inclusion has little or no effect. The analysis was conducted using four Convolutional Neural Networks (CNNs) to predict four different multiscale basis functions for the mixed Generalized Multiscale Finite Element Method (GMsFEM). These models were applied to 249,375 samples. Three skip connection schemes were added to the base structure: Scheme 1 from the first convolutional block to the last, Scheme 2 from the middle to the last block, and Scheme 3 from the middle to the last and the second-to-last blocks. The results demonstrate that the third scheme is most effective, as it increases the coefficient of determination (R^2) value by 0.0224–0.044 and decreases the Mean Squared Error (MSE) value by 0.0027–0.0058 compared to the base structure. Hence, it is concluded that enriching the last convolutional blocks with the information hidden in neighboring blocks is more effective than enriching using earlier convolutional blocks near the input layer.

Keywords: deep neural network, backpropagation, vanishing gradient phenomenon, skip connection, GMsFEM, heterogeneous porous media

1 Introduction

Deep feed-forward networks are typically generated using some form of gradient-based training, such as backpropagation, whereby the gradient of the loss function (cost function) is calculated using the values assigned to weights and biases. The loss function measures how well the network is operating by considering the similarity between real and predicted outputs. There are various parameter optimizers that can be adopted to arrive at the minimum loss value. Multi-layer perceptron neural networks typically include a feed-forward pathway in which different layers are arranged, and parameters are initialized, as well as a backward pathway which progressively modifies parameters, thus gradually improving a model's performance.

Deep neural networks contain several hidden layers as opposed to shallow networks, which have only one hidden layer. A significant advantage of deep networks with high complexity is that they can represent complex functions and learning features at various levels of abstraction. However, the disadvantage is the vanishing gradient phenomenon which may occur during the training process. As the network backpropagates the error gradient from the final layers to layers closer to the input layer, the gradient can descend rapidly to zero. This issue causes those parameters associated with layers near the input layer not to change as much as they should. In the worst case, the updating of these layers may cease to happen. This in turn is likely to have an adverse effect on the operation of the network.

The Convolutional Neural Network (CNN) has, over recent years, become one of the most trusted Deep Learning (DL) models with respect to many application domains, particularly applications involving image data [1,2]. A classic CNN model is typically composed of alternate convolutional and pooling layers, followed by one or more Fully Connected (FC) layers (dense layers) at the end. In some circumstances, it is possible to replace an FC layer with a global average pooling layer. The convolutional and pooling layers perform feature extraction, while the FC layers map the extracted features into an output layer [3,4].

The concept of using a skip connection (a shortcut) in a neural network was first proposed by He et al. as a way of mitigating the vanishing gradient problem in deep CNN models [5]. The shortcut enables the cost function gradient to be directly backpropagated to layers close to the input layer. In traditional CNN architectures, and the layers come one after another. Using the skip connection idea, a shortcut is added to the main path in the network.

Despite much recent research directed at adopting skip connections in highly complex deep networks [6–10], there is a lack of critical analysis of the nature and causes of the vanishing gradient problem, and the comparative advantages gained when using skip connections in neural networks of low complexity. Additionally, although classification and regression algorithms are both categorized as supervised learning algorithms, reported work on the utilization of skip connections has mostly been directed at classification; there is much less reported work directed at regression. The major contribution of this paper is to demonstrate the impact on the error performance of a low-complexity CNN when skip connections are added into the network. There are two important motivations for doing so. First, the gradient in such networks might descend so rapidly to zero, possibly causing early convergence. Second, it might be the case that attempting to improve the performance of a CNN by simply adding more convolution blocks into the network has zero effect because of the vanishing gradient problem. The work presented is illustrated using a high-dimensional regression problem taken from the domain of subsurface fluid flow modeling. This problem can be modeled in a variety of ways; the method used in this paper is founded on the mixed Generalized Multiscale Finite Element Method (GMsFEM) [11]. The disadvantage of this method is that it requires the solution of large numbers of Partial Differential Equations (PDEs) to produce a set of "basis functions." However, it is possible to predict each basis function and avoid their time-consuming calculation using DL, for example, a CNN. We consider four such CNNs coupled with three different kinds of identity skip connections.

The paper continues by providing a detailed overview of the mixed GMsFEM case study. Then, the seven steps involved in developing the models are presented. The statistical results with regard to the coefficient of determination (R^2) and Mean Squared Error (MSE) are given in the following. A discussion of the sources of prediction error along with some suggested solutions is given in the final section.

2 Case Study

The application focus for our work is modeling fluid flow in the earth's subsurface. This is of interest to both engineers and scientists with respect to, for example, the seepage of waste water through soil, the flow of oil and gas to wells, and land subsidence as a consequence of groundwater extraction. Despite taking serious steps toward renewable energy, the oil/gas industry still provides a high proportion of the world's energy. The main goal is to predict the performance of reservoirs at any future point in time and to optimize the petroleum fluid recovery under different operating conditions. In oil/gas heterogeneous porous media, formation-related properties can have multiple scales. More specifically, there may be numerous fractures with different lengths, whose width is much smaller than the domain size. The permeability, defined as the ability of a rock to permit fluids to pass through it, in fractures is generally much higher than that of the matrix. This is why the effect of fractures must be considered when modeling flow processes. In the mesh generation stage of numerical modeling, adequately fine grids are used to resolve small-scale fractures. By doing so, the discrete formulation of such problems produces a large system of equations, and consequently, the number of unknown parameters increases. The computation of the solution therefore becomes expensive. Multiscale techniques can be used to decrease the degrees-of-freedom and solve subsurface flow problems on coarse grids. These methods can considerably decrease resolution times without a significant loss of precision.

Local mass conservation is of great importance for subsurface flow problems. The mixed multiscale finite element method is one of the most used mass conservative multiscale techniques. A mixed GMsFEM has recently been presented to solve Darcy's flow (a linear relationship between pressure gradient and velocity) of a single-phase fluid in two-dimensional fractured porous media [11]. This new method approximates the pressure in the multiscale finite element space between the coarse-grid space and fine-grid space; several multiscale basis functions are obtained in a single coarse-grid element, and the velocity is directly approximated in the fine-grid space.

The following flow problem in the mixed formulation is considered:

$$\begin{gathered} k^{-1}u + \nabla p = 0 \;\; in \;\; \Omega \\ \nabla \cdot u = f \;\; in \;\; \Omega \end{gathered} \tag{1}$$

with nonhomogeneous boundary conditions:

$$u \cdot n = g \;\; on \;\; \partial\Omega \tag{2}$$

where k is permeability, u is the Darcy velocity, p is the pressure, f is the source term, g is the given normal component of the Darcy velocity on the boundary, Ω is the computational domain, and n is the outward unit norm vector on the boundary.

To illustrate the general solution framework of the mixed GMsFEM, τ^H is considered a confirming partition of Ω into finite elements with coarse block size H, and τ^h is the fine-grid partition with mesh size h. By defining $V = H(div, \Omega)$ and $W = L^2(\Omega)$, the mixed finite element spaces will be

$$\begin{aligned} V_h = \{ & v_h \in V\colon v_h(t) = (b_t x_1 + a_t,\, d_t x_2 + c_t), \\ & a_t,\ b_t,\ c_t,\ d_t \in \mathbb{R}, t \in \tau^h \} \end{aligned}$$

$$W_h = \{ w_h \in W\colon w_h \; is \; a \; constant \; on \; each \; element \; in \; \tau^h \}$$

Supposing $\{\Psi_j\}$ is the set of multiscale base functions for the coarse element, the multiscale space for the pressure p is defined as the linear span of all local basis functions, which can be denoted as

$$W_H = \oplus \{\Psi_j\} \;\; in \;\; \tau^H$$

The mixed GMsFEM is directed at finding $(u_H, p_H) \in (V_h, W_H)$ such that

$$\begin{gathered} \int k^{-1} u_H \cdot v_H - \int div(v_H) p_H = 0 \;\; \forall \; v_H \in V_h^0 \\ \int div(u_H) w_H = \int f w_H \;\; \forall \; w_H \in W_H \end{gathered} \tag{3}$$

where $u_H \,.\, n = g_H$ on $\partial\Omega$ for each coarse edge on the boundary and g_H is the average of function g on the corresponding coarse edge.

To systematically approximate the pressure p, the following approach is adopted to construct the multiscale space W_H. A snapshot space is defined by solving a series of local cell problems on every coarse-grid element with Dirichlet's boundary condition. In Dirichlet's condition, a value is first assigned to the pressure. Then, the snapshot space is further decreased to find the dominant modes, where a local eigenvalue problem (one for each coarse element) needs to be solved. The linear span of these modes is termed the offline space.

The number of PDEs that need to be solved to construct multiscale basis functions is equal to the number of local cell problems plus the number of local eigenvalue problems. The local cell problem defined in the coarse grid is the same as the original problem except that the source function is omitted. A delta function is assigned to the boundary of the coarse grid as a boundary condition. The delta function is defined as follows: a value of 1 is given for a fine-grid edge and a value of 0 for all other grid edges. Consequently, the number of local cell problems that need to be solved equals the number of fine-grid edges contained in the coarse-grid boundary (for the work presented here this is 12). In this research, the computational domain was set to be $\Omega = [0, 1]^2$. A 30×30 uniform mesh was selected for the fine-grid system and a 10×10 uniform mesh was used for the coarse-grid system (Fig. 1). Given 100 coarse elements, there are 1300 PDEs: 1200 (100×12) local cell problems, and 100 (100×1) local eigenvalue problems. For each randomly generated permeability field (as the only input), there were five basis functions, identified as Bases 1, 2, 3, 4, and 5. The first was a piecewise constant, containing only -1 and $+1$. Because these two values were the same for the finite element method, there was no need to train this basis. Bases 2, 3, 4, and 5 were in a range of $(-1, +1)$. To produce these basis functions, many PDEs (here 1300) must first be solved; this presents considerable overhead. For this reason, it makes sense to replace PDE solvers with data-driven methods, given their outstanding capabilities and general acceptance in recent decades. The mixed GMsFEM can be accelerated by the constructed data-driven-based models.

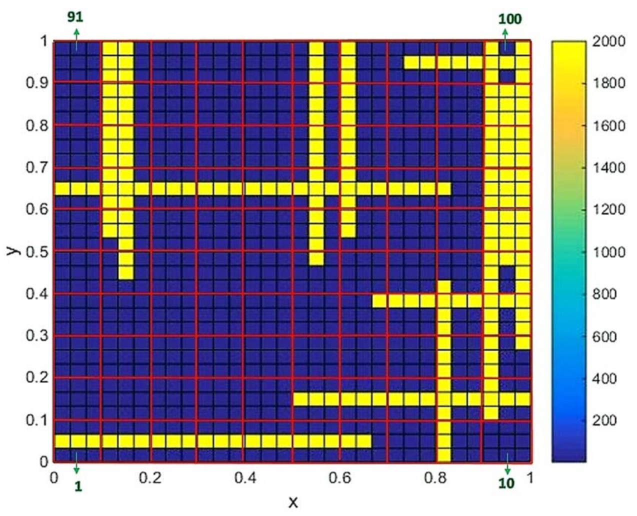


Fig. 1 A permeability field of a fractured porous medium with K_m of 4 millidarcy and K_f of 2000 millidarcy. The fine grid squares in blue refer to the matrix and those in yellow to the fracture. The red grid indicates the coarse grid. Each coarse grid square contains nine fine grids. The number of fractures is 15.

3 Model Development

Figure 2 illustrates the steps involved for CNN model generation. The "Spyder" module provided with the "Anaconda Distribution" of PYTHON 3.5 was used. A seven-step process was adopted as follows:

Step 1: Load "Keras" with "TensorFlow" as the backend, "Numpy," "Pandas," "Sklearn," and "Glob" libraries.

Step 2: Organize and input the data sets. A range of values for the permeability of the matrix (K_m) was chosen from 1 to 5 millidarcy incrementing in steps of 1; and for the permeability of the fracture (K_f) from 500 to 2000 millidarcy incrementing in steps of 250. The number of fractures was set to 1, 2, 3, …, 23, 24, and 25 (25 cases). The format of permeability fields was initially in a 900×1 vector, which transformed to a 100×9 two-dimensional tensor. Here, 100 indicates the number of coarse grids, and 9 refers to the number of fine grids in each coarse grid (Fig. 1), meaning that each row is affiliated with a coarse grid. This replacement enables the use of two-dimensional convolutional kernels while developing the model. On the other hand, basis functions remained in the form of a 900×1 vector due to adding FC layers at the end of the network.

For each of the 875 ($5 \times 7 \times 25 = 875$) cases, the "MATLAB" code was run as many as 280 times over the training data, two times for validation, and three times for testing. Accordingly, 249,375 samples were produced with 245,000 examples for training, 1750 for validation, and 2625 for testing.

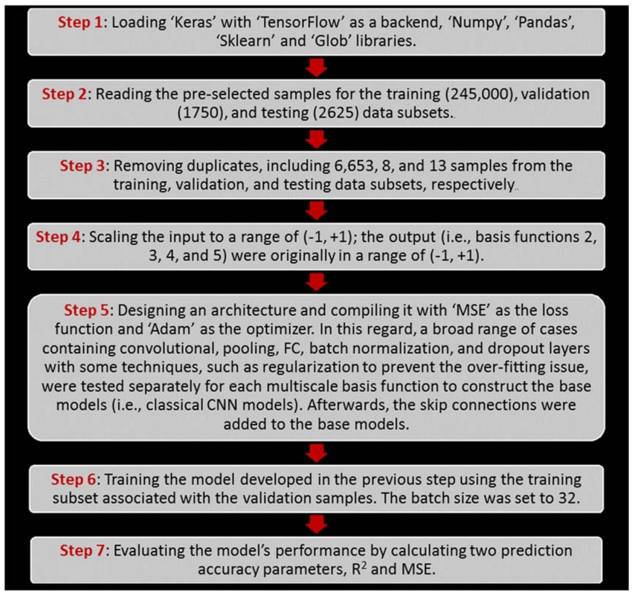
Step 3: Remove duplicates. Since the permeability fields were randomly generated, duplicates could have been included. These were removed to avoid giving samples an advantage or bias when running the algorithm. In this regard, 6653 training, 8 validation, and 13 testing samples were excluded. This decreased the training, validation and testing samples to 238,347, 1742, and 2612, respectively.

Step 4: Scale the permeability field (as the only input) to a range of (−1, +1).

Step 5: Design an optimal architecture and then add the skip connection. To develop an optimal base model for each multiscale basis function, we started with a low number of epochs to ensure the model was still improving based on the obtained MSE values for the training and validation subsets. Thereafter, the number of epochs was increased up to 100. The above five steps were performed separately for each basis function. The various cases containing convolutional, pooling, FC, batch normalization, regularization, and dropout layers were tested separately for each basis function. The same optimal architecture was coincidentally obtained for all Bases 2, 3, 4, and 5 (Fig. 3 (base)). The architecture has five convolutional layers and two FC layers, without any pooling layer. The number of kernels in each convolutional layer (referred to as CONV1 to CONV5) is 5, 10, 15, 20, and 25, respectively. CONV1, CONV2, CONV3, CONV4, and CONV5 have the sizes 98×7, 96×5, 94×3, 92×1, 92×1, respectively. A batch normalization layer can be added after each convolutional layer without changing its size. Neural networks use higher learning rates and converge faster by normalization of the input layer. This normalization maintains the output average at a value close to zero and the standard deviation close to one. Each FC layer contains 2000 neurons. For a given neuron or kernel, the inputs are multiplied by the weights and summed together. Then, a bias term is added. By doing so, only a linear transformation is performed on the inputs using the weights and biases. Although this operation makes the neural network simpler, it is less powerful and unable to learn complex patterns in a data set. This is where the activation function is beneficial. Mathematically, this can be represented as shown in Eq. (4) where w_i represents the weight value, z_i is the input value, b is the bias, f is the activation function, and y is the output. The models use the "Rectified Linear Unit (ReLU)" activation function for the convolutional layers, "sigmoid" for the FC layers, and "linear" for the output

$$y = f\left(\sum_{i=1}^{n} (w_i z_i) + b\right) \tag{4}$$

It is useful to compare the optimal base architecture with well-known structurally similar CNN architectures, such as AlexNet [12] and VGGNet (also known as VGG16) [13] to provide a better insight. Like common CNN models, when going deeper through the structure of the developed models, the number of feature maps increases and their size decreases. However, the number of feature maps (the filters number) defined here is significantly less than that of common CNN models. Another difference is that there is no pooling layer in our developed models, given that the input dimension is 100×9. Unlike AlexNet and VGGNet, no dropout layer is used in the FC layers. Considering the fact that AlexNet and VGGNet are very complex, with many convolutional layers and filters, we can call the presented architecture here a "deep neural network of low complexity."


Step 1: Loading 'Keras' with 'TensorFlow' as a backend, 'Numpy', 'Pandas', 'Sklearn' and 'Glob' libraries.

Step 2: Reading the pre-selected samples for the training (245,000), validation (1750), and testing (2625) data subsets.

Step 3: Removing duplicates, including 6,653, 8, and 13 samples from the training, validation, and testing data subsets, respectively.

Step 4: Scaling the input to a range of (-1, +1); the output (i.e., basis functions 2, 3, 4, and 5) were originally in a range of (-1, +1).

Step 5: Designing an architecture and compiling it with 'MSE' as the loss function and 'Adam' as the optimizer. In this regard, a broad range of cases containing convolutional, pooling, FC, batch normalization, and dropout layers with some techniques, such as regularization to prevent the over-fitting issue, were tested separately for each multiscale basis function to construct the base models (i.e., classical CNN models). Afterwards, the skip connections were added to the base models.

Step 6: Training the model developed in the previous step using the training subset associated with the validation samples. The batch size was set to 32.

Step 7: Evaluating the model's performance by calculating two prediction accuracy parameters, R^2 and MSE.

Fig. 2 Steps involved in developing a CNN model

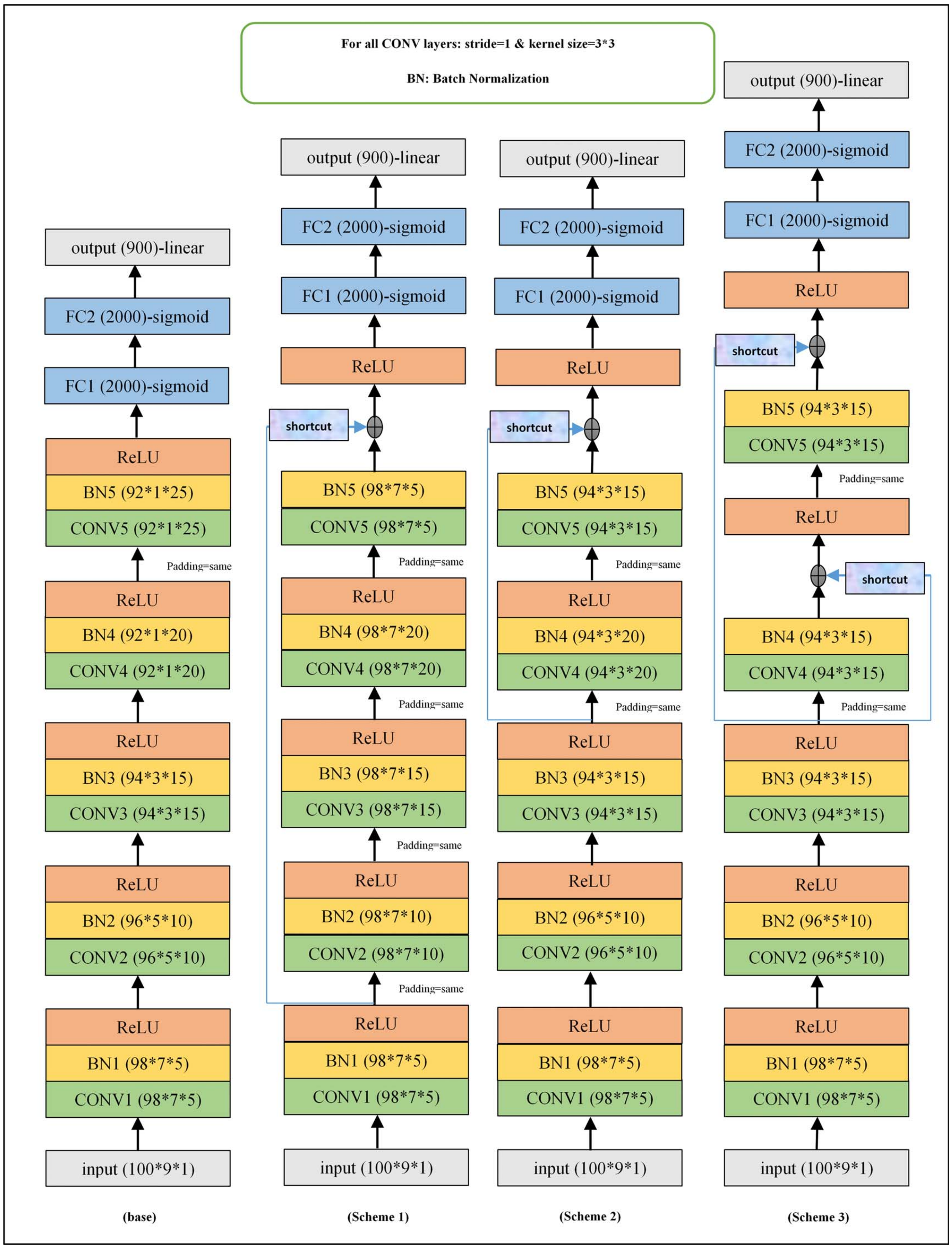


Fig. 3 **Structure of the three skip connection schemes added to the base architecture**

To address the central objective of this paper, comparisons were conducted using three different shortcut schemes as shown in Fig. 3. In Scheme 1, a single shortcut is added from the output of the first convolution process to the last convolutional block. The input and output of this part have the same dimension of 98*7 because an identity type of shortcut is used. The second scheme is designed to discover how much a shortcut from the middle layer to the final layer can affect the performance of a model. Here, the input and output of this section with the shortcut have a dimension of 94*3. The third scheme is mainly considered to gain knowledge

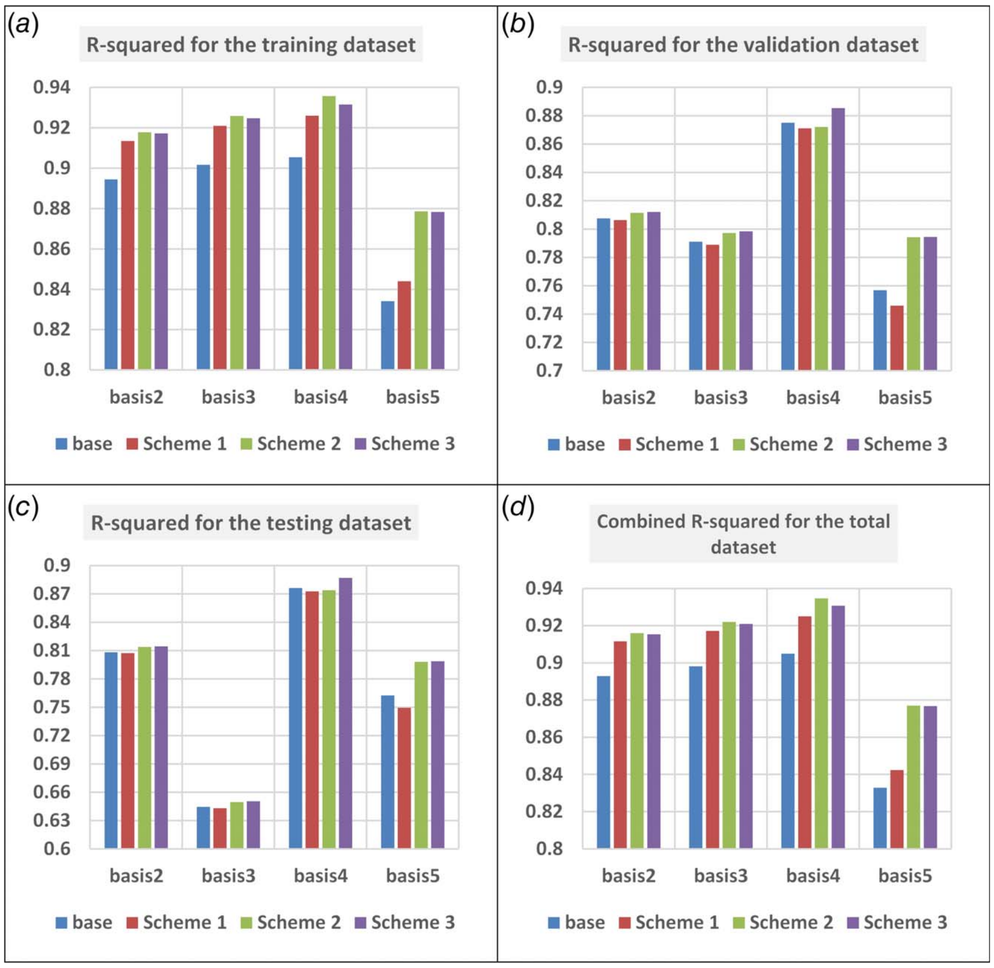


Fig. 4 R^2 performance comparison using skip connection Schemes 1, 2, and 3, and the base case, for (*a*) the training dataset, (*b*) the validation dataset, (*c*) the testing dataset, and (*d*) the total dataset

of the effect of involving the raw input features along with Scheme 2. In all three cases, the main path and the shortcut meet each other before applying the activation function. The structure of the FC layers remains unchanged for all three architectures.

Step 6: Train the network. Training a CNN model refers to finding weights and biases of the kernels in the convolutional layers as well as of the neurons (nodes) in the FC layers to minimize the difference between the outputs of the model and known values as much as possible.

Step 7: Evaluate the trained network's performance.

4 Results

The performance of the developed models is evaluated using two prediction error metrics: R^2 and MSE. Figure 4 presents four bar charts giving the R^2 results of the developed models for the training, validation, and testing subsets, and the entire dataset, respectively. Figure 4(*a*) demonstrates that all three skip connection schemes produce an improved performance over the base case for all multiscale basis functions with respect to the training dataset. For example, Scheme1 increases it from 0.9054 to 0.926 for Basis 4. Both Schemes 2 and 3 perform marginally better than Scheme 1. From Figs. 4(*b*) and 4(*c*), it is shown that Scheme 1 has an adverse effect on the validation and testing subsets. For example, with respect to Basis 5, the R^2 value decreases from 0.7569 and 0.7625, to 0.746 and 0.7495 for validation and testing, respectively. Except for Basis 4, Scheme 2 demonstrates an improved performance over the other base models. Scheme 3 is beneficial in all cases, especially for Bases 4 and 5. The trend that can be observed in Figs. 4(*a*)–4(*d*) is the same because a major part of the data generated was designed as the training data.

Figure 5 presents four bar charts giving the MSE results obtained. The intuition here was that the MSE results would reflect the R^2 results (Fig. 4). From Fig. 5(*a*), it can be seen that for the training subset, all three skip connection schemes serve to decrease the MSE of the base models. Additionally, Schemes 2 and 3 produce a better performance than Scheme 1. From Figs. 5(*b*) and 5(*c*), it is shown that the MSE fractionally increases for the validation and testing subsets when using Scheme 1. For instance, it increases from 0.0158 to 0.0165 for Basis 5 in the case of the validation dataset. According to Figs. 5(*b*) and 5(*c*), Scheme 2 does not favorably influence Basis 4. Figure 5(*d*) demonstrates that the "total data" MSE values follow the same trend as in Fig. 5(*a*), again because the training data dominates the total dataset.

In addition to the graphical comparisons presented in Figs. 4 and 5, the R^2 and MSE values are listed in full in Table 1. For each subset and multiscale basis function, the best result is highlighted in pink.

The results obtained for Scheme 1 imply that transferring feature maps of earlier convolutional layers to final ones has only a marginal positive effect with respect to the training dataset. In other words, the corresponding skip connection makes the predictive model concentrate on capturing the underlying trend of the training (seen) subset. Flowing information from the middle convolutional layer to the last layer (Scheme 2) works for Bases 2, 3, and 5 regarding all three subsets. This means that the feature maps of the middle convolution process contain important information. Adding two

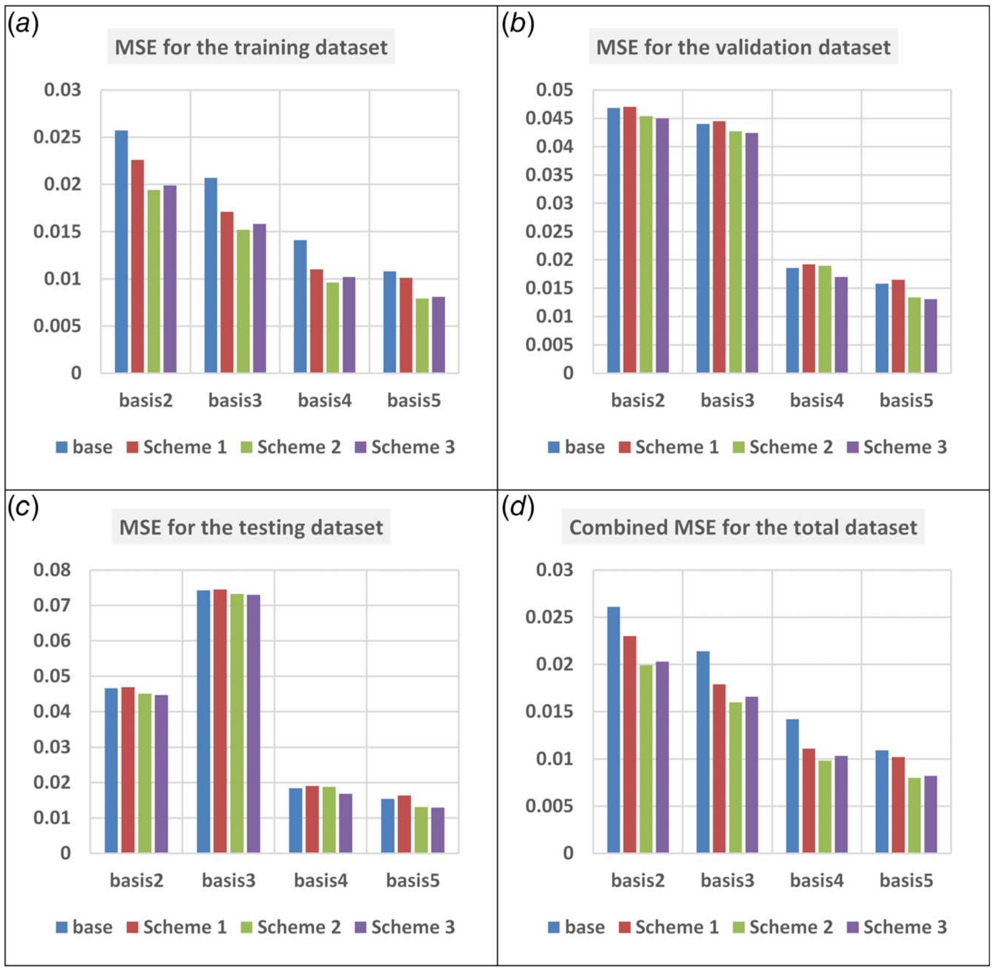


Fig. 5 MSE performance comparison using skip connection Schemes 1, 2, and 3, and the base case, for (*a*) the training dataset, (*b*) the validation dataset, (*c*) the testing dataset, and (*d*) the total dataset

skip connections (Scheme 3) favorably affects all the basis functions with respect to the training, validation, and testing datasets. By comparing the architecture and results produced using Schemes 2 and 3, the positive role of transferring raw feature maps is understandable. Therefore, enriching the last convolutional blocks with information hidden in the neighboring layers is more efficient than enriching them using earlier convolutional blocks near the input layer.

Multiscale basis functions are defined in a single coarse grid element. In order to better understand what they are, the pattern

Table 1 The R^2 and MSE values obtained for skip connection Schemes 1, 2, and 3, and the base case

		R^2				MSE			
		Basis 2	Basis 3	Basis 4	Basis 5	Basis 2	Basis 3	Basis 4	Basis 5
Training	Base	0.8945	0.9017	0.9054	0.8341	0.0257	0.0207	0.0141	0.0108
	Scheme 1	0.9135	0.921	0.926	0.844	0.0226	0.0171	0.011	0.0101
	Scheme 2	0.9178	0.9259	0.9358	0.8786	0.0194	0.0152	0.0096	0.0079
	Scheme 3	0.9172	0.9248	0.9316	0.8783	0.0199	0.0158	0.0102	0.0081
Validation	Base	0.8076	0.7911	0.8751	0.7569	0.0468	0.044	0.0186	0.0158
	Scheme 1	0.8063	0.789	0.8711	0.746	0.047	0.0445	0.0192	0.0165
	Scheme 2	0.8115	0.7973	0.8721	0.7942	0.0454	0.0427	0.019	0.0134
	Scheme 3	0.8121	0.7985	0.8854	0.7945	0.045	0.0424	0.017	0.0131
Testing	Base	0.8083	0.6445	0.8762	0.7625	0.0466	0.0743	0.0184	0.0154
	Scheme 1	0.8074	0.6429	0.8726	0.7495	0.0469	0.0745	0.019	0.0163
	Scheme 2	0.8138	0.6496	0.874	0.7981	0.0451	0.0732	0.0188	0.0131
	Scheme 3	0.8146	0.6503	0.887	0.7987	0.0447	0.073	0.0168	0.0129
Total	Base	0.8929	0.8981	0.9049	0.8328	0.0261	0.0214	0.0142	0.0109
	Scheme 1	0.9116	0.9171	0.925	0.8423	0.023	0.0179	0.0111	0.0102
	Scheme 2	0.9159	0.922	0.9347	0.8771	0.0199	0.016	0.0098	0.008
	Scheme 3	0.9153	0.9209	0.9308	0.8768	0.0203	0.0166	0.0103	0.0082

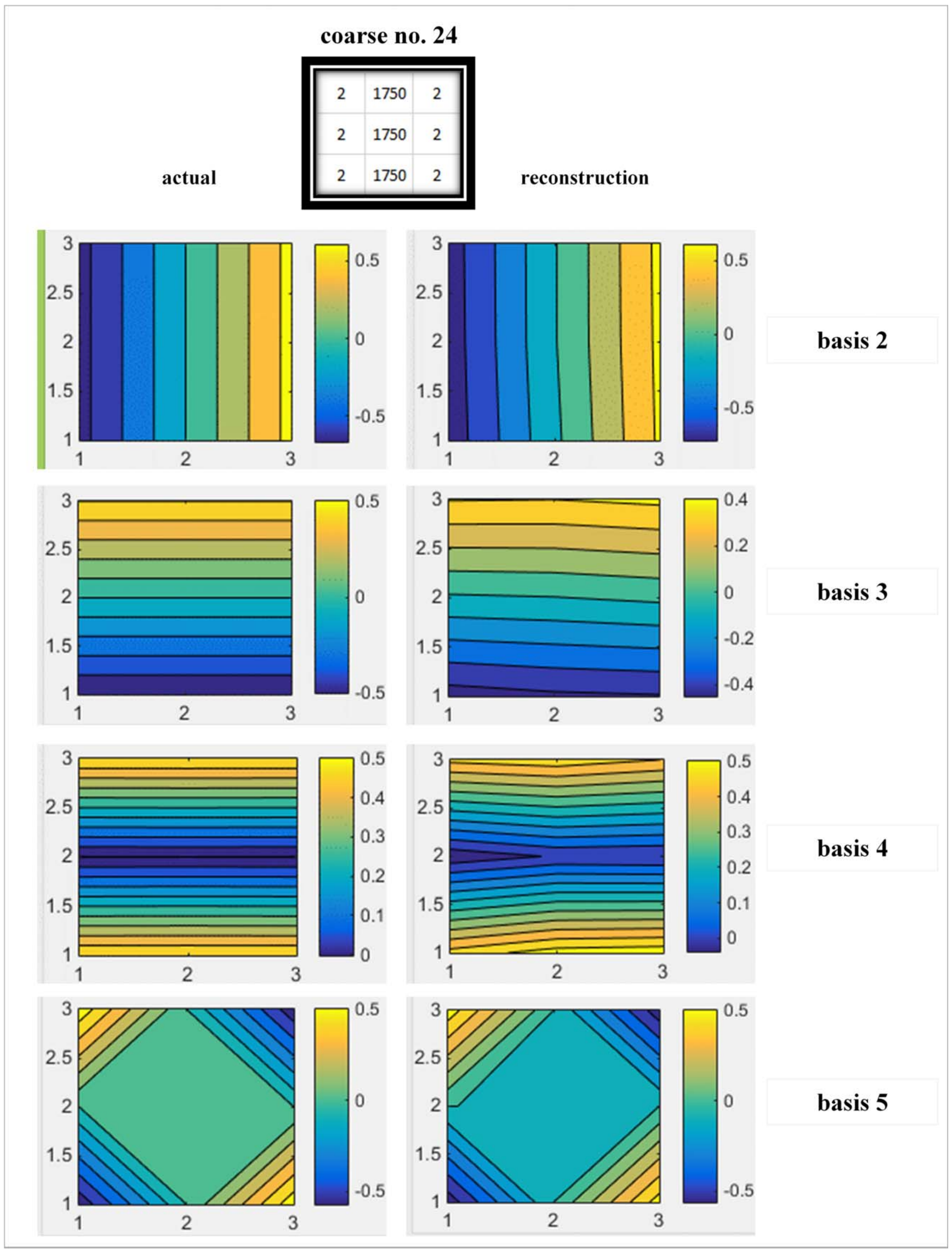


Fig. 6 A comparison between the actual and reconstructed patterns for a fractured case

available in a fractured case with coarse block no. 24 (as a representative sample) is tracked in Fig. 6. The figure demonstrates an excellent match between actual and reconstructed patterns for this coarse element.

Figure 7 offers an example of the pressure and velocity distributions for a representative permeability field. According to this figure, there is a very close match between the actual distributions and the ones obtained by the developed models in this study.

5 Discussion

Four classical CNN models, with five convolutional and two FC layers, were separately developed to predict four different multiscale basis functions in a mixed GMsFEM. It is generally accepted that adjusting the architecture can obtain better DL accuracy. This is why three skip connection schemes were added to the base structures in this paper. The results are generally promising, especially for Scheme 3. However, the accuracy of the models over the testing data set is not as high as we would like. There are both irreducible and reducible prediction errors. The irreducible error is caused by noise in the problem itself and its data. This error cannot be removed. The case study presented here is a high-dimensional problem with 249,375 samples, mapping an input of 100*9 to an output of 900*1. Adding the skip connections increases the complexity of the base structure, making it more challenging to train the network. There are as many as 10,414,170 trainable and 150 non-trainable parameters for the base structure, without shortcuts. By adding Scheme 1, the number of trainable and non-trainable parameters increases to 12,670,510 and decreases to 110, respectively. For the second scheme, there are 14,272,340 trainable and 130 non-trainable parameters. For Scheme 3, the number of trainable and non-trainable parameters changes to 14,270,975 and 120, respectively. Several solutions are suggested to reduce the prediction error parameters: generating more data, increasing the number of

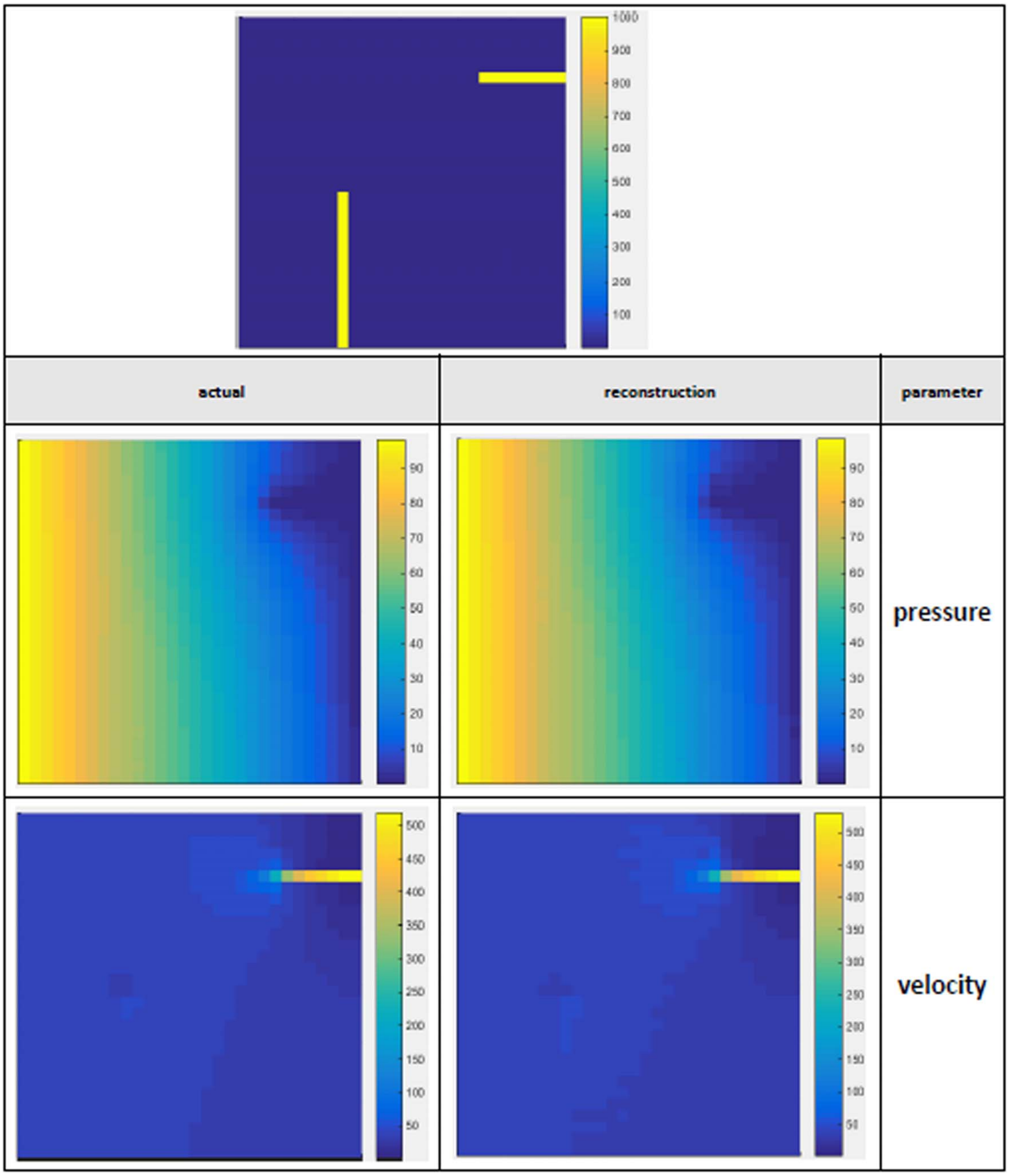


Fig. 7 A comparison between the actual pressure and velocity and the ones obtained by the developed models for a representative permeability field

epochs, and combining the currently developed low-complexity models into an ensemble system. Furthermore, applying skip connections to other case studies on low-complexity deep neural networks could reinforce the finding that skip connections improve the model performance.

Acknowledgement

We would like to thank Ms. Roslyn Joy Irving and Dr. Trevor Mahy for providing helpful comments and suggestions.

Funding Data

- This work is partially supported by Key Program Special Fund in XJTLU (KSF-E-50, KSF-E-21), XJTLU Postgraduate Research Scholarship (PGRS1912009), and XJTLU Research Development Funding (RDF-19-01-15).

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The data sets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

References

[1] Nie, Z., Jiang, H., and Kara, L. B., 2020, "Stress Field Prediction in Cantilevered Structures Using Convolutional Neural Networks," ASME J. Comput. Inf. Sci. Eng., **20**(1), p. 011002.

[2] Deshpande, S., and Purwar, A., 2021, "An Image-Based Approach to Variational Path Synthesis of Linkages," ASME J. Comput. Inf. Sci. Eng., **21**(2), p. 021005.

[3] Yamashita, R., Nishio, M., Do, R. K. G., and Togashi, K., 2018, "Convolutional Neural Networks: an Overview and Application in Radiology," Insights Imaging, **9**(4), pp. 611–629.

[4] Elgendy, M., 2020, *Deep Learning for Vision Systems*, Manning Publications.

[5] He, K., Zhang, X., Ren, S., and Sun, J., "Deep Residual Learning for Image Recognition," in Proc. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.

[6] Drozdzal, M., Vorontsov, E., Chartrand, G., Kadoury, S., and Pal, C., 2016, "The Importance of Skip Connections in Biomedical Image Segmentation," *Deep Learning and Data Labeling for Medical Applications*, Springer, New York, pp. 179–187.

[7] Ahn, H., and Yim, C., 2020, "Convolutional Neural Networks Using Skip Connections with Layer Groups for Super-Resolution Image Reconstruction Based on Deep Learning," Appl. Sci., **10**(6), p. 1959.

[8] Wu, L., Lin, X., Chen, Z., Huang, J., Liu, H., and Yang, Y., 2021, "An Efficient Binary Convolutional Neural Network with Numerous Skip Connections for Fog Computing," IEEE Internet Things J., **8**(14), pp. 11357–11367.

[9] Das, S. K., Roy, P., and Mishra, A. K., 2022, "Recognition of Ischaemia and Infection in Diabetic Foot Ulcer: a Deep Convolutional Neural Network Based Approach," Int. J. Imaging Syst. Technol., **32**(1), pp. 192–208.

[10] Paul, A., Kundu, A., Chaki, N., Dutta, D., and Jha, C., 2022, "Wavelet Enabled Convolutional Autoencoder Based Deep Neural Network for Hyperspectral Image Denoising," Multimedia Tools Appl., **81**(2), pp. 2529–2555.

[11] Chen, J., Chung, E. T., He, Z., and Sun, S., 2020, "Generalized Multiscale Approximation of Mixed Finite Elements with Velocity Elimination for Subsurface Flow," J. Comput. Phys., **404**, p. 109133.

[12] Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012, "Imagenet Classification With Deep Convolutional Neural Networks," Adv. Neural Inf. Process. Syst., **25**, pp. 1097–1105.

[13] Simonyan, K., and Zisserman, A., 2014, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint.