Electronic Transactions on Numerical Analysis.
Volume 56, pp. 187–208, 2022.
Copyright © 2022, Kent State University.
ISSN 1068–9613.
DOI: 10.1553/etna_vol56s187

ETNA
Kent State University and
Johann Radon Institute (RICAM)

A DEEP LEARNING BASED NONLINEAR UPSCALING METHOD FOR TRANSPORT EQUATIONS[*]

TAK SHING AU YEUNG[†], ERIC T. CHUNG[‡], AND SIMON SEE[§]

Abstract. We will develop a nonlinear upscaling method for nonlinear transport equations. The proposed scheme gives a coarse scale equation for the cell average of the solution. In order to compute the parameters in the coarse scale equation, a local downscaling operator is constructed. This downscaling operation recovers fine scale properties using cell averages. This is achieved by solving the equation on an oversampling region with the given cell average as constraint. Due to the nonlinearity, one needs to compute these downscaling operations on the fly and cannot pre-compute these quantities. In order to give an efficient downscaling operation, we apply a deep learning approach. We will use a deep neural network to approximate the downscaling operation. Our numerical results show that the proposed scheme can achieve good accuracy and efficiency.

Key words. nonlinear upscaling, transport equations, deep learning

AMS subject classifications. 65N30, 65N40

1. Introduction.

Multiscale methods or numerical upscaling for linear problems have been widely developed and are very useful for multiscale problems. Nevertheless, many realistic problems are of nonlinear and multiscale nature. For example, the dynamics of multi-phase flow and transport in heterogeneous media varies over multiple space and time scales. In order to accurately capture the coarse scale dynamics, some types of nonlinear upscaling are necessary. There are many linear and nonlinear upscaling techniques in the literature, some of these include [1, 2, 3, 4, 7, 10, 11, 12, 15, 19, 20, 21, 22, 23, 26, 27, 32, 33, 35, 37, 40, 41, 42, 45, 50]. Nonlinear upscaling methods, such as the pseudo-relative permeability approach [6, 8, 38], compute nonlinear relative permeability functions based on single cell two-phase flow computations. It is known that these nonlinear approaches lack robustness and that they are processes dependent [24, 25]. To overcome these difficulties, one needs to find better nonlinear upscaling techniques, and it is the goal of this paper to do this.

Some nonlinear upscaling approaches are available in the literature. One example is the nonlinear homogenization [29, 44], for which local nonlinear problems are solved on each coarse grid block and are used in the construction of global coarse grid formulation. Nonlinear homogenization is applied to many problems including the p-Laplace equations, pseudo-elliptic equations, and parabolic equations [28, 29, 30]. These methods typically require the assumption of scale separation, and may give limited accuracy for applications that do not admit such assumption. Another nonlinear upscaling approach popular in computational mechanics is the computational continua framework [34]. These methods, for example [31, 34], use nonlocal quadrature to couple the coarse scale system stated on a union of some disjoint computational unit cells with the aim of solving problems with non-scale-separation heterogeneous media.

The nonlinear upscaling method developed in this paper is motivated by the general framework, called Nonlinear Nonlocal Multicontinua Upscaling (NLNLMC), of nonlinear upscaling presented in [18]. This framework originates from the Constraint Energy Minimizing

[*]Received July 07, 2020. Accepted September 20, 2021. Published online on March 8, 2022. Recommended by Axel Klawonn.

[†]Department of Mathematics,The Chinese University of Hong Kong, Hong Kong Special Administrative Region (iauyeung@math.cuhk.edu.hk).

[‡]Department of Mathematics,The Chinese University of Hong Kong, Hong Kong Special Administrative Region (tschung@math.cuhk.edu.hk).

[§]NVIDIA – NVAITC, Santa Clara, United States (ssee@nvidia.com).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Generalized Multiscale Finite Element Method (CEM-GMsFEM) [13, 14, 16]. The CEM-GMsFEM gives multiscale basis functions that are localizable even for the case of highly heterogeneous and high contrast media. The idea there is the use of local spectral problems and an energy minimization principle. The basis functions are constructed by solving problems on some oversampling regions. In addition, a rigorous convergence analysis is presented showing that the convergence depends only on the coarse grid size and is independent of the media. However, the degrees of freedom in CEM-GMsFEM do not have physical meaning. In upscaling, one desires unknown variables that have physical meanings such as cell averages. With this goal in mind, the Nonlocal Multicontinua Upscaling (NLMC) is introduced [17]. The key idea follows CEM-GMsFEM except that the multiscale basis functions are modified so that the degrees of freedom represent the average of the solutions. Therefore, the resulting coarse scale equation gives a relation between the cell averages of the solutions on the coarse grid, and a convergence analysis is given in [52]. We remark that both CEM-GMsFEM and NLMC methods are suitable for linear multiscale problems.

The framework of NLNLMC extends the concept of the NLMC method with the aim of finding nonlinear upscaling for nonlinear problems [18]. In this case, one needs to avoid the concept of basis functions, as they are only applicable to linear problems. In general, the NLNLMC framework has three important methodological ingredients. First, we identify macroscopic variables for each coarse grid block, similar to multicontinua variables [5, 39, 43, 49]. Secondly, we will construct local downscaling functions. In particular, given the macroscopic variables, we will solve local problems on some oversampling regions to recover fine scale properties. It is important that these local problems are solved on oversampling regions, since this allows connectivity of neighboring macroscopic variables. Finally, a global coarse scale equation is obtained by combining all local downscaling functions and using a suitable coarse scale solver. The resulting scheme is a coarse scale equation that relates all macroscopic variables. Moreover, the connectivity is nonlocal as the oversampling regions can be several coarse grid layers wide.

The goal of this paper is to construct a nonlinear upscaling approach for nonlinear transport equations. We will apply the general concept of NLNLMC together with deep learning techniques. For the macroscopic variable, we will use cell average on coarse elements, and for the global coarse scale solver, we will use an upwind finite volume scheme. The main component of the proposed scheme is the local downscaling function as these will provide the parameters required in the final coarse grid equation. Given a set of macroscopic values, we will solve a local problem on an oversampling region to construct a fine scale downscaling function, whose mean values on coarse elements match the given macroscopic values. In general, this is an expensive task as these local problems are solved on-the-fly when the solution averages are given since one cannot easily pre-compute these problems. This fact motivates the use of deep learning. There are in the literature some works on using deep neural networks to learn macroscopic parameters in coarse scale or reduced order models; see for example [9, 46, 47, 48, 51]. The main idea is to consider the macroscopic variables as input and the downscaling functions or their average values as output. Then suitable deep neural networks are trained and used to approximate this expensive procedure. The resulting approach allows the use of deep neural network to learn the parameters required in the coarse scale equations. This can give a significant improvement in the computational times, as we will see in our numerical simulations. We remark that using deep neural network for reduced models allows a robust learning process as there are fewer parameters to be learned.

The paper is organized as follows. In Section 2, we will present the problem formulations and some basic notations. In Section 3, the key elements of the proposed method will be discussed in detail. This includes the construction of the method and the deep neural network, as well as some implementation details. Computational results will be presented in Section 4 to validate our scheme. We will show the performance of our method, and compare our method with a standard scheme without using nonlinear upscaling. We will also compare the accuracy and efficiency with and without the use of deep neural networks. Finally, a conclusion is given in Section 5.

2. Preliminaries.

2.1. Basic setup.

2.1. Basic setup. Let $\Omega \subset \mathbb{R}^2$ be a computational domain in two space dimensions and let $T > 0$ be a fixed time. Our goal is to design a nonlinear upscaling method for the following transport equation:

$$\begin{aligned} \frac{\partial S}{\partial t} + \text{div}\,(v\lambda(S)) &= f, && \text{in } (0,T)\times\Omega, \\ S &= g, && \text{on } (0,T)\times\Gamma, \\ S(0,x) &= S_0(x), && \text{in } \Omega, \end{aligned}$$

where $\Gamma = \{x \in \partial\Omega \;:\; v\cdot n < 0\}$ is the inflow boundary of Ω, n is the outward unit normal vector of $\partial\Omega$ and f, g, and S_0 are given functions. Motivated by applications, we assume that the velocity v is divergence free, that is, $\text{div}(v) = 0$. Moreover, the function $\lambda : \mathbb{R} \to \mathbb{R}$ is a nonlinear function. This problem is motivated by two phase flow and transport problems, in which the velocity v is given by Darcy's law [36]. In general, the transport equation and Darcy's law are coupled. In this paper, we focus only on solving the nonlinear transport equations. The development of upscaling methods for the coupled problem will be considered in a forthcoming paper.

Next, we introduce the notions of coarse grids. Let $\mathcal{T}_H$ be a conforming partition of Ω into finite elements, where H is the coarse mesh size. This partition is called coarse grid. Let N_K be the number of elements in the coarse mesh. Then we assume that each coarse element is partitioned into a connected union of fine-grid blocks and this partition is called $\mathcal{T}_h$. Note that $\mathcal{T}_h$ is a conforming refinement of the coarse grid $\mathcal{T}_H$ with the mesh size h. That is, for any $\tau \in \mathcal{T}_h$, there exist $K \in \mathcal{T}_H$ such that $\tau \subset K$; see Figure 2.1 for an illustration. In our method, we will develop a nonlinear upscaling technique that gives the coarse grid mean value of the solution S as a function of time. At the same time, we will use the fine grid and local problems to construct the parameters in the proposed upscaling model.

2.2. Oversampling domains. The use of oversampling domains plays a crucial role in our nonlinear upscaling technique. Two types of oversampling domains will be used. The first type of oversampling domains, simply called oversampling domain, will be used for local problems. These local problems will be solved on oversampling regions, and the parameters in the nonlinear upscaled model will be determined based on the these local solutions. For the transport type problems considered in this paper, the size of the oversampling domains is determined by a suitable coarse time scale and the speed of propagation. The second type of oversampling domains, called double oversampling domain, will be used for the purpose of imposing boundary conditions for local problems. This kind of domains allows an artificial layer so that artificial boundary condition imposed on them will not affect the solution in the interior. In the next paragraph, definitions of the oversampling domains are given.

Let K be a coarse element in $\mathcal{T}_H$. We will construct an oversampling domain K^+ and a double oversampling domain K^{++}. The oversampling domain K^+ is obtained by enlarging K by several coarse grid layers, while the double oversampling domain K^{++} is obtained by enlarging K^+ by several coarse grid layers. An illustration of the fine grid, coarse grid, and oversampling domain are depicted in Figure 2.1. In this example, the oversampling domain K^+ is obtained by enlarging K by one coarse grid layer, and the double oversampling domain K^{++} is obtained by enlarging K^+ by one coarse grid layer. The oversampling domain K^+ contains a total of nine coarse rectangles and the double oversampling domain K^{++} contains twenty-five coarse rectangles if the coarse rectangle is not near the boundary. These oversampling domains are used for the computations regarding the local problem.

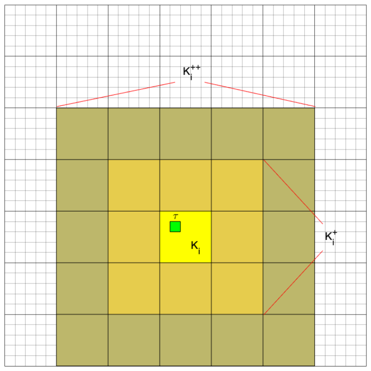

FIG. 2.1. *An illustration of the coarse grid, fine grid and oversampling domains.*

3. Method description.

3.1. Algorithm.

In this section, we are going to present the construction of our proposed method. The main idea is based on the nonlinear NLMC framework proposed in [18]. We will apply the general concept from [18] but with some new concepts specific for the transport problem considered in this paper. We consider the following nonlinear transport problem, in which we assume the sources $f = 0$ and $g = 0$ to simplify the notations. The proposed method can be easily extended to the case of nonzero sources. We consider

$$\begin{aligned} \frac{\partial S}{\partial t} + \text{div}\,(v\lambda(S)) &= 0, && \text{in } (0,T)\times\Omega,\\ S &= 0, && \text{on } (0,T)\times\Gamma,\\ S(0,x) &= S_0(x), && \text{in } \Omega. \end{aligned}$$

Our goal is to design a nonlinear upscaling method that can provide the cell averages of the solution $S(t,x)$ on the coarse grid $\mathcal{T}_H$.

For the time discretization, we will apply the backward Euler scheme. Let $\Delta t > 0$ be the time step size and let $t_n = n\Delta t$. The backward Euler scheme reads

$$\gamma S^{n+1} + \text{div}\left(v\lambda(S^{n+1})\right) = \gamma S^n \tag{3.1}$$

where $\gamma = \frac{1}{\Delta t}$ and $S^n = S(t_n, x)$. Our proposed nonlinear upscaling method is designed to solve (3.1). That is, given the mean values of the solution at time t_n, we will find the mean values of the solution at t_{n+1}. We use the notation $\bar{S}^n_\alpha$ to denote the average of S^n on the coarse cell $K_\alpha \in \mathcal{T}_H$. Our proposed scheme will compute $\{\bar{S}^{n+1}_\alpha\}$ by using $\{\bar{S}^n_\alpha\}$.

Assume that $\{\bar{S}^n_\alpha\}$ are known. Following the general framework proposed in [17, 18, 52], we will solve a local problem to find a downscaling function. To construct this downscaling function, we consider a set of values $\{\bar{S}_\beta\}$, where each $\bar{S}_\beta$ represents the average value of a certain function on the coarse cell K_β. We remark that these values $\bar{S}_\beta$ will be the required values $\{\bar{S}_\beta^{n+1}\}$ after solving our upscaled model. To get the required downscaling function, we will solve a local problem to find a local downscaling function $\tilde{\psi}$ such that the mean values of $\tilde{\psi}$ on the coarse cells are given by the values $\{\bar{S}_\beta\}$. Specifically, let K_α^{++} be a double oversampling domain of the coarse cell $K_\alpha \in \mathcal{T}_H$. We consider the local problem:

$$\gamma\tilde{\psi} + \operatorname{div}\left(v\lambda(\tilde{\psi})\right) = \gamma\bar{S}^n, \qquad \text{in } K_\alpha^{++}, \tag{3.2}$$

$$\tilde{\psi} = 0, \qquad \text{on } \partial^- K_\alpha^{++}, \tag{3.3}$$

$$\frac{1}{|K_\beta|}\int_{K_\beta} \tilde{\psi} = \bar{S}_\beta, \qquad K_\beta \subset K_\alpha^{+} \text{ and } K_\beta \in \mathcal{T}_H \tag{3.4}$$

where $\partial^- K_\alpha^{++}$ is the inflow boundary of K_α^{++}. The above problem is solved numerically on the fine grid $\mathcal{T}_h$. The solution $\tilde{\psi}$ will give the required local downscaling function. Next, we restrict the solution $\tilde{\psi}$ on K_α^+ and denote it as ψ_α, which is the required local downscaling function whose support is K_α^+. We remark that a suitable Lagrange multiplier is needed to solve (3.2)–(3.4).

Using these local solutions ψ_α, we define a global downscaling function $\psi = \sum \chi_\alpha\psi_\alpha$, where $\{\chi_\alpha\}$ is a set of partition of unity functions corresponding to the overlapping partition $\{K_\alpha^+\}$ of the domain. We remark that this global function ψ depends on the mean values $\{\bar{S}_\beta\}$. Now we solve (3.1) using this global downscaling function. We will integrate the equation (3.1) on each coarse element K_i and replace the true solution S^{n+1} by ψ. In particular, we have

$$\gamma\int_{K_i} \psi + \int_{\partial K_i} \lambda(\psi)v\cdot n = \gamma|K_i|\bar{S}_i^n. \tag{3.5}$$

Here, we use the notation $\bar{S}^n$ to denote the values $(\bar{S}_1^n, \ldots, \bar{S}_{N_K}^n)$. We will solve this nonlinear equation and obtain the unknowns $\{\bar{S}_\beta\}$. These values will be the required solution at the time step t_{n+1}. That is, we set $\bar{S}_\beta^{n+1}$ to be the solution obtained by solving (3.5).

The numerically solution of (3.5) is computed by applying a fixed point type iteration. Let $\bar{S}^{(m)} = (\bar{S}_1^{(m)}, \ldots, \bar{S}_{N_K}^{(m)})$ be the m-th iterate. We can then use the values $\bar{S}_i^{(m)}$ to construct the global downscaling function $\psi(\bar{S}_1^{(m)}, \ldots, \bar{S}_{N_K}^{(m)})$. Finally we perform the following updating procedure to find the mean values $\bar{S}_i^{(m+1)}$:

$$\begin{cases} F_i^{(m)}(\overline{S}^{(m)}) = \gamma\int_{K_i}\psi(\overline{S}^{(m)}) - \gamma|K_i|\bar{S}_i^n + \int_{\partial K_i}\lambda(\psi(\overline{S}^{(m)})v\cdot n \\ \bar{S}^{(m+1)} = \bar{S}^{(m)} - F^{(m)} \end{cases} \tag{3.6}$$

with a given initial guess $\bar{S}^{(0)}$. Assume that the above iteration converges at the M-th iteration. We will set $\bar{S}^{n+1} = \bar{S}^{(M)}$. We remark that each iteration of the above updates requires the construction of the global downscaling function $\psi(\bar{S}_1^{(m)}, \ldots, \bar{S}_{N_K}^{(m)})$, which requires the solution of many local nonlinear problems (3.2)–(3.4). With the use of the proposal deep learning approach, which will be presented in Section 3.3, this step becomes much more efficient.

3.2. Computation of local problems. We will use Newton's method to solve the local problem (3.2)–(3.4). We recall that the value $\bar{S}^n$ is fixed. For a given $\{\bar{S}_\beta\}$, the problem (3.2)–(3.4) defines a function $\tilde{\psi}$ on K_α^{++}. We note that the value $\{\bar{S}_\beta\}$ can be considered as a piecewise constant function, denoted as $\bar{S}$, defined on the coarse grid supported in K_α^{++}. Thus, the system (3.2)–(3.4) defines a map which takes a piecewise constant function $\{\bar{S}_\beta\}$ as input and returns a fine scale function $\tilde{\psi}$. In our numerical simulations, we take $\tilde{\psi}$ to be

ETNA
Kent State University and
Johann Radon Institute (RICAM)

piecewise constant function on the fine grid. In order to ensure the constraint (3.4) is satisfied, we will introduce an additional variable μ. This variable μ is a piecewise constant function on the coarse grid whose support is K_α^{++}. In addition, the system (3.2)–(3.4) will be solved on the fine grid by using the upwind finite volume scheme. In the following, we will present the mathematical details.

The equation (3.2) will be discretized by the upwind finite volume scheme. Accordingly, for each fine grid element $\tau_j \in \mathcal{T}_h$, we will apply the upwind finite volume scheme to (3.2). This motivates us to define the following two operators:

$$\begin{cases} F_{\text{eqn},j}(\tilde{\psi},\mu) = \gamma|\tau_j|\tilde{\psi}_j + \int_{\partial\tau_j} \lambda(\tilde{\psi}_{\text{up}}) v\cdot n - \int_{\tau_j} \bar{S}^n - \int_{\tau_j}\mu, & \forall \tau_j \subset K_\alpha^{++},\\ F_{\text{mean},\beta}(\tilde{\psi},\mu) = -|\tau_j|\sum_{\tau_j\subset K_\beta}\tilde{\psi}_j + |K_\beta|\bar{S}_\beta^{(m)}, & \forall K_\beta \subset K_\alpha^{+},\end{cases}$$

where $\tilde{\psi}_{\text{up}}$ denotes the upwind flux. We use the notation F_{eqn} to denote the vector whose j-th component is $F_{\text{eqn},j}$, and use the notation F_{mean} to denote the vector whose β-th component is $F_{\text{mean},\beta}$. We remark that $F_{\text{eqn},j}$ gives the discretization of (3.2) on the fine grid cell $\tau_j \in \mathcal{T}_h$, and $F_{\text{mean},\beta}$ corresponds to (3.4) on the coarse element K_β. Next, we define

$$F(\tilde{\psi},\mu) = \begin{cases} F_{\text{eqn}}(\tilde{\psi},\mu)\\ F_{\text{mean}}(\tilde{\psi},\mu).\end{cases}$$

The goal is to apply Newton's method to solve $F(\tilde{\psi},\mu)=0$, which gives (3.2) and (3.4). The boundary condition (3.3) is imposed by the numerical flux $\tilde{\psi}_{\text{up}}$.

In order to apply Newton's method, we need the Jacobian matrix J of F, which is given by

$$J(\tilde{\psi},\mu) = \begin{bmatrix} \frac{\partial F_{\text{eqn}}}{\partial\tilde{\psi}} & \frac{\partial F_{\text{eqn}}}{\partial\mu}\\ \frac{\partial F_{\text{mean}}}{\partial\tilde{\psi}} & \frac{\partial F_{\text{mean}}}{\partial\mu}\end{bmatrix} = \begin{bmatrix} \frac{\partial F_{\text{eqn}}}{\partial\tilde{\psi}} & B\\ B^T & 0\end{bmatrix},$$

where

$$\frac{\partial F_{\text{eqn},j}}{\partial\tilde{\psi}_i} = \gamma|\tau_j|\delta_{i,j} + \int_{\partial\tau_j}\lambda'(\tilde{\psi}_{\text{up}})\, v\cdot n\,\delta_{i,j} \quad\text{and}\quad B_{\beta,j} = -|\tau_j|\delta_{\beta,j}^K.$$

where $\delta_{i,j}$ is the delta function and

$$\delta_{\beta,j}^K = \begin{cases} 1 & \text{if } \tau_j\subset K_\beta,\\ 0 & \text{otherwise.}\end{cases}$$

Newton's method is then given by

$$\begin{pmatrix}\tilde{\psi}^{(n+1)}\\ \mu^{(n+1)}\end{pmatrix} = \begin{pmatrix}\tilde{\psi}^{(n)}\\ \mu^{(n)}\end{pmatrix} - J(\tilde{\psi}^{(n)},\mu^{(n)})^{-1}F(\tilde{\psi}^{(n)},\mu^{(n)}) \tag{3.7}$$

with a suitable initial guess $(\tilde{\psi}^{(0)},\mu^{(0)})$. We will stop the iteration if the value $F(\tilde{\psi}^{(n)},\mu^{(n)})$ is sufficiently small. This procedure yields the solution of (3.2)–(3.4).

3.3. Deep neural network model for local downscaling functions. We let the function $\mathcal{N}$ be a network of L layers, x be the input and y be the corresponding output. We write

$$\mathcal{N}(x;\theta) = \sigma\left(W_L\sigma\left(\cdots\sigma\left(W_2\sigma\left(W_1x+b_1\right)+b_2\right)\cdots\right)+b_L\right)$$

where $\theta := (W_1, \dots, W_L, b_1, \dots, b_L)$, W_i are the weight matrices, b_i are the bias vectors, and σ is the activation function. A neural network describes the connection of a collection of nodes (neurons) sitting in successive layers. The output neurons in each layer are simultaneously the input neurons of the next layer. The data propagate from the input layer to the output layer through hidden layers. The neurons can be switched on or off as the input is propagated forward through the network.

Suppose we are given a collection of data samples $\{(x_j, y_j)\}$. The goal is then to find θ^* by solving an optimization problem

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N_s} \sum_{j=1}^{N_s} \left\| y_j - \mathcal{N}(x_j; \theta) \right\|_2^2$$

where N_s is the number of the samples. The function $\frac{1}{N_s} \sum_{j=1}^{N_s} \|\mathbf{y}_j - \mathcal{N}(\mathbf{x}_j; \theta)\|_2^2$ is known as the loss function. One needs to select suitable numbers of layers and neurons in each layer, the activation function, the loss function, and the optimizers for the network.

We will use a deep neural network $\mathcal{N}$ to model the process of constructing downscaling functions. Recall that the local downscaling function is defined by (3.2)–(3.4). We note that there are two sources of inputs. The first input contains the values $\{\bar{S}_i^n\}$ from the time step n. The second input contains the values $\{\bar{S}_\beta^{(m)}\}$ in the fixed point iteration (3.6). Moreover, we observe that the required outputs are the values of the global downscaling function ψ restricted to the coarse grid edges, since only these values are used in the numerical scheme (3.5).

Therefore, we use the following choices in our deep neural network:

- Input: $x = \{\bar{S}^{(m)}, \bar{S}^n\}$.
- Output: $y = \psi|_{\cup_j E_j}$, where E_j is the j-th coarse edge. This is the restriction of ψ on all coarse edges.
- Data: $N_s = 5000$, that is 5000 training data samples are used.
- Standard loss function: $\frac{1}{N_s} \sum_{j=1}^{N_s} \|y_j - \mathcal{N}(x_j; \theta)\|_2^2$.
- Activation function: The popular ReLU function (the rectified linear unit activation function) is a common choice for activation function in training deep neural network architectures.

As for the input of the network, we use $x = \{\bar{S}^{(m)}, \bar{S}^n\}$, which are the vectors containing the approximate mean value $\bar{S}^{(m)}$ from the current iteration in (3.6) and the mean value $\bar{S}^n$ at the n-th time step of the scheme (3.5). The input x is a random vector such that each entry is ranged from 0 to 1. The range is based on the range of the initial condition. Since the range of the initial condition will affect the range of the mean value of the next time step, we choose the range of random input vector x to be $[0, 1]$.

The corresponding output data are $y = \psi|_{\cup_j E_j}$, which contains the values of the global downscaling function on each coarse edge. In order to obtain the data y, we use randomly generated input x and solve the local system (3.2)–(3.4) together with $\psi = \sum \chi_\alpha \psi_\alpha$ to obtain the global downscaling function ψ, which will give the output data y.

In between the input and output layer, we test on 2 hidden layers with the number of neurons ranging from $4N_K$ to $8N_K$ in each hidden layer, where N_K is the number of the coarse elements. We note that the size of input vector x is $2N_K$ and the size of output vector y is $n_E N_E$, where N_E is the number of coarse grid edges and n_E is the number of fine grid cells on the coarse edge E. In the training, there are $N_s = 5000$ data samples of $(\mathbf{x}_j, \mathbf{y}_j)$ collected.

In between layers, we need the activation function. The ReLU function (rectified linear unit activation function) is a popular choice for activation function in training deep neural network architectures. We will use the standard loss function: $\frac{1}{N_s} \sum_{j=1}^{N_s} \|y_j - \mathcal{N}(x_j; \theta)\|_2^2$. As for the training optimizer, we use AdaMax, which is a stochastic gradient descent (SGD) type algorithm well-suited for high-dimensional parameter space, in minimizing the loss function. We remark that the network is a fully-connected network. In particular, all the inputs x and outputs y are connected. Also, a more systematic strategy to select the hyper parameters will be investigated in the future.

3.4. Stacked local neural network model. As mentioned in the above section, the neural network $\mathcal{N}$ is fully connected. For small numbers of input and output, this fully connected network works well, as it can be seen in the first three examples in our numerical tests. For larger scale problems, training a fully connected network is challenging due to the large number of parameters θ. Because of this, we design a local neural network model, which is illustrated in Figure 3.1. Some numerical performances will be shown in Section 4.4.

Our local upscaling model is the key to the design of the stacked local neural network architecture. The input of the network is the set of all cell averages over the whole computational domain as before. In the first layer of the local network model, we will form subsets of these input cell averages. In particular, each subset contains cell averages for an oversampling region K^{++}. Each set of these cell averages will be the input of a local fully connected neural network, as shown in the middle part of Figure 3.1. The action of the local neural network is to provide an approximate solution of (3.2)–(3.4). Accordingly, the output of the local neural network is the local downscaling function restricted to the coarse grid edges. In the last layer of the stacked local neural network model, the outputs of the local networks are combined using partition of unity functions. The output is the global downscaling function restricted to the coarse grid edges. We remark that, due to the local connectivity, there are much fewer network parameters to be trained, and we observe much better performance compared with a global fully connected network. We also remark that weights of the first and the last layers of the networks are fixed. We will give a quantitative discussion in Section 4.4.

4. Numerical examples.

In this section, we will present some numerical examples to demonstrate the performance of our proposed deep learning based nonlinear upscaling method. In our simulations, we will take $\lambda(S) = S^2$ for the first three examples and $\lambda(S) = \frac{S^2}{S^2+(1-S)^2}$ for the last one. Moreover, the oversampling region K^+ is chosen by enlarging the coarse cell K by one coarse grid layer, and the double oversampling region K^{++} is chosen by enlarging K^+ by one more coarse grid layer. This choice of oversampling regions is motivated by the coarse time step size and the velocity of propagation. We will show the performance by using our upscaling method without using deep learning in the construction of the local downscaling functions. In this case, the error will only come from the approximation of the upscaling method. In addition, we will present the performance of our method with the use of deep neural network to approximate the local downscaling functions. In this case, the error in this deep neural network approximation will be added to the solutions. However, the computational efficiency improves significantly. In both cases, we observe that our proposed upscaling method is able to provide accurate numerical approximations. In our simulations, reference solutions are obtained by the standard upwind finite volume scheme on the fine grid and the use of Newton's method as nonlinear solver. All the experiments are preformed on a server with the Ubuntu 18.04 system with a 32-core 2.2 GHz Xeon CPU, 200 GB RAM, and 4 NVIDIA Tesla V100 GPUs with 32 GB each. The neural network model is build up by Python Deep Learning Library TensorFlow and other part of algorithm is speed up by using open-source array library Cupy accelerated with NVIDIA CUDA.

4.1. Example 1. For the first numerical example, we use the time step $\Delta t = 0.1$ and the constant velocity $v = (1, 1)$ to test our algorithm. As a result, the constant $\gamma = \frac{1}{\Delta t} = 10$. The coarse mesh is 20×20 and the fine mesh is 100×100. In this case, due to the finite speed of propagation, the size of the oversampling region K^+ is enough to capture the solution at the

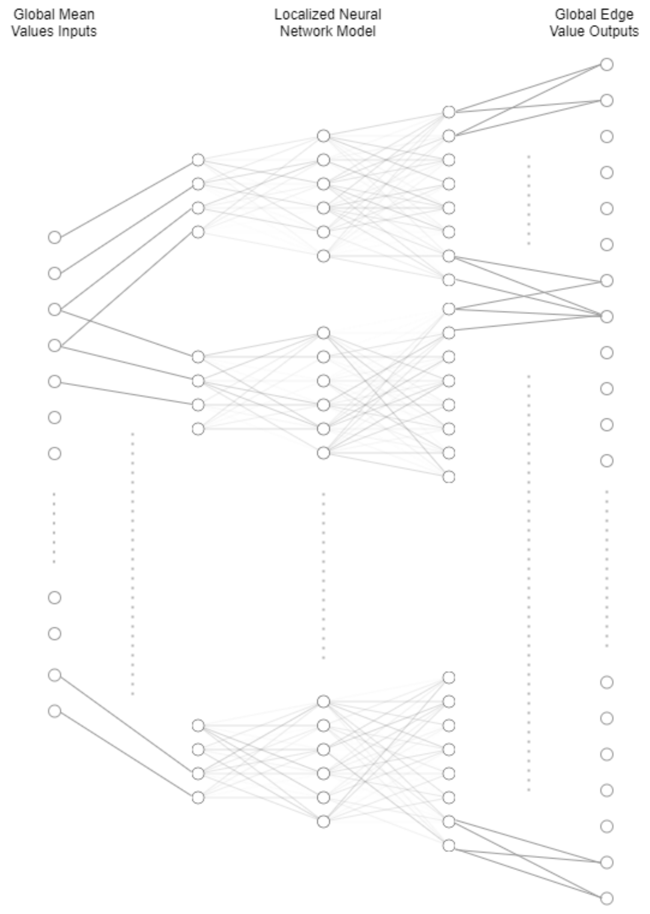

FIG. 3.1. *Stacked local neural network model.*

next time step from the solution at the previous time step originated in K. Also, we choose the initial condition as $S_0(x,y) = (1+\sin(2\pi x)\sin(2\pi y))/2$. Figure 4.1 displays a plot of S_0. The initial condition is bounded between 0 and 1. For the numerical computation of (3.5) to find the mean value $\bar{S}^{n+1}$ for the next time step, we use the fixed point type iteration (3.6) so we need a stopping criterion to stop the iteration. The fixed point type iteration (3.6) takes about 100 iterations to converge. In our simulations, we choose the condition $\|\bar{S}^{(m+1)} - \bar{S}^{(m)}\| \leq 0.0001$ as the stopping criterion. Also, the Newton's method to solve the local problem (3.2)–(3.4) takes about 10 iterations to converge. Furthermore, each training data needs approximately 10 s to generate, and the network needs about 15 to 30 minutes for the training process.

In this numerical example, we test the algorithm by running five time steps. Figure 4.2 displays the solutions for the first five time steps of our proposed scheme using neural network model. We also depict the average value of the reference solution on the coarse grid in this figure. From these results, we observe a very good agreement between these two solutions. We also observe that the numerical solution is bounded between 0 and 1, which holds for the reference solution. We remark that we did not impose any bound preserving properties in the training of our deep neural network. An improved network with this desirable property will be developed in future research.

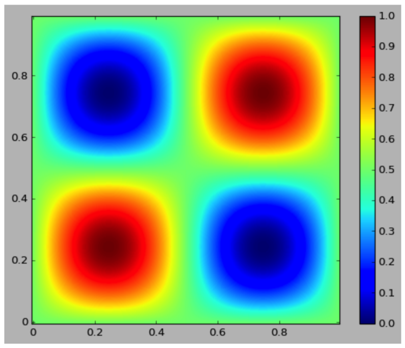

FIG. 4.1. *Initial condition for Example 1.*

TABLE 4.1
Relative root mean square error of the mean value for Example 1.

time step	relative error using neural network model	relative error using the upscaled model without neural network	relative error using the finite volume method
$t = 0.1$	0.04379	0.03210	0.06180
$t = 0.2$	0.04728	0.04070	0.08007
$t = 0.3$	0.05264	0.04574	0.09705
$t = 0.4$	0.05837	0.05227	0.10891
$t = 0.5$	0.06569	0.05887	0.11433

For the proposed neural network model, the relative error of the mean value at the first time step is 0.04379. The relative error of the mean value at the fifth time step is 0.06569. Table 4.1 lists the relative errors for the mean value of the solution using our neural network model. From the results in this table, we observe that our scheme exhibits a very good performance. We also observe that the relative error of the mean value rises slowly with increasing time. In terms of the computational efficiency, about 45 s were used for computing one time step for our neural network model. As a comparison, if we compute the downscaling functions without the use of neural networks, we need about 4500 s for the computation of one time step. Thereby most of the computational time is consumed on solving the local downscaling functions. We conclude that the use of deep neural network model can reduce the computational time without sacrificing the accuracy.

We also report the results using the downscaling functions without deep neural network in the construction of the proposed upscaling model. The relative error of the mean value at the first time step is 0.03210. The relative error of the mean value at the fifth time step is 0.05887. Table 4.1 shows the relative errors of the mean value when our method is applied without the use of deep neural network. The performance of solving the local downscaling functions numerically without the use of deep neural network and solving the local downscaling functions using neural network model are similar as can be seen from Table 4.1.

We next compare our scheme with a standard coarse-grid finite volume method with upwind flux. The relative error of the mean value at the first time step is 0.06180 and at the fifth time step 0.11433. Table 4.1 shows the relative errors of the mean value when a standard upwind finite volume method is applied on the coarse grid. We notice that the performance of our method with the use of deep neural network model is better than a simple application of a coarse-grid finite volume method.

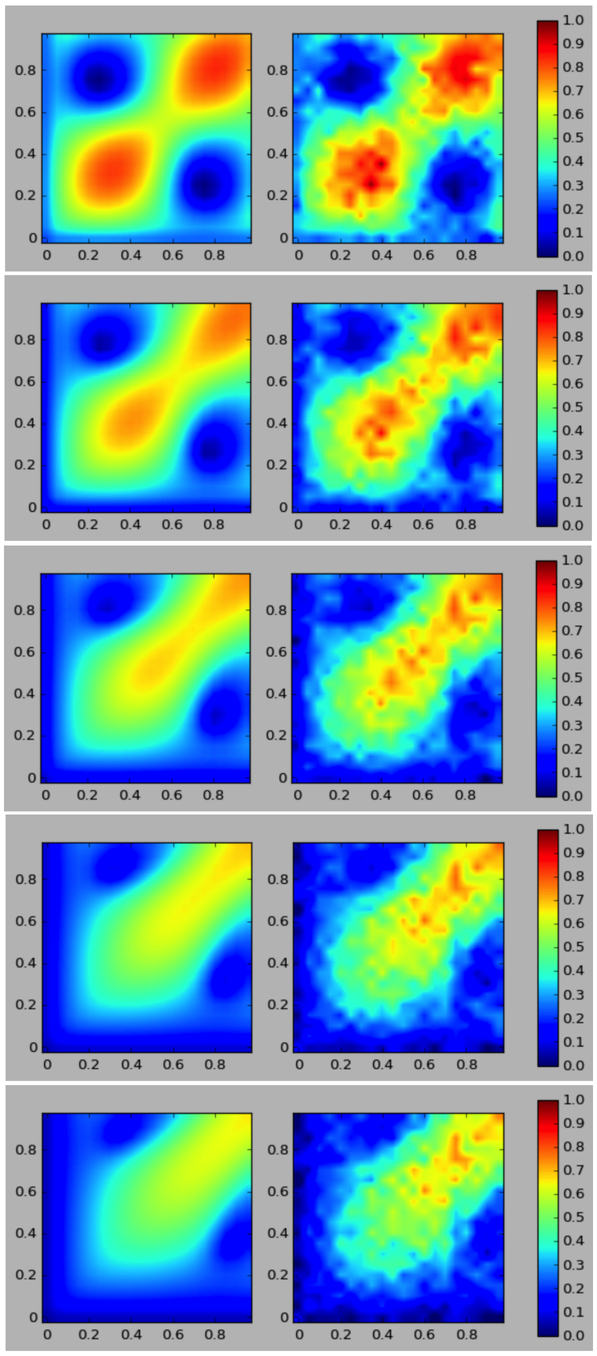

FIG. 4.2. *Solutions at the first 5 time steps* $t = 0.1, 0.2, \ldots, 0.5$ *for Example 1. Left: upscaled reference solution. Right: numerical solution using deep learning based upscaling.*

4.2. Example 2. For the second numerical example, we also use the time step $\Delta t = 0.1$. We use a velocity given by

$$v(x, y) = \big(1 + \sin^2(2\pi x)\sin(2\pi y), 1 + \sin(4\pi x)\cos(2\pi y)\big)$$

to test our algorithm. Figure 4.3 displays the plot of $v(x, y)$. The coarse mesh is also chosen as 20×20 and the fine mesh is 100×100. Since the velocity is bounded between 0 and 2, the oversampling region K^+ is able to keep all information originated from the coarse element K within one coarse time step. Also, we choose the same initial condition as before, $S_0(x, y) = (1 + \sin(2\pi x)\sin(2\pi y))/2$, and the same stopping criterion as in Example 1, $\|\bar{S}^{(m+1)} - \bar{S}^{(m)}\| \leq 0.0001$.

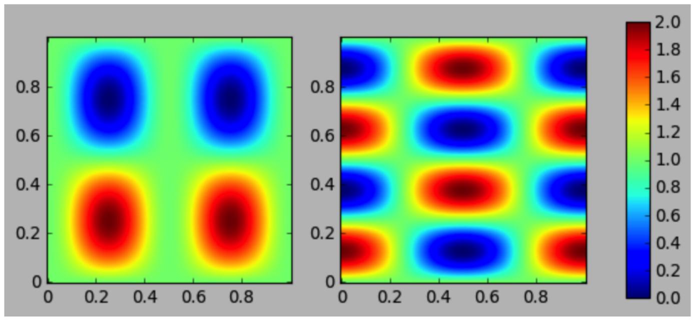

FIG. 4.3. *The velocity $v(x, y) = (v_1, v_2)$ for Example 2. The left figure is v_1 and the right figure is v_2.*

In this numerical example, we test the algorithm with 5 time steps. Figure 4.4 displays the numerical solutions for the first 5 time steps computed using our proposed scheme with deep neural network model. In the same figure, we also show the corresponding average values of the reference solution on the coarse grid. From these figures, we observe very good agreement between the numerical solution and the average reference solution.

TABLE 4.2
Relative root mean square error of the mean value for Example 2.

time step	relative error using neural network model	relative error using the upscaled model without neural network	relative error using the finite volume method
$t = 0.1$	0.04468	0.02904	0.06116
$t = 0.2$	0.04587	0.03579	0.07144
$t = 0.3$	0.04727	0.04223	0.08546
$t = 0.4$	0.04795	0.04829	0.09330
$t = 0.5$	0.05655	0.05499	0.09634

For our proposed scheme with the use of deep neural network model, the relative error of the numerical solution at the first time step is 0.04468 and at the fifth time step 0.05655. Table 4.2 shows in the second column the relative errors of the numerical solution computed using our proposed scheme with neural network model. We observe a very good performance. Also the computational time is about 50 s for the computation of one time step of using the proposed neural network model. As a comparison, if we compute the downscaling functions without neural networks, we need about 5500 s for computation of one time step.

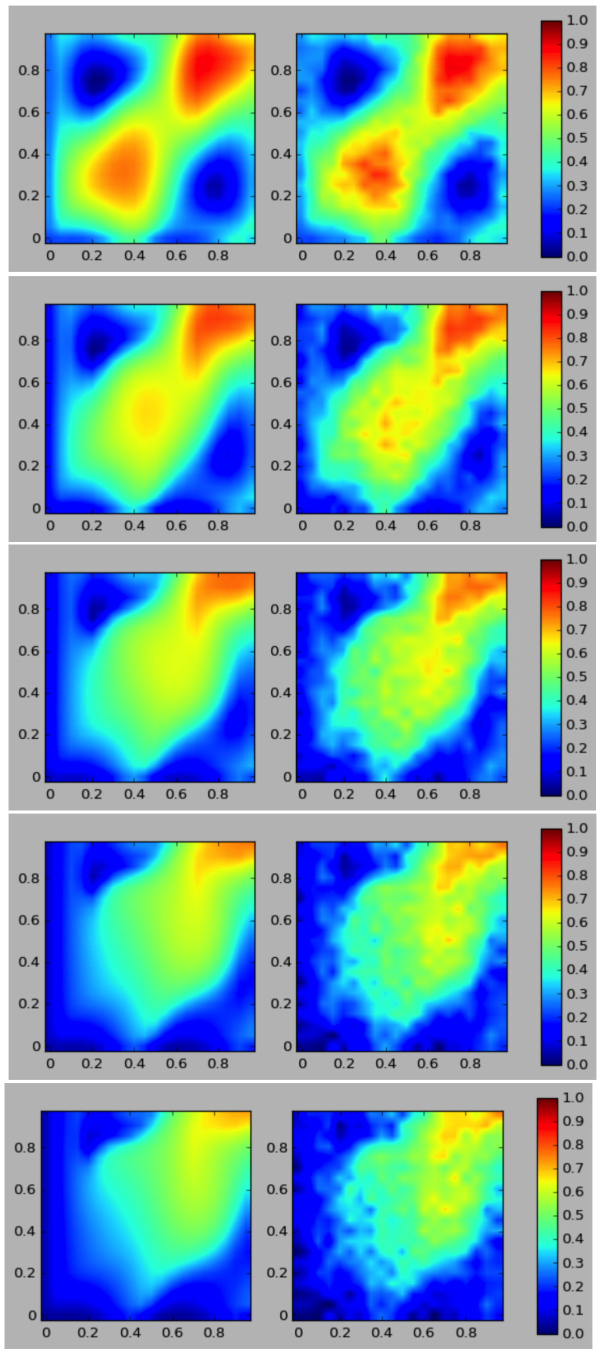

FIG. 4.4. *Solutions at the first 5 time steps* $t = 0.1, 0.2, \ldots, 0.5$ *for Example 2. Left: upscaled reference solution. Right: numerical solution using deep learning based upscaling.*

As a comparison, we present the results for our scheme without neural network. The relative error of the numerical solution at the first time step is 0.02904 and at the fifth 0.05499. Table 4.2 shows in the third column the relative errors of the numerical solution computed using our scheme without neural network. That is, we solve the local problem (3.2)–(3.4) directly. We observe that the performance of using neural network and without the use of neural network are similar. This shows that our proposed deep neural network can increase the efficiency of the method without losing accuracy.

Again, we show the performance of using a standard coarse-grid finite volume method with upwind flux. The relative error of the numerical solution at the first time step is 0.06116 and at the fifth 0.09634. Table 4.2 shows in the last column the relative errors of the numerical solution computed using finite volume method. Comparing the results in Table 4.2, we observe that our proposed nonlinear upscaling is able to give better numerical solutions.

4.3. Example 3. In this example, we test a more complicated initial condition,

$$S_0(x, y) = (1 + \sin(4\pi x)\sin(8\pi y))/2.$$

Figure 4.5 displays the figure of $S_0(x, y)$. This new initial condition is also bounded between 0 and 1. For this third numerical example, we will also consider the time step as $\Delta t = 0.05$ and use the second example's velocity given by

$$v(x, y) = (1 + \sin^2(2\pi x)\sin(2\pi y), 1 + \sin(4\pi x)\cos(2\pi y))$$

to test our algorithm. The coarse and the fine meshes are again 20×20 and 160×160 respectively. Same as before, we choose the condition $\|\bar{S}^{(m+1)} - \bar{S}^{(m)}\| \leq 0.0001$ as the stopping criterion.

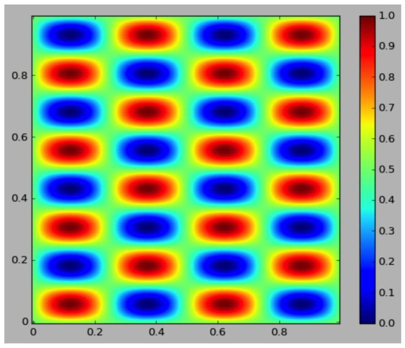

FIG. 4.5. *Initial condition for Example 3.*

Figure 4.6 displays the learning curve of neural network for Example 3. We take about 5000 training data into the neural network and about 1000 validation data to monitor the learning process of the neural network. Thanks to enough training data, the validation loss is only slightly higher than the loss of training data.

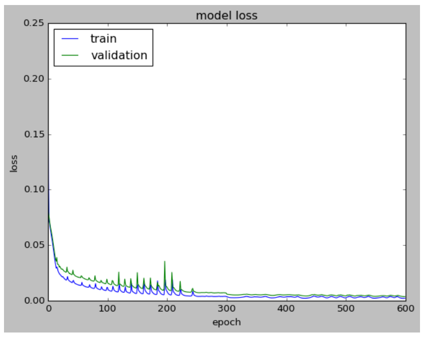

FIG. 4.6. *Learning curve of neural network for Example 3.*

In this numerical example, we test the algorithm with 10 time steps. Figure 4.7 show the first 10 time steps computed using our proposed scheme with deep neural network model. The corresponding average values of the reference solution on the coarse grid are shown in the same figure. We observe that these two solutions are in very good agreement.

TABLE 4.3
Relative root mean square error of the mean value for Example 3.

time step	relative error using neural network model	relative error using the upscaled model without neural network	relative error using the finite volume method
$t = 0.05$	0.02351	0.01689	0.09954
$t = 0.10$	0.02734	0.01971	0.12713
$t = 0.15$	0.03157	0.02378	0.13426
$t = 0.20$	0.03408	0.02654	0.13236
$t = 0.25$	0.03669	0.02903	0.12727
$t = 0.30$	0.03862	0.03116	0.12182
$t = 0.35$	0.04108	0.03304	0.11698
$t = 0.40$	0.04404	0.03464	0.11338
$t = 0.45$	0.04720	0.03602	0.11118
$t = 0.50$	0.05050	0.03737	0.10990

For the proposed neural network model, the relative error of the numerical solution of first time step is 0.02351 and of the tenth time step is 0.05050. Table 4.3 shows the relative errors of the numerical solution of using our proposed upscaling scheme with deep neural network model. We also observe that the relative error of the numerical solution rises slowly with increasing t. The time used here is about 70 s for computation of one time step of using neural network model. As a comparison, if we compute the local downscaling functions without the use of deep neural network, we need about 6000 s for computation of one time step. Since the velocity of Examples 2 and 3 is the same, we only need to use the same neural network model from Example 2 and apply for this new initial condition.

As a comparison, we report the results using the downscaling functions without neural network in the third column of Table 4.3. The relative error of the numerical solution at the first time step is 0.01689 and 0.03737 at the tenth time step. The performance of solving the local downscaling functions numerically without the use of deep neural network and solving the local downscaling functions using neural network model are similar.

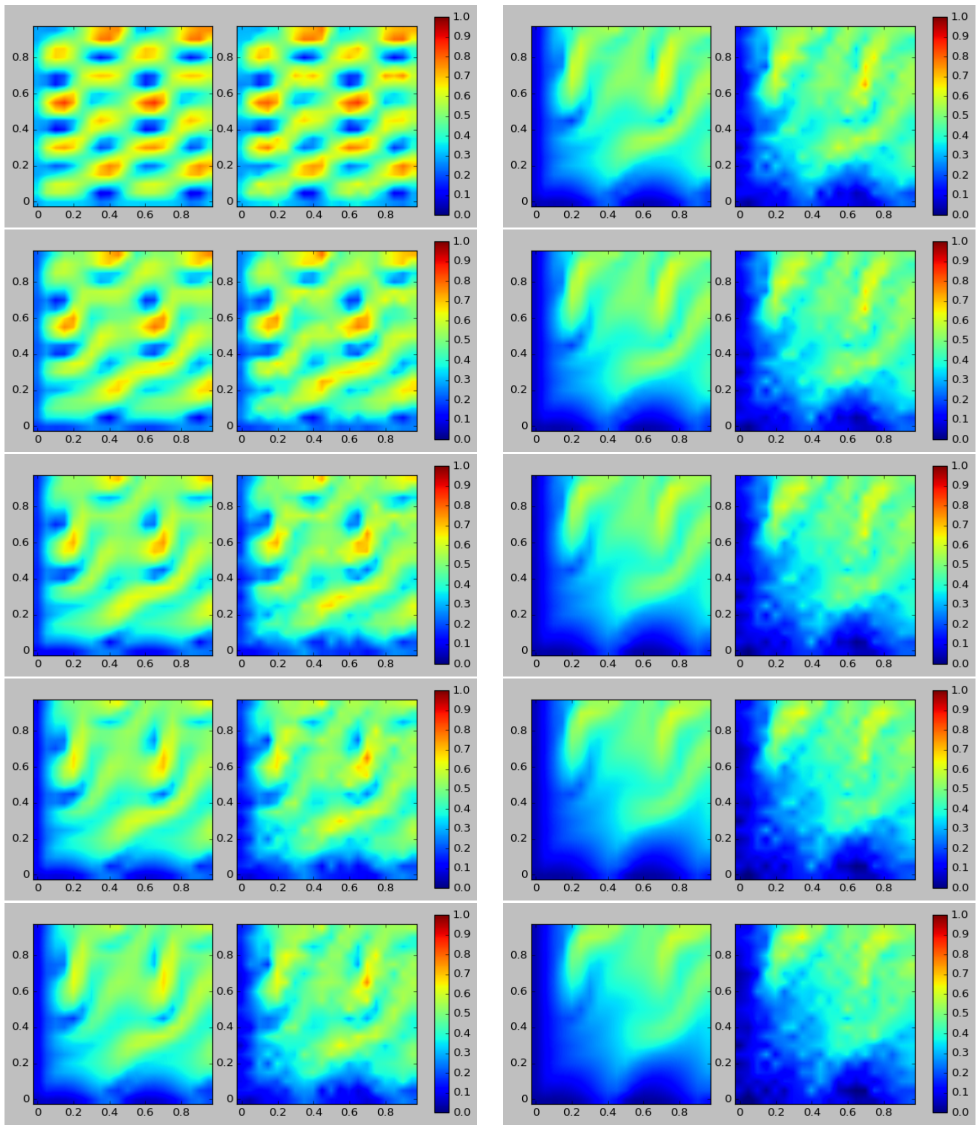

FIG. 4.7. *Solutions at the first 5 time steps(left) and last 5 time steps(right) for Example 3. First and third columns: upscaled reference solution. Second and Forth columns: upscaling numerical solution using deep learning based upscaling.*

The final comparison is the standard coarse-grid finite volume method with upwind flux. The relative error of the numerical solution at the first time step is 0.09954 and 0.10990 at tenth time step. Table 4.3 shows the relative errors of the numerical solution computed using the finite volume method in the last column. We notice that our proposed nonlinear upscaling is able to achieve better numerical solutions.

4.4. Example 4. In our final example, we use another initial condition

$$S_0(x,y) = \begin{cases} 1, & \text{if } x \in [0.01875, 0.0625], \\ 0.7, & \text{if } x \in [0.0625, 0.5], \\ 0.3, & \text{if } x \in [0.5, 1], \\ 0, & \text{otherwise.} \end{cases}$$

Figure 4.8 displays the image of $S_0(x,y)$. This new initial condition is also bounded between 0 and 1. For this numerical example, we also consider the time step $\Delta t = 0.05$ and use the second example's velocity given by

$$v(x,y) = \big(1 + \sin^2(2\pi x)\sin(2\pi y), 1 + \sin(4\pi x)\cos(2\pi y)\big)$$

to test our algorithm. We choose a new nonlinear function

$$\lambda(S) = \frac{S^2}{S^2 + (1-S)^2}\,.$$

The coarse and the fine meshes are again 20×20 and 160×160 respectively. Same as before, we choose the condition $\|\bar{S}^{(m+1)} - \bar{S}^{(m)}\| \le 0.0001$ as the stopping criterion. In this example, we will use the stacked local neural network model presented in Section 3.4 to show the performance of this model.

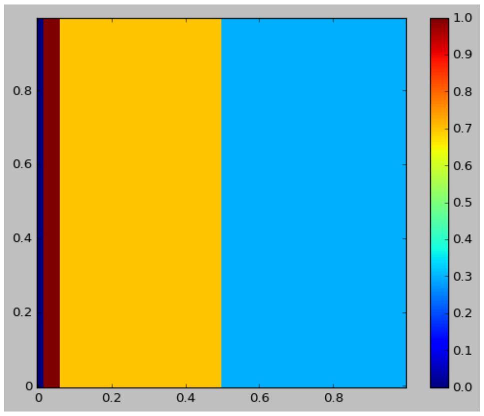

FIG. 4.8. *Initial condition for Example 4.*

Figure 4.9 displays the learning curve of fully connected and stacked local neural networks for Example 4. We take about 5000 training data into the neural network and about 1000 validation data to monitor the learning process of the neural network. From the learning curve, we can observe that the stacked local neural network has a more smooth and faster convergence rate than fully connected neural network. This is because the local property reduces a lot of irrelevant network parameters and hence enhances the convergence rate. Moreover, we observed approximately 50% reduction in the training time.

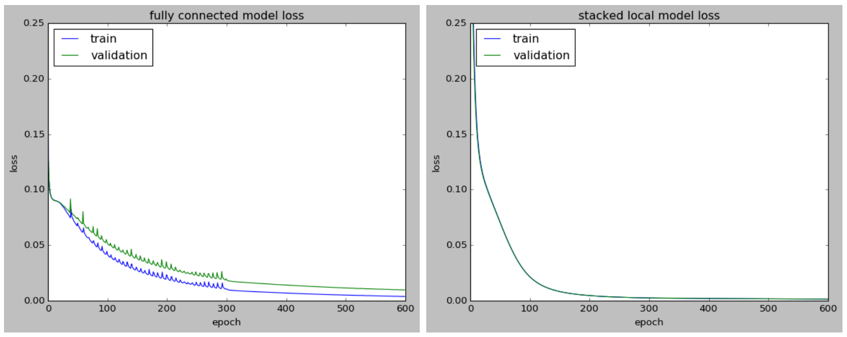

FIG. 4.9. *Learning curve of fully connected and stacked local neural network for Example 4.*

TABLE 4.4
Relative root mean square error of the mean value for Example 4.

time step	stacked local neural network model	fully connected neural network	upscaled model without neural network	finite volume method
$t = 0.05$	0.03423	0.03200	0.01515	0.07567
$t = 0.10$	0.03325	0.04250	0.02105	0.08525
$t = 0.15$	0.03390	0.05174	0.02245	0.09121
$t = 0.20$	0.02943	0.05549	0.01983	0.09727
$t = 0.25$	0.02670	0.05635	0.02211	0.10220
$t = 0.30$	0.02592	0.05996	0.02338	0.10435
$t = 0.35$	0.02574	0.06624	0.02669	0.10366
$t = 0.40$	0.02614	0.07303	0.03097	0.10206
$t = 0.45$	0.02698	0.07935	0.03614	0.10124
$t = 0.50$	0.02812	0.08559	0.04189	0.10133

We also test the algorithm with 10 time steps in this final example. Figure 4.10 displays the first 10 time no steps computed using our proposed scheme with deep neural network model. The corresponding average values of the reference solution on the coarse grid are shown in the same figure. We observe that these two solutions have very good agreement.

Table 4.4 displays the relative errors of the numerical solution of using our proposed upscaling scheme with stacked local neural network model, using fully connected neural network model and standard coarse-grid finite volume method with upwind flux. We can observe that the first step and final step of relative error of using neural network model are 0.03423 and 0.02812. Also, the relative error of the numerical solution rises slowly when time increases. The stacked local neural network and fully connected neural network model use the same number of training data. The number of weights and bias of the stacked local neural network is about 600,000 and that of fully connected neural network is about 30,000,000 which is 50 times of stacked local neural network. This shows that the memory usage of stacked local neural network is much less than the original fully connected one. Also, the number of epoch need for convergence and relative error of stacked local neural network are smaller than fully connected one. The final comparison is the standard coarse-grid finite volume method with upwind flux. The relative error of the numerical solution at the first time step is 0.10133. We notice that our proposed nonlinear upscaling is able to give better numerical solutions.

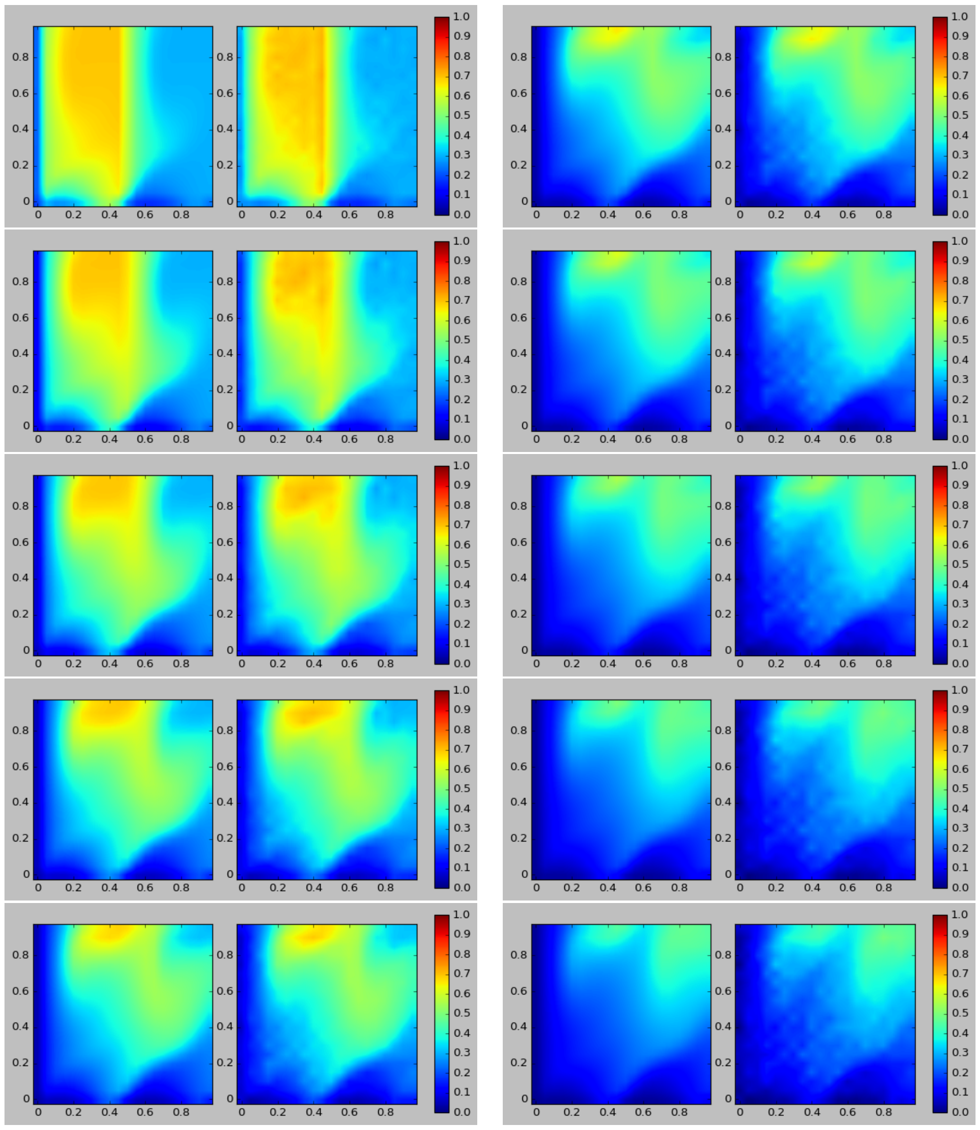

FIG. 4.10. *Solutions at the first 5 time steps(left) and last 5 time steps(right) for Example 4. First and third columns: upscaled reference solution. Second and Forth columns: upscaling numerical solution using stacked local neural network.*

5. Conclusion. In this paper, we develop deep learning based nonlinear upscaling for nonlinear transport equations with heterogeneous velocity. The technique is based on the recently developed NLMC method. To construct the coarse scale model, local downscaling operations are used to reconstruct fine scale information from coarse grid averages. Due to the nonlinearity of the problem, these local solutions are expensive to compute. To overcome this bottleneck, we propose the use of deep neural network to approximate this procedure. Since a reduced model is used, the learning process is robust, and the resulting neural network gives accurate approximations. With the use of our upscaling model, we only need to learn some local maps. This means that the number of parameters to be trained is much less compared with the learning of a global map. It is this local learning idea that helps the approximation using the proposed deep neural network method. The proposed approach shows promising numerical results.

Our proposed approach is based on the training of neural networks in oversampling regions. This locality framework has potentials for future developments. In future work, we plan to design local neural networks that allow the use of media properties, physical parameters in the equations and local mesh configuration as inputs. In this case, we expect that only one single neural network is enough, and this network can be applied to a wider class of problems and to many local oversampling regions with various geometries.

Acknowledgements. Eric Chung's work is partially supported by the Hong Kong RGC General Research Fund (Project numbers 14304719 and 14302018) and the CUHK Faculty of Science Direct Grant 2020-21. The first two authors thank NVIDIA – NVAITC for providing computing resources.

REFERENCES

[1] A. ABDULLE AND Y. BAI, *Adaptive reduced basis finite element heterogeneous multiscale method*, Comput. Methods Appl. Mech. Engrg., 257 (2013), pp. 203–220.

[2] G. ALLAIRE AND R. BRIZZI, *A multiscale finite element method for numerical homogenization*, Multiscale Model. Simul., 4 (2005), pp. 790–812.

[3] T. ARBOGAST, *Implementation of a locally conservative numerical subgrid upscaling scheme for two-phase Darcy flow*, Comput. Geosci, 6 (2002), pp. 453–481.

[4] T. ARBOGAST, G. PENCHEVA, M. WHEELER, AND I. YOTOV, *A multiscale mortar mixed finite element method*, Multiscale Model. Simul., 6 (2007), pp. 319–346.

[5] G. BARENBLATT, I. P. ZHELTOV, AND I. KOCHINA, *Basic concepts in the theory of seepage of homogeneous liquids in fissured rocks [strata]*, J. Appl. Math. Mech., 24 (1960), pp. 1286–1303.

[6] J. BARKER AND S. THIBEAU, *A critical review of the use of pseudorelative permeabilities for upscaling*, SPE Reserv. Eng., 12 (1997), pp. 138–143.

[7] D. L. BROWN AND D. PETERSEIM, *A multiscale method for porous microstructures*, arXiv preprint arXiv:1411.1944, 2014. https://arxiv.org/abs/1411.1944

[8] Y. CHEN, L. DURLOFSKY, M. GERRITSEN, AND X. WEN, *A coupled local-global upscaling approach for simulating flow in highly heterogeneous formations*, Adv. Water Resour., 26 (2003), pp. 1041–1060.

[9] S. W. CHEUNG, E. T. CHUNG, Y. EFENDIEV, E. GILDIN, Y. WANG, AND J. ZHANG, *Deep global model reduction learning in porous media flow simulation*, Comput. Geosci., 24 (2020), pp. 261–274.

[10] E. T. CHUNG AND Y. EFENDIEV, *Reduced-contrast approximations for high-contrast multiscale flow problems*, Multiscale Model. Simul., 8 (2010), pp. 1128–1153.

[11] E. T. CHUNG, Y. EFENDIEV, AND S. FU, *Generalized multiscale finite element method for elasticity equations*, Int. J. Geomath., 5(2) (2014), pp. 225–254.

[12] E. T. CHUNG, Y. EFENDIEV, AND W. T. LEUNG, *Generalized multiscale finite element method for wave propagation in heterogeneous media*, Multicale Model. Simul., 12 (2014), pp. 1691–1721.

[13] ———, *Constraint energy minimizing generalized multiscale finite element method in the mixed formulation*, Comput. Geosci., 22 (2018), pp. 677–693.

[14] ———, *Constraint energy minimizing generalized multiscale finite element method*, Comput. Methods Appl. Mech. Eng., 339 (2018), pp. 298–319.

[15] E. T. CHUNG, Y. EFENDIEV, W. T. LEUNG, M. VASILYEVA, AND Y. WANG, *Online adaptive local multiscale model reduction for heterogeneous problems in perforated domains*, Appl. Anal., 96 (2017), pp. 2002–2031.

[16] ———, *Fast online generalized multiscale finite element method using constraint energy minimization*, J. Comput. Phys., 355 (2018), pp. 450–463.

[17] ———, *Non-local multi-continua upscaling for flows in heterogeneous fractured media*, J. Comput. Phys., 372 (2018), pp. 22–34.

[18] E. T. CHUNG, Y. EFENDIEV, W. T. LEUNG, AND M. WHEELER, *Nonlinear nonlocal multicontinua upscaling framework and its applications*, Int. J. Multiscale Comput. Eng., 16 (2018), pp. 487–507.

[19] E. T. CHUNG, Y. EFENDIEV, AND G. LI, *An adaptive GMsFEM for high contrast flow problems*, J. Comput. Phys., 273 (2014), pp. 54–76.

[20] E. T. CHUNG AND W. T. LEUNG, *A sub-grid structure enhanced discontinuous galerkin method for multiscale diffusion and convection-diffusion problems*, Commun. Comput. Phys., 14 (2013), pp. 370–392.

[21] E. T. CHUNG, M. VASILYEVA, AND Y. WANG, *A conservative local multiscale model reduction technique for stokes flows in heterogeneous perforated domains*, J. Comput. Appl. Math., 321 (2017), pp. 389–405.

[22] M. DROHMANN, B. HAASDONK, AND M. OHLBERGER, *Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation*, SIAM J. Sci. Comput., 34 (2012), pp. A937–A969.
[23] W. E AND B. ENGQUIST, *Heterogeneous multiscale methods*, Comm. Math. Sci., 1 (2003), pp. 87–132.
[24] Y. EFENDIEV AND L. DURLOFSKY, *Numerical modeling of subgrid heterogeneity in two phase flow simulations*, Water Resour. Res., 38 (2002), Art. 1128, 11 pages.
[25] ———, *A generalized convection-diffusion model for subgrid transport in porous media*, Multiscale Model. Simul., 1 (2003), pp. 504–526.
[26] Y. EFENDIEV, J. GALVIS, AND T. HOU, *Generalized multiscale finite element methods (gmsfem)*, J. Comput. Phys., 251 (2013), pp. 116–135.
[27] Y. EFENDIEV, J. GALVIS, AND X. WU, *Multiscale finite element methods for high-contrast problems using local spectral basis functions*, J. Comput. Phys., 230 (2010), pp. 937–955.
[28] Y. EFENDIEV AND A. PANKOV, *Numerical homogenization of monotone elliptic operators*, Multiscale Model. Simul., 2 (2003), pp. 62–79.
[29] ———, *Numerical homogenization of nonlinear random parabolic operators*, Multiscale Model. Simul., 2 (2004), pp. 237–268.
[30] ———, *Homogenization of nonlinear random parabolic operators*, Adv. Differential Equations, 10 (2005), pp. 1235–1260.
[31] D. FAFALIS AND J. FISH, *Computational continua for linear elastic heterogeneous solids on unstructured finite element meshes*, Int. J. Numer. Methods Eng., 115 (2018), pp. 501–530.
[32] J. FISH AND W. CHEN, *Space–time multiscale model for wave propagation in heterogeneous media*, Comput. Methods Appl. Mech. Eng., 193 (2004), pp. 4837–4856.
[33] J. FISH AND R. FAN, *Mathematical homogenization of nonperiodic heterogeneous media subjected to large deformation transient loading*, Int. J. Numer. Methods Eng., 76 (2008), pp. 1044–1064.
[34] J. FISH AND S. KUZNETSOV, *Computational continua*, Int. J. Numer. Methods Eng., 84 (2010), pp. 774–802.
[35] J. FISH, K. SHEK, M. PANDHEERADI, AND M. S. SHEPHARD, *Computational plasticity for composite structures based on mathematical homogenization: theory and practice*, Comput. Methods Appl. Mech. Eng., 148 (1997), pp. 53–73.
[36] S. FU AND E. T. CHUNG, *A local-global multiscale mortar mixed finite element method for multiphase transport in heterogeneous media*, J. Comput. Phys., 399 (2019), Art. 108906, 13 pages.
[37] P. HENNING AND M. OHLBERGER, *The heterogeneous multiscale finite element method for elliptic homogenization problems in perforated domains*, Numer. Math., 113 (2009), pp. 601–629.
[38] J. KYTE AND D. BERRY, *New pseudofunctions to control numerical dispersion*, Soc. Pet. Eng. J., 15 (1975), pp. 269–276.
[39] S. H. LEE, M. LOUGH, AND C. JENSEN, *Hierarchical modeling of flow in naturally fractured formations with multiple length scales*, Water Resour. Res., 37 (2001), pp. 443–455.
[40] A.-M. MATACHE AND C. SCHWAB, *Two-scale fem for homogenization problems*, ESAIM Math. Model. Numer. Anal., 36 (2002), pp. 537–572.
[41] C. OSKAY AND J. FISH, *Eigendeformation-based reduced order homogenization for failure analysis of heterogeneous materials*, Comput. Methods Appl. Mech. Eng., 196 (2007), pp. 1216–1243.
[42] H. OWHADI AND L. ZHANG, *Metric-based upscaling*, Comm. Pure. Appl. Math., 60 (2007), pp. 675–723.
[43] G. PANASENKO, *Multicontinuum wave propagation in a laminated beam with contrasting stiffness and density of layers*, J. Math. Sci, (2018), pp. 1–13.
[44] A. PANKOV, *G-Convergence and Homogenization of Nonlinear Partial Differential Operators*, Kluwer, Dordrecht, 1997.
[45] M. PESZYŃSKA, M. WHEELER, AND I. YOTOV, *Mortar upscaling for multiphase flow in porous media*, Comput. Geosci., 6 (2002), pp. 73–100.
[46] M. VASILYEVA, W. T. LEUNG, E. T. CHUNG, Y. EFENDIEV, AND M. WHEELER, *Learning macroscopic parameters in nonlinear multiscale simulations using nonlocal multicontinua upscaling techniques*, J. Comput. Phys., (2020), Art. 109323, 22 pages.
[47] M. WANG, S. W. CHEUNG, W. T. LEUNG, E. T. CHUNG, Y. EFENDIEV, AND M. WHEELER, *Reduced-order deep learning for flow dynamics. the interplay between deep learning and model reduction*, J. Comput. Phys., 401 (2020), Art. 108939, 24 pages.
[48] Y. WANG, S. W. CHEUNG, E. T. CHUNG, Y. EFENDIEV, AND M. WANG, *Deep multiscale model learning*, J. Comput. Phys., 406 (2020), Art. 109071, 23 pages.
[49] J. WARREN AND P. J. ROOT, *The behavior of naturally fractured reservoirs*, Soc. Pet. Eng. J., 3 (1963), pp. 245–255.
[50] Z. YUAN AND J. FISH, *Multiple scale eigendeformation-based reduced order homogenization*, Comput. Methods Appl. Mech. Eng., 198 (2009), pp. 2016–2038.
[51] Z. ZHANG, E. T. CHUNG, Y. EFENDIEV, AND W. T. LEUNG, *Learning algorithms for coarsening uncertainty space and applications to multiscale simulations*, Mathematics, 8 (2020), Art. 720, 17 pages.

[52] L. ZHAO AND E. T. CHUNG, *An analysis of the NLMC upscaling method for high contrast problems*, J. Comput. Appl. Math., 367 (2020), Art. 112480, 15 pages.