Journal of Computational Physics 490 (2023) 112329

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A learned conservative semi-Lagrangian finite volume scheme for transport simulations

Yongsheng Chen [a], Wei Guo [b,*], Xinghui Zhong [a]

[a] School of Mathematical Sciences, Zhejiang University, Hangzhou, 310027, China
[b] Department of Mathematics and Statistics, Texas Tech University, Lubbock, TX, 70409, USA

ARTICLE INFO

Article history:
Received 14 February 2023
Received in revised form 12 June 2023
Accepted 25 June 2023
Available online 3 July 2023

Keywords:
Semi-Lagrangian
Machine learning
Neural network
Finite volume method
Transport equation

ABSTRACT

Semi-Lagrangian (SL) schemes are known as a major numerical tool for solving transport equations with many advantages and have been widely deployed in the fields of computational fluid dynamics, plasma physics modeling, numerical weather prediction, among others. In this work, we develop a novel machine learning-assisted approach to accelerate the conventional SL finite volume (FV) schemes. The proposed scheme avoids the expensive tracking of upstream cells but attempts to learn the SL discretization from the data by incorporating specific inductive biases in the neural network, significantly simplifying the algorithm implementation and leading to improved efficiency. In addition, the method delivers sharp shock transitions and a level of accuracy that would typically require a much finer grid with traditional transport solvers. Numerical tests demonstrate the effectiveness and efficiency of the proposed method.

1. Introduction

Transport phenomena are ubiquitous in nature and are characterized by a type of partial differential equations (PDEs), namely transport equations. Efficiently simulating transport equations represents a great challenge mainly due to the potential presence of multiple spatiotemporal scales and discontinuities in the solution structure. The last several decades have witnessed tremendous developments of numerical methods for transport equations, yielding numerous successful applications in both science and engineering. Among them, the semi-Lagrangian (SL) transport schemes attract lots of attention [11], and have become a major tool in simulating transport equations arising from numerical weather prediction [48,51] and plasma simulations [7,50]. Such an approach updates the mesh-based solution of a transport equation by tracing the characteristics, offering several computational benefits including high order accuracy and unconditional stability. Additionally, by incorporating an advanced discretization technique such as the weighted essentially non-oscillatory (WENO) method [21] or the discontinuous Galerkin method [9] into the SL framework, the resulting methods, see e.g., [41,42,49,43], attain outstanding performance for transport simulations. On the other hand, despite the effectiveness of the SL methodology, the direct numerical simulations may require an exceedingly large amount of computational resources when full resolution of fine-scale structures is of interest. Another difficulty associated with the SL approach is that tracking the characteristics accurately, including tracing and approximating upstream cells, is not only expensive in computational cost but also demands

* Corresponding author.
E-mail addresses: 22035024@zju.edu.cn (Y. Chen), weimath.guo@ttu.edu (W. Guo), zhongxh@zju.edu.cn (X. Zhong).

https://doi.org/10.1016/j.jcp.2023.112329
0021-9991/

complex algorithm implementation, especially in high dimensions. For instance, the SL method proposed in [25] implements a rather sophisticated search algorithm.

With the rapid advancements in computing power and machine learning (ML) software over the past few decades, integrating ML techniques in simulating PDEs has become a thriving area of research. One notable example is the physics informed neural networks (PINNs) [46,45], where the solution to the underlying PDE is parameterized with a neural network (NN) and trained using a physics informed loss function. The partial derivatives of the approximate solution are obtained via automatic differentiation [2]. As an effective alternative to the traditional numerical algorithms, PINNs have been widely deployed to solve complex problems in various fields [20,55,36,37,44,35,38]. A comprehensive review of the literature on PINNs was provided in [12]. We also mention one popular research direction of leveraging NNs to improve overall performance for the traditional numerical methods, and related works include the troubled-cell indicator in [47], weights estimation to enhance WENO scheme in [54], shock detection for WENO schemes in [52], and the total variation bounded constant estimation for limiters in [56].

Another group of NN-based methods for simulating PDEs is the so-called neural operator approach. Two widely recognized works are the DeepONet [33] which consists of two sub-networks and learns the nonlinear operators for identifying differential equations, and the Fourier neural operator (FNO) [30] which is designed based on parameterization of the integral kernel in the Fourier space and most closely resembles the reduced basis method. A performance comparison of these two methods was documented in [34]. Other works in this approach include but not limited to [22,31,3,53]. For time-dependent PDEs, the cell-averaged neural network (CANN) was developed in [40,5], which explores the approximation of the cell average difference of the solution between two consecutive time steps. Another different approach known as autoregressive methods was developed in [1,15,19,4], where the PDEs are simulated iteratively, resembling conventional numerical methods with time marching. Specifically, an autoregressive method predicts the solution at the next time step with a NN based on the current state, and this approach is closely related to flow map learning methods; see, e.g., [6,8,16,17].

In this paper, we consider the conservative SL finite volume (FV) formulation proposed in [25], and propose a novel ML-based approach to enhance the performance and accelerate the computation. The proposed methodology belongs to the category of autoregressive methods and is motivated by the recent ML-based discretization for PDEs [1,57,23]. In contrast to the neural operator methods, the methods proposed in [1,57,23] are designed under the classical finite difference or FV framework, while a key component is replaced by an ML technique for improved performance. In particular, instead of polynomials, such an approach employs NNs to learn an optimal discretization for approximating derivatives. Additionally, the architecture of such ML-based methods can benefit from Tensor Processing Units to accelerate computation [23], hence substantially reducing the computational cost.

Instead of the Eulerian method-of-line framework employed in [1,57,23], our method is built in the SL setting, and the discretization is learned using a convolutional NN (CNN) architecture [26,28] and incorporating specific inductive biases. In particular, we explicitly include a key quantity called the normalized shifts as part of the input in the NN, observing that such quantities contribute to computing the traditional SL FV discretization but in a rather complex manner, see Section 2 below. With high-resolution high-fidelity data, the proposed method can effectively learn and predict the SL discretization. Hence, by replacing the most expensive and complex component of the SL formulation with an ML-based approach, the proposed method avoids the explicit implementation in tracing upstream cells as in [25], achieving enhanced efficiency. In addition, as with the ML-based method using CNNs [57], the learned SL discretization with a coarse grid can accurately capture fine-scale features of the solution which often demands much higher grid resolution for a traditional discretization, leading to significant computational savings. We further propose to add a constraint layer to the NN to enforce exact mass conservation, which is a highly desired property for transport simulations and plays a vital role in long term accuracy and stability. The ML model can further enjoy improved generalization by incorporating such an additional level of inductive bias.

It follows from the dependence of the proposed methodology on the CNNs that the discretizations are restricted to a Cartesian grid. Different from the traditional SL formulations, the proposed ML-based method in this paper employs a set of fixed stencils for the SL reconstruction, which incurs the time step restriction due to the Courant-Friedrichs-Lewy (CFL) condition for numerical stability. However, it is observed in the numerical experiments in Section 3 that the CFL number can be set as large as 1.8 if a fixed stencil consisting of five cells is employed, which is often adequate and efficient for many applications. It is worth mentioning that, although the traditional SL approach is unconditionally stable in theory, choosing an exceedingly large CFL increases the complexity of the search algorithm for tracing and approximating upstream cells, especially in high dimensions, and also hinders parallel efficiency [25,18].

The rest of the paper is organized as follows. In Section 2, we first review the conventional SL FV scheme for one-dimensional (1D) and two-dimensional (2D) transport equations, and provide a sufficient condition for mass conservation. Then we introduce the proposed conservative ML-based SL FV method based on a CNN architecture. In Section 3, numerical results are presented to demonstrate the performance of the proposed method. The conclusion and future works are discussed in Section 4.

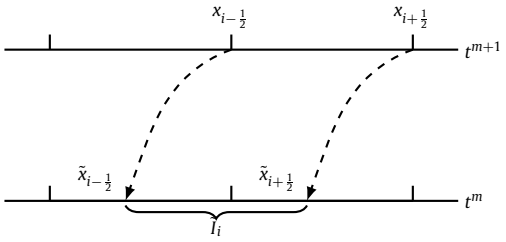

Fig. 2.1. Schematic illustration of the 1D SL FV scheme.

2. Algorithm

2.1. Semi-Lagrangian finite volume scheme

In this section, we review the classical SL FV scheme with remapping for linear transport equations; see e.g., [25,13]. We start with the following 1D equation in the conservative form

$$u_t + (a(x,t)u)_x = 0, \quad x \in \Omega, \tag{2.1}$$

where $a(x,t)$ is the velocity function. For simplicity, consider a uniform partition of the domain Ω with N cells, i.e., $\Omega = \bigcup_{i=1}^{N} I_i$, where $I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$. Denote the mesh size as $h = |I_i|$. The numerical solution $\{U_i^m\}_{i=1}^N$ approximates the cell averages at the time step t^m. The SL FV method updates the cell averages to next time step t^{m+1} by tracking the characteristics governed by the ordinary differential equation

$$\frac{dx(t)}{dt} = a(x(t),t). \tag{2.2}$$

The upstream cell $\tilde{I}_i = [\tilde{x}_{i-\frac{1}{2}}, \tilde{x}_{i+\frac{1}{2}}]$ of $I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ is defined by evolving (2.2) with final values $x(t^{m+1}) = x_{i\pm\frac{1}{2}}$ backward to t^m, as shown in Fig. 2.1.

Then the exact solution $u(x,t)$ satisfies

$$\int_{I_i} u(x,t^{m+1})dx = \int_{\tilde{I}_i} u(x,t^m)dx, \quad i = 1,\dots,N, \tag{2.3}$$

based on which we can formulate the SL FV method with remapping as follows. In each cell I_j, we reconstruct a polynomial $\phi_j(x)$ of degree K using the $K+1$ cell averages from its neighboring cells in the spirit of the standard FV methodology, aiming for $(K+1)$-th order accuracy. Then, the right hand side of (2.3) can be approximated by integrating the reconstructed polynomials in $\tilde{I}_i$. Meanwhile, it is noted that reconstructed polynomials are defined piece-wisely and hence are discontinuous across the cell interfaces. In addition, $\tilde{I}_i$ will not coincide with a background Eulerian cell in general. Therefore, the integrals must be computed in a subcell-by-subcell fashion. In particular, we denote by $\tilde{I}_{i,(j)}$ as the intersection of $\tilde{I}_i$ and the background Eulerian cell I_j. The collection of indices of the Eulerian cells which have non-empty intersection with the upstream cell $\tilde{I}_i$ is defined as

$$\mathcal{L}_i = \{j : \tilde{I}_i \cap I_j \neq \emptyset\}.$$

We write $\phi_j(x) = \sum_{k=0}^{K} c_j^{(k)} v_j^{(k)}(x)$, where $v_j^{(k)}(x)$ are the local polynomial basis in the cell I_j. At the discrete level, (2.3) becomes

$$U_i^{m+1}|I_i| = \sum_{j\in\mathcal{L}_i}\sum_{k=0}^{K} c_j^{(k)} \int_{\tilde{I}_{i,(j)}} v_j^{(k)}(x)dx \doteq \sum_{j\in\mathcal{L}_i}\sum_{k=0}^{K} c_j^{(k)}\omega_{i,j}^{(k)}, \tag{2.4}$$

yielding the SL FV scheme of order $K+1$. A key observation of the SL FV scheme is that $\{U_i^{m+1}\}$ are fully determined by cell averages at previous time step $\{U_i^m\}$ together with the normalized shifts

$$\xi_{i-\frac{1}{2}} := \frac{\tilde{x}_{i-\frac{1}{2}} - x_{i-\frac{1}{2}}}{h}, \quad i = 1,\dots,N,$$

which encode the geometry information of the upstream cells. In particular, the coefficients $\{\omega_{i,j}^{(k)}\}$ in (2.4) can be expressed using $\{\xi_{i-\frac{1}{2}}\}$. More details can be found in [42]. Therefore, the SL FV scheme (2.4) can be rewritten into the following formulation

$$U_i^{m+1} = \sum_{\ell \in \mathcal{S}_i^m} d_{i,\ell}^m U_\ell^m, \tag{2.5}$$

where $\mathcal{S}_i^m$ denotes the stencil employed to update U_i^{m+1}, and $\{d_{i,\ell}^m\}$ are the associated coefficients. It is worth mentioning that $\{d_{i,\ell}^m\}$ may depend on the numerical solution if a nonlinear reconstruction is used.

It can be shown that the SL FV scheme (2.4) conserves mass at the discrete level, i.e.,

$$\sum_i U_i^{m+1} = \sum_i U_i^m. \tag{2.6}$$

For an FV method in the form of (2.5), the condition for mass conservation is discussed in the following theorem.

Theorem 2.1. *An FV method in the form of (2.5) is mass conservative, i.e., satisfying (2.6), when*

$$\sum_{i \in \mathcal{E}_\ell^m} d_{i,\ell}^m = 1, \quad \forall l, \tag{2.7}$$

where $\mathcal{E}_\ell^m = \{i : \ell \in \mathcal{S}_i^m\}$ is the index set for a collection of cells $\{I_i\}$ for which U_ℓ^m contributes to the update of $\{U_i^{m+1}\}$, i.e., the region of influence of U_ℓ^m.

If the FV method is linear, i.e., the coefficients $d_{i,\ell}^m$ are independent of the numerical solution $\{U_i^m\}$, then (2.7) is also a necessary condition of mass conservation.

Proof. Summing (2.5) over i gives

$$\begin{aligned}\sum_i U_i^{m+1} &= \sum_i \sum_{\ell \in \mathcal{S}_i^m} d_{i,\ell}^m U_\ell^m \\ &= \sum_\ell \left(\sum_{i \in \mathcal{E}_\ell^m} d_{i,\ell}^m \right) U_\ell^m.\end{aligned}$$

Then, if (2.7) holds, then $\sum_i U_i^{m+1} = \sum_\ell U_\ell^m$. Hence the scheme is conservative.

Further, if the scheme is linear and conservative, then

$$\sum_\ell \left(\sum_{i \in \mathcal{E}_\ell^m} d_{i,\ell}^m \right) U_\ell^m = \sum_\ell U_\ell^m$$

holds for arbitrary $\{U_\ell^m\}$. By letting $U_\ell^m = 1$ and $U_j^m = 0$ for $j \neq \ell$, we have (2.7). □

Hence, (2.7) provides a feasible way to enforce mass conservation for any FV schemes expressed in the form of (2.5).

The FV SL scheme with remapping can be extended to the 2D linear transport equation with variable coefficients

$$u_t + \nabla \cdot (\boldsymbol{v}(x, y, t)u) = 0, \quad (x, y) \in \Omega, \tag{2.8}$$

where $\boldsymbol{v}$ denotes the velocity field $\boldsymbol{v}(x, y, t) = (a(x, y, t), b(x, y, t))$. The associated characteristic system writes

$$\begin{cases} \frac{dx(t)}{dt} &= a(x(t), y(t), t), \\ \frac{dy(t)}{dt} &= b(x(t), y(t), t). \end{cases} \tag{2.9}$$

Assume the domain Ω is partitioned uniformly with a collection of rectangular cells, i.e., $\Omega = \bigcup_{i,j} I_{ij}$, where $I_{ij} = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}]$. Denote by $h_x = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$ and by $h_y = y_{j+\frac{1}{2}} - y_{j-\frac{1}{2}}$ the mesh sizes in x and y directions, respectively. Similar to the 1D case, by solving (2.9) backward in time from t^{m+1} to t^m, we obtain the upstream cell $\tilde{I}_{ij}$ of cell I_{ij}, as shown in Fig. 2.2. Denote the upstream point of each grid point $(x_{i-\frac{1}{2}}, y_{j-\frac{1}{2}})$ as $(\tilde{x}_{i-\frac{1}{2},j-\frac{1}{2}}, \tilde{y}_{i-\frac{1}{2},j-\frac{1}{2}})$. Then define

$$\xi_{i-\frac{1}{2},j-\frac{1}{2}} = \frac{\tilde{x}_{i-\frac{1}{2},j-\frac{1}{2}} - x_{i-\frac{1}{2}}}{h_x}, \quad \eta_{i-\frac{1}{2},j-\frac{1}{2}} = \frac{\tilde{y}_{i-\frac{1}{2},j-\frac{1}{2}} - y_{j-\frac{1}{2}}}{h_y}$$

as the normalized shifts of grid point $(x_{i-\frac{1}{2}}, y_{j-\frac{1}{2}})$ in the x and y directions, respectively.

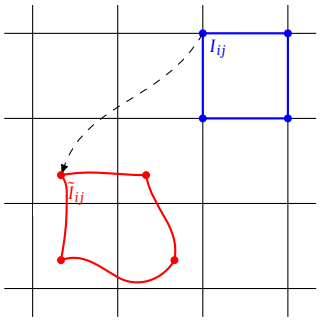

Fig. 2.2. Schematic illustration of the 2D SL FV scheme.

The 2D SL FV scheme is then formulated with the following identity

$$\iint_{I_{ij}} u(x, y, t^{m+1})dxdy = \iint_{\tilde{I}_{ij}} u(x, y, t^{m})dxdy. \tag{2.10}$$

Denote by U_{ij}^m the cell average of the numerical solution in the cell I_{ij} at time step $t = t^m$. Similar to the 1D case, to update the numerical solution, a polynomial is reconstructed over each cell using cell averages, and the integral on the right-hand side of (2.10) is computed in the subcell-by-subcell fashion. The scheme can be written as

$$U_{ij}^{m+1} = \sum_{\ell \in \mathcal{S}_{ij}^m} d_{ij,\ell}^m U_\ell^m, \tag{2.11}$$

where $\mathcal{S}_{ij}^m$ denotes the stencil employed to update U_{ij}^{m+1}. Again, the coefficients $\{d_{ij,\ell}^m\}$ are determined by the solution averages $\{U_{ij}^m\}$ together with normalized shifts $\{\xi_{i-\frac{1}{2},j-\frac{1}{2}}^m\}$, $\{\eta_{i-\frac{1}{2},j-\frac{1}{2}}^m\}$. In addition, as with the 1D case, it can be shown that the scheme (2.11) is mass conservative if

$$\sum_{(i,j)\in\mathcal{E}_\ell^m} d_{ij,\ell}^m = 1, \quad \forall l, \tag{2.12}$$

where $\mathcal{E}_\ell^m = \{(i, j) : \ell \in \mathcal{S}_{ij}^m\}$.

Remark 2.1. The SL FV methods are high order accurate in space but exact in time if the upstream cells are traced exactly. Further, the schemes are free of the CFL time step restriction for stability as the reconstructions are local.

Remark 2.2. The most computationally intensive part of the SL FV methods introduced above lies in tracking the geometry information of the upstream cells, including organizing the overlap regimes and integrating the local polynomial basis in a subregime-by-subregime manner. Such a search algorithm becomes highly challenging and expensive in high dimensions when the upstream cells may deform into irregular shapes as shown in Fig. 2.2. In next section, we propose an ML-assisted SL FV method to avoid such expensive tracing of upstream cells.

2.2. Data-driven conservative SL FV scheme

In this section, we introduce a novel data-driven SL FV scheme with enhanced accuracy and efficiency. This is motivated by a class of successful ML-based approaches for optimal discretizations for PDEs [1,57,23]. The main idea of such methods is that the solution manifold of a PDE often exhibits low dimensional structures, such as the recurrent patterns and coherent structures. Given high resolution training data, the ML-based discretization can effectively parameterize the solution manifold with coarse grids and attain a level of accuracy which often requires an order-of-magnitude finer grid for a traditional method using polynomial-based approximations. Hence, the methods can potentially capture the dynamics of interest even with an under-resolved grid. Furthermore, under the standard method-of-line framework, it is natural for the methods to satisfy inherent physical constraints such as conservation of mass, momentum and energy of the underlying physical system, as opposed to the purely data driven approach, and such an inductive bias aids in reliability and generalization of the ML-based model. The proposed data-driven SL FV scheme aims to take advantage of the methodology of ML-based discretization [1,57] and the SL FV formulation reviewed in the previous section for efficient transport simulations with the mass conservation.

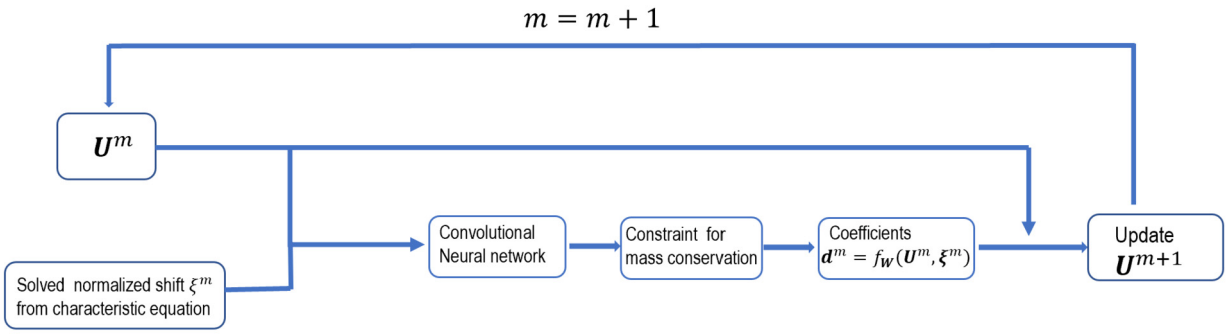

Fig. 2.3. Illustration of the proposed ML-based SL FV method.

2.2.1. Data generation

The training data is generated as in [1,57]. In particular, for each problem, we first sample a set of initial conditions from a given distribution. Then, staring from each sampled initial condition, we employ an accurate and reliable finite volume method such as an SL WEMO method or an Eulerian WENO method to generate a trajectory of solutions with a high-resolution mesh. It is critical to make sure that the solution structures of interest are resolved with sufficient resolution. Then the data are obtained by coarsening these solution trajectories to a mesh with an order-of-magnitude lower resolution. Noteworthy, we do not necessarily require an SL FV method to generate the data. Hence, if an Eulerian WENO method is utilized, then the expensive upstream cell tracking is completely avoided in both data generation and training phases.

2.2.2. 1D case

We first consider the training process for the 1D case and then briefly discussed the generalization to the 2D case. Without abuse of notations, we denote by $\boldsymbol{U}^m$, $\boldsymbol{\xi}^m$, and $\boldsymbol{d}^m$ the collection of $\{U_i^m\}$, $\{\xi_{i-\frac{1}{2}}^m\}$, and $\{d_{i,\ell}^m\}$, respectively. The proposed ML-based SL FV method is schematically illustrated in Fig. 2.3.

The main idea is that, instead of employing formulation (2.4), we propose to work with (2.5) in which the coefficients are inferred by a trained feed-forward NN. Motivated by the observation that $\boldsymbol{d}^m$ is determined by $\boldsymbol{U}^m$ and $\boldsymbol{\xi}^m$ for the traditional SL FV scheme, we propose to design

$$\boldsymbol{d}^m = f_{\boldsymbol{W}}(\boldsymbol{U}^m, \boldsymbol{\xi}^m), \tag{2.13}$$

where the NN $f_{\boldsymbol{W}}$ takes $\boldsymbol{U}^m$ and $\boldsymbol{\xi}^m$ as two-channel input and is constructed as a stack of convolutional layers with trainable parameters $\boldsymbol{W}$ and nonlinear activation functions, such as ReLU. The employed CNN can effectively capture hierarchical features of the solution [27,26], which are highly desired for transport modeling. A constraint layer is further built to the NN for exact mass conservation, which is known as a critical requirement for transport simulations and is closely related to the long term accuracy and stability. The proposed method attempts to replace the most expensive component of the traditional SL methods with a data-driven approach, with the goal of significantly simplifying the algorithm implementation and improving efficiency while simultaneously maintaining the mass conservation.

Once $\boldsymbol{d}^m$ is obtained, the solution $\boldsymbol{U}^{m+1}$ is updated with (2.5). However, unlike the standard SL formulation, we employ a set of fixed centered stencils $\mathcal{S}_i^m = \{i-s, i-s+1, \ldots, i+s-1, i+s\}$ consisting of $2s+1$ cells. Not only will this greatly simplify the algorithm development but also make it convenient to satisfy the mass conservation constraint. In fact, it follows from Theorem 2.1 that the mass is conserved as long as $\sum_{j=i-s}^{i+s} d_{j,i}^m = 1$, since U_i^m contributes to computing $\{U_{i-s}^{m+1}, U_{i-s+1}^{m+1}, \ldots, U_{i+s}^{m+1}\}$ as shown in Fig. 2.4. Therefore, we can simply add a constraint layer to $f_{\boldsymbol{W}}$ in (2.13) to enforce mass conservation for the proposed ML model. It is worth mentioning that it is nontrivial to enforce mass conservation with a non-fixed stencil under the employed CNN framework. Fig. 2.4 presents an illustration when $s=2$ and $\mathcal{S}_i^m$ consists of 5 cells. The NN $f_{\boldsymbol{W}}$ predicts the coefficients $\{d_{i,\ell}^m,\ \ell = i-2, \ldots, i+2\}$, and U_i^{m+1} is given by $U_i^{m+1} = \sum_{\ell=i-2}^{i+2} d_{i,\ell}^m U_\ell^m$. In this case, $\boldsymbol{d}^m$, $\boldsymbol{U}^m$, and $\boldsymbol{\xi}^m$ are 2D tensors of dimensions $N\times 5$, $N\times 1$, and $N\times 1$, respectively.

It is worth emphasizing that using fixed stencils will destroy the desired unconditional stability of the SL method. In [24], an ML-based SL approach was developed in the context of the level-set method. Such a method is designed based on correcting the local error incurred by the standard SL method using a NN and resembles a localized version of the error-correcting method using ML [39], and hence the CFL time step restriction can be avoided.

Note that for the generation of the training data, we do not need to collect the corresponding weights $\boldsymbol{d}^m$ but only the coarsened solution trajectories $\boldsymbol{U}^m$. On the other hand, the proposed end-to-end ML-based solver can effectively learn the optimal SL FV discretization, i.e., $\boldsymbol{d}^m$. It is numerically observed in Section 3 that the proposed ML-based SL FV method outperforms immensely the WENO method [21] for a collection of benchmark tests.

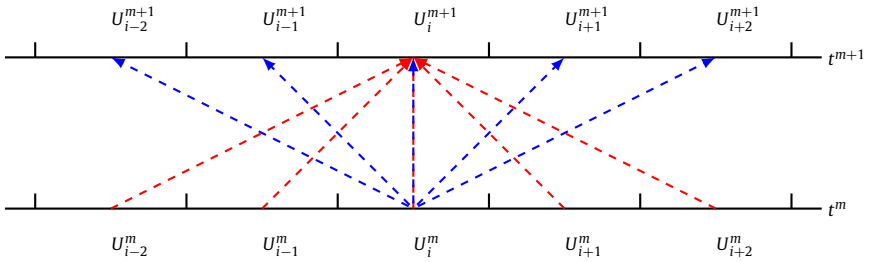

Fig. 2.4. Schematic illustration of the FV SL scheme with fixed 5-cell stencil $\{I_{i-2}, I_{i-1}, I_i, I_{i+1}, I_{i+2}\}$. U_i^{m+1} directly depends on $\{U_{i-2}^m, U_{i-1}^m, U_i^m, U_{i+1}^m, U_{i+2}^m\}$ and U_i^m contributes to computing $\{U_{i-2}^{m+1}, U_{i-1}^{m+1}, U_i^{m+1}, U_{i+1}^{m+1}, U_{i+2}^{m+1}\}$.

2.2.3. 2D case

The proposed algorithm can be generalized to the 2D case with a few simple modifications. We propose the following NN

$$\boldsymbol{d}^m = f_{\boldsymbol{W}}(\boldsymbol{U}^m, \boldsymbol{\xi}^m, \boldsymbol{\eta}^m), \tag{2.14}$$

where the tensors $\boldsymbol{U}^m, \boldsymbol{\xi}^m, \boldsymbol{\eta}^m$ are defined by collecting $\{U_{ij}^m\}$, $\{\xi_{i-\frac{1}{2},j-\frac{1}{2}}^m\}$, $\{\eta_{i-\frac{1}{2},j-\frac{1}{2}}^m\}$, respectively. $f_{\boldsymbol{W}}$ takes a 3-channel input and outputs the coefficient tensor $\boldsymbol{d}^m$ which is used to update the solution. The NN $f_{\boldsymbol{W}}$ is constructed by staking a sequence of 2D convolutional layers together with a constraint layer to enforce the condition (2.12) for exact mass conservation. Note that similar to the 1D case, we employ a set of squared fixed stencils, each of which consists of $(2s+1)\times(2s+1)$ cells. In particular, if a fixed 5×5-cell stencil and a grid of $N\times N$ cells are used, $\boldsymbol{d}^m, \boldsymbol{U}^m, \boldsymbol{\xi}^m$, and $\boldsymbol{\eta}^m$ are tensors of dimensions $N\times N\times 5\times 5$, $N\times N\times 1$, $N\times N\times 1$, and $N\times N\times 1$, respectively. Again, due to the use of fixed stencils, the SL formulation is constrained by the CFL condition.

2.2.4. Discussion

We end this section with some comments on the proposed method in the following:

- The weights $\left\{d_{i,\ell}^m\right\}$ may depend on the solution values and the shifts in the distant cells as opposed to a local method such as [24], since the employed NN incorporates multiple convolutional layers. Therefore, the weights may be different at different spatial locations, even if the corresponding $\{U_i^m\}$ and $\{\xi_{i-1/2}^m\}$ are the same locally.
- The solver is trained with a fixed mesh resolution and has limitations for the mesh refinement of the standard finite volume methods. However, thanks to the underlying CNN architecture, it is able to be generalized to larger computational domains by keeping the same mesh size.
- Compared with the ML-assisted Eulerian transport method in [57], the proposed SL FV method has the advantage of avoiding the use of any explicit time integrators. As a result, only a single evaluation of the neural network $f_{\boldsymbol{W}}$ is required per time evolution.
- The incorporation of characteristic information allows for a higher CFL of up to 2 while maintaining numerical stability with the use of fixed 5-cell stencils.
- The proposed method exhibits translation equivariance due to the inherent property of the underlying CNN [27,26]. This means that if the input, including the solution and the shift, is translated and then passed through the model, then the result is identical to passing the original input through the model and subsequently translating the output. As transport equations possess such translation symmetry, translation equivariance of the method can lead to improved data efficiency and generalization capabilities. It is possible to utilize group equivariant CNN [10] to exploit other types of symmetries including rotation and reflection present in the solution structures.

3. Numerical results

In this section, we carry out a series of numerical experiments to demonstrate the performance of our data-driven SL FV scheme for a collection of 1D and 2D transport equations. Noteworthy, the performance of the proposed scheme depends on the choice of hyperparameters of the CNN structure, and numerical results with default settings are presented in this section for simplicity. For both 1D and 2D problems, we employ 6 convolutional layers with 32 filters per layer, utilizing a kernel size of 3 or 5 and 3×3 or 5×5 for 1D and 2D cases, respectively, similar to [57]. A constraint layer is added to ensure mass conservation. The activation function used is ReLU. Following [32], Adam optimization algorithm is applied to train the network. We employ the Eulerian fifth-order FV WENO (WENO5) method [21], combined with the third order strong-stability-preserving Runge-Kutta (SSPRK3) time integrator [14] over fine-resolution grids to produce ground-truth reference solution trajectories, and the training data is generated by coarsening the reference solutions by a certain factor. For all the test examples, we mainly report the results by the proposed ML-based SL FV method and the WENO5 method

with the same mesh resolution, together with the reference solutions for comparison. It is worth emphasizing that we can use any accurate and reliable transport methods (e.g. WENO5 + SSPRK3) to generate training data. In all the plots reported below, "Neural net" denotes the proposed method and "WENO5" denotes the WENO5 method combined with SSPRK3.

3.1. One-dimensional transport equations

In this subsection, we present numerical results for simulating 1D transport equations.

Example 3.1. In this example we consider the following advection equation with a constant coefficient

$$u_t + u_x = 0, \quad x \in [0, 1], \tag{3.1}$$

and periodic conditions are imposed.

The training data is generated by coarsening 30 high-resolution solution trajectories over a 256-cell grid by a factor of 8. The initial condition for each trajectory is a square wave with height randomly sampled from $[0.1, 1]$ and width from $[0.2, 0.4]$. In addition, each coarsened trajectory contains 256 sequential time steps, and the CFL number is within the range of $[0.3, 1.8]$. The centered 5-cell stencils are employed to update the solution. For testing, initial conditions are square functions randomly selected from the same width and height range. For comparison, the reference solution is produced using WENO5 on a high-resolution 256-cell grid and then down-sampled to a coarse grid of 32 cells, the procedure of which is the same as generating the training data.

Fig. 3.1 plots three test samples during forward integration at several instances of time with CFL = 0.6. It is observed that the WENO5 method exhibits significant smearing near discontinuities, which deteriorates over time as a result of the accumulation of numerical diffusion. In contrast, the proposed ML-based solver has much improved shock resolution compared to WENO5 with the same mesh resolution: the numerical results are free of spurious oscillation and having very sharp shock transition. Notably, after long time simulations of 2560 time steps, which is 10 times of time steps for training, the results by the proposed method still stay very close to the reference solution, while a large amount of accumulated numerical diffusion of WENO5 dramatically smears the discontinuities.

Fig. 3.2 plots the time histories of the mean square errors which are averaged over all test samples. Compared with WENO5, the proposed ML-based method achieves a factor of approximately 8.6 less error in magnitude as shown in Fig. 3.2(a). Moreover, it is observed that the error by WENO5 increases over time, while the error by our method stays the same in magnitude with slight fluctuation. We further investigate the performance of the proposed method with different CFL numbers, i.e., different time step sizes, and report the result in Fig. 3.2(b). We observe that the errors by our method with different CFLs are almost of the same magnitude over time. Fig. 3.3 presents the time evolution of deviation in total mass for three test solution trajectories generated by our method. Evidently, the total mass is conserved up to the machine precision as expected.

Although our solver is trained with a data set where each trajectory contains a single square wave, it is observed that the model can be generalized to simulate the advection equation (3.1) with an initial condition consisting of two square waves, as shown in Fig. 3.4. Again, our solver demonstrates superior performance over WENO5. It is worth mentioning that the traditional reduced order models employing a direct parameterization of the underlying solution manifold are not capable of such generalization.

Note that the model trained on the data set of square waves is limited in its ability to be generalized to other initial value types. For instance, when the trained solver is directly applied to Gaussian initial conditions, it is observed that the numerical solutions are gradually transformed from Gaussian waves into square shapes, as shown in Fig. 3.5 (a). To improve the performance, we further include Gaussian shapes in the training data, enabling the model to accurately simulate both square and Gaussian waves, as shown in Fig. 3.5 (b). Hence, the proposed ML-based SL solver can effectively simulate problems with local solution structures that are similar to those observed during its training phase. For optimal accuracy and stability, the testing distribution should resemble the training distribution.

Last, we remark that the CFL number cannot be chosen above 2, otherwise the loss would not decrease during training. Such an observation is partly attributed to the fact that the region of dependence for updating one cell average is not completely contained within the 5-cell stencil if the CFL number is greater than 2.

Example 3.2. In this example, we consider the advection equation (3.1) with a more complicated solution profile consisting of triangle and square waves.

Similar to the previous example, we generate 30 high-resolution solution trajectories, each consisting of one triangle and one square waves with heights randomly sampled from $[0.2, 0.8]$ and widths from $[0.2, 0.3]$ over the 256-cell grid. By reducing the resolution from 256 to 32 (by a factor of 8), we obtain our training data. For this example, we set the stencil size to be 3, and the maximum CFL number allowed for training is reduced to 0.975 based on our numerical experiments. Each solution trajectory in the training data set contains 256 time steps with the CFL number ranging in [0.3,0.975]. The test data is randomly sampled from the same ranges of width and height. To evaluate the performance of the proposed

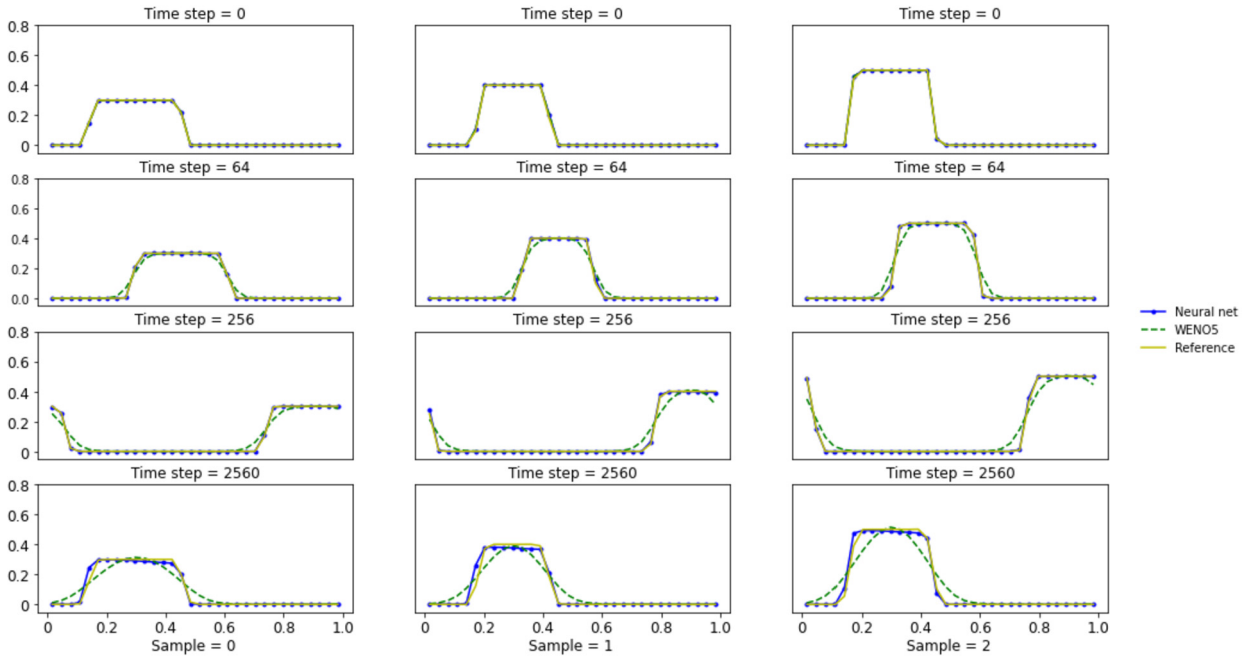

Fig. 3.1. Three test samples for square waves in Example 3.1. CFL=0.6 for both our method and WENO5.

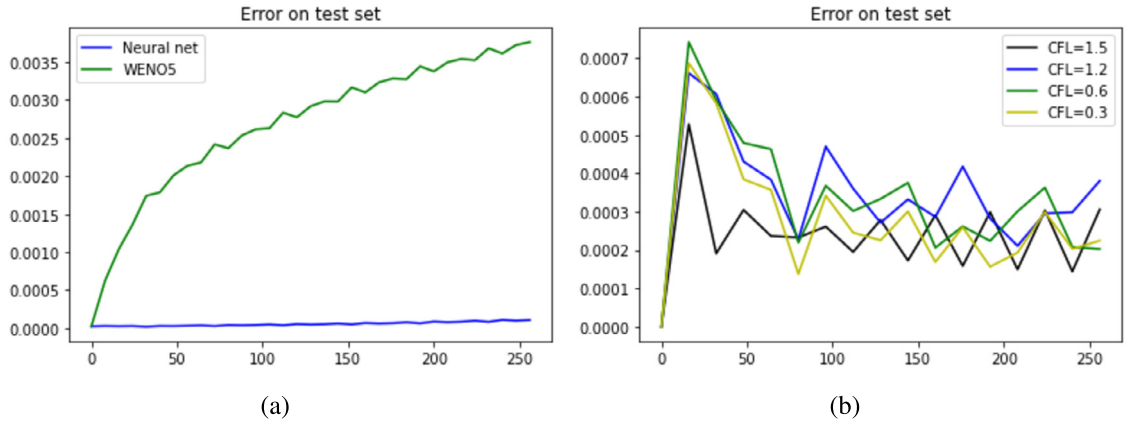

Fig. 3.2. Time histories of the errors in Example 3.1. (a): Errors by the proposed method compared with WENO5 averaged over all test samples. (b): Errors by the proposed method with different CFLs.

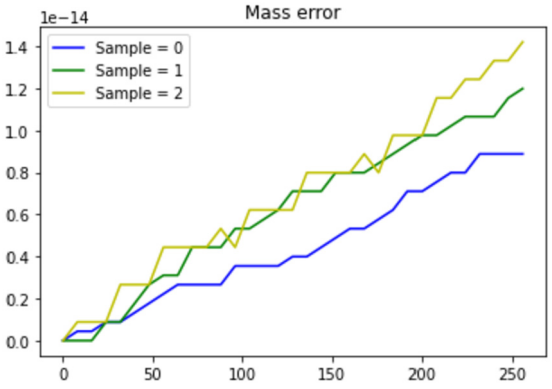

Fig. 3.3. Time histories of the deviation of total mass for three test samples in Example 3.1. CFL=0.6.

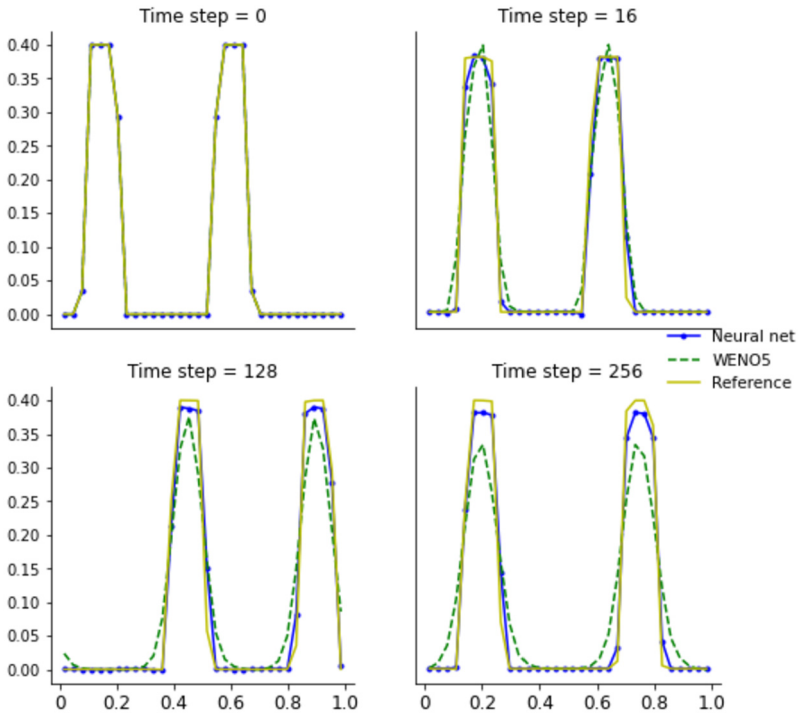

Fig. 3.4. Numerical solutions for Example 3.1 with an initial condition of two square waves. The proposed ML model is trained with a data set for which each trajectory only contains a single square wave. CFL=0.6.

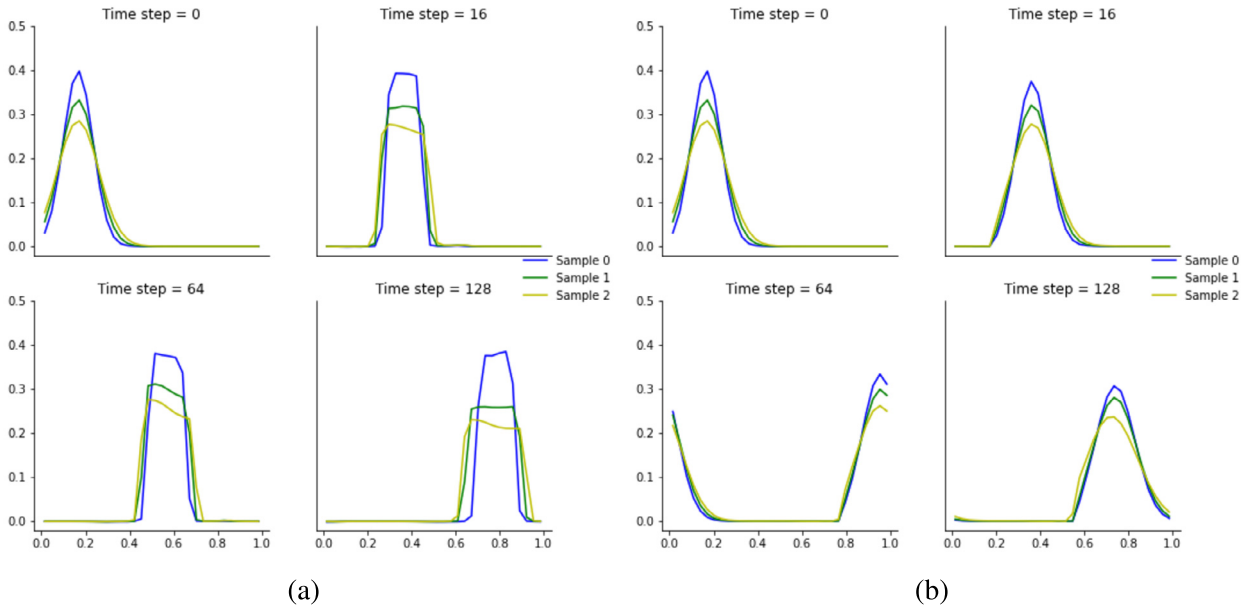

Fig. 3.5. Neural network prediction on Gaussian initial conditions. (a): The training data only include square waves. (b): The training data include both Gaussian and square shapes.

method, we calculate the ground-truth reference solution using WENO5 on a high-resolution grid with 256 cells and then reduce the resolution by a factor of 8 to a coarse grid of 32 cells.

In Fig. 3.6, we plot three test samples at several instances of time during forward integration with CFL = 0.6. The proposed ML-based solver generates numerical results with significantly higher resolution of non-smooth structures compared with WENO5. We then present the time histories of the mean square error in Fig. 3.7 (a), averaged over all test examples. Our ML-based solver achieves a reduction of error magnitude by a factor of approximately 7.8 compared with the traditional

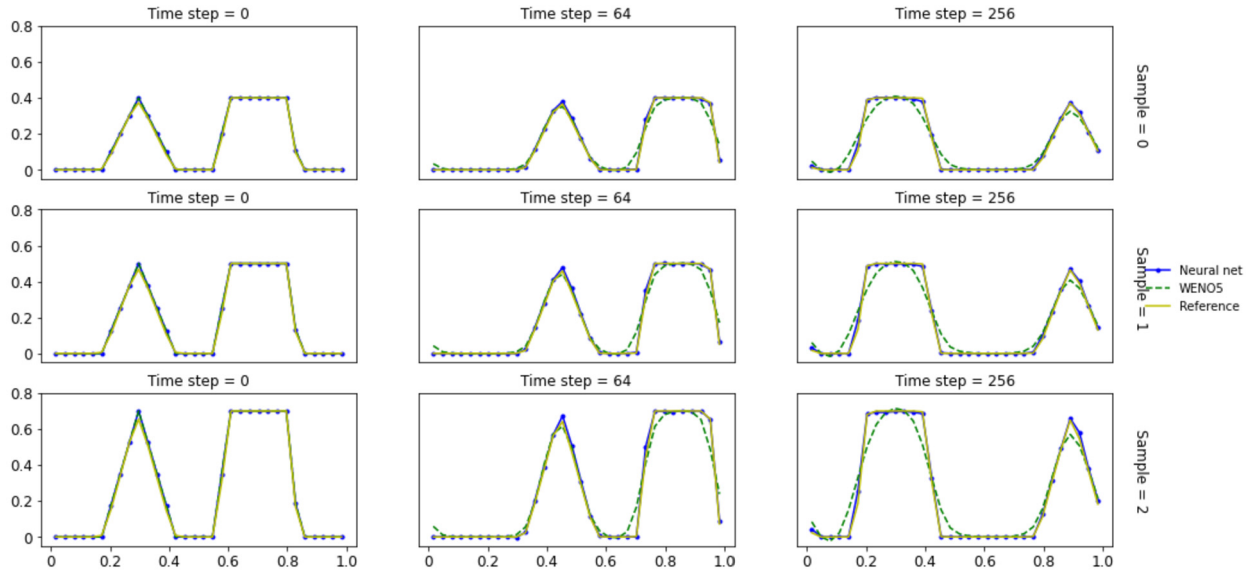

Fig. 3.6. Numerical solutions of three test samples for advection of triangle and square waves in Example 3.2. CFL=0.6.

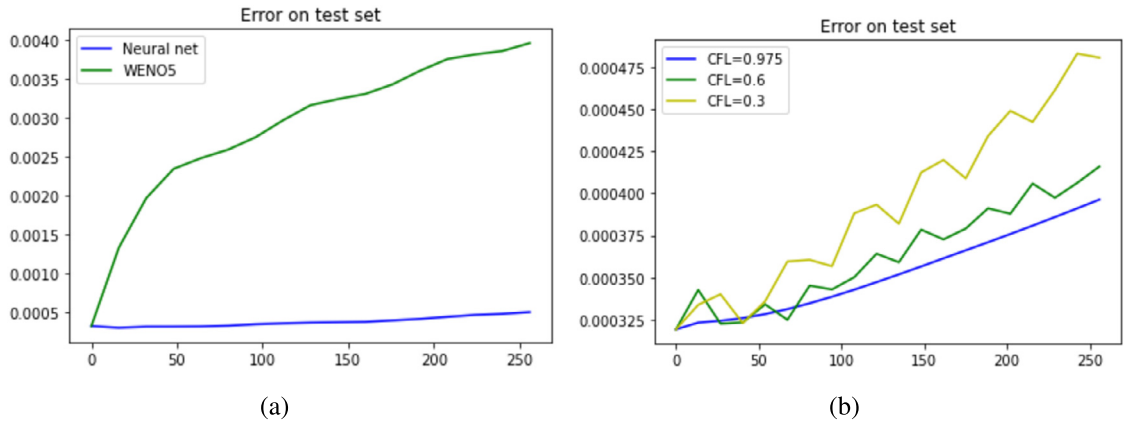

Fig. 3.7. Time histories of the errors in Example 3.2. (a): Errors by the proposed method compared with WENO5 averaged over all test samples. (b): Errors by the proposed method with different CFLs.

WENO5 solver. Additionally, the error of our solver grows at a much slower rate with time compared with WENO5. To further validate our solver, we consider three CFL numbers for testing, and present the time histories of errors in Fig. 3.7(b). It is observed that employing a larger CFL results in a smaller error and slower growth in time. Similar to the previous example, our method is mass conservative up to machine precision as demonstrated in Fig. 3.8.

Example 3.3. We simulate the following 1D advection equation with a variable coefficient

$$u_t + (u\sin(x+t))_x = 0, \quad x \in [0, 2\pi], \tag{3.2}$$

and periodic conditions are imposed.

We generate 90 solution trajectories, and each initial condition is a square function with heights randomly sampled from $[0.1, 1]$ and widths from $[2.5, 3.5]$ over the high-resolution grid of 256 cells. By reducing the grid resolution with a factor of 8, we obtain the training data over a 32-cell grid. In addition, each trajectory in the training data contains 30 sequential time steps, with the CFL number ranging in $[0.3, 1.2]$. In addition, the center of each square function is randomly sampled from the whole domain $[0, 2\pi]$. We set the stencil size to be 5. Again, the reference solution is generated by WENO5 over the 256-cell grid and down-sampled to the coarse grid of 32 cells. Note that the solution structure is more complicated than previous examples.

We first plot three test samples at several instances of time during forward integration with CFL=0.5 in Fig. 3.9. It can be observed that our solver can accurately resolve the solution structures with sharp shock transition and outperforms WENO5.

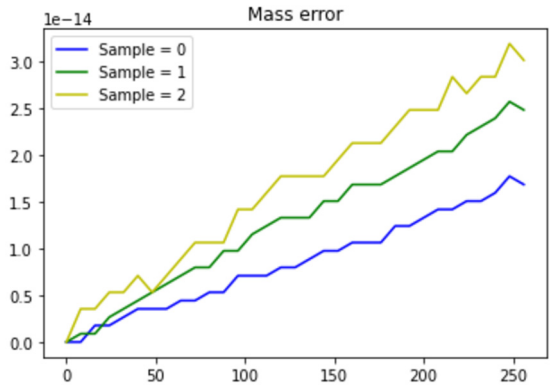

Fig. 3.8. Time evolution of the deviation of the total mass for three test samples in Example 3.2.

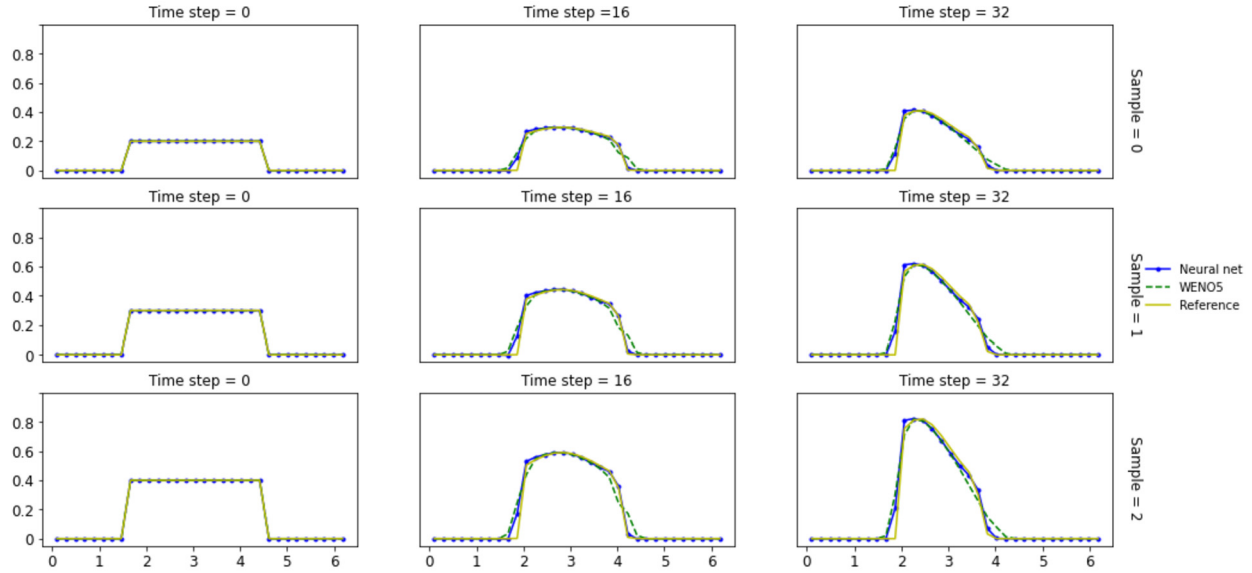

Fig. 3.9. Numerical solutions of three test samples for the transport equation with a variable coefficient in Example 3.3. CFL=0.5.

Fig. 3.10(a) shows the time histories of the mean square errors, averaged over all test examples. The proposed method achieves the reduction of the error by a factor of 3.7 in comparison with the WENO5 method. We further report the time histories of errors for the method with three different CFL numbers in Fig. 3.10(b), and it is observed the errors are comparable. As demonstrated in Fig. 3.11, the proposed method can conserve the total mass up to machine precision.

3.2. Two-dimensional transport equations

In this subsection, we present the numerical results for simulating several 2D benchmark advection problems.

Example 3.4. We solve the following constant-coefficient 2D transport equation

$$u_t + u_x + u_y = 0, \quad (x, y) \in [-1, 1]^2,$$

with periodic boundary conditions.

The training data is generated by coarsening 30 high-resolution solution trajectories over a 256×256-cell grid by a factor of 8 in each dimension, and each trajectory is initialized as a square wave with height randomly sampled from $[0.5, 1]$ and width from $[0.3, 0.5]$. One trajectory contains 256 sequential time steps, and the CFL number is chosen within the range of $[0.3, 1.8]$. We set the stencil size to be 5×5. For testing, the initial conditions are sampled from the same range of width and height. The reference solution is generated by WENO5 with the mesh of 256×256 cells and down sampled to the original coarse grid.

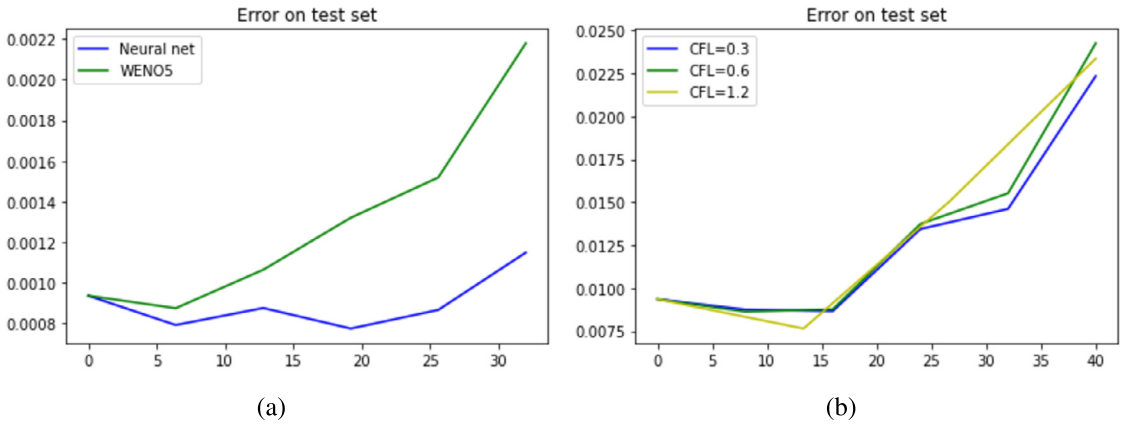

Fig. 3.10. Time histories of the errors for the transport equation with a variable coefficient in Example 3.3. (a): Errors by the proposed method compared with WENO5 averaged over all test samples. (b): Errors by the proposed method with different CFLs.

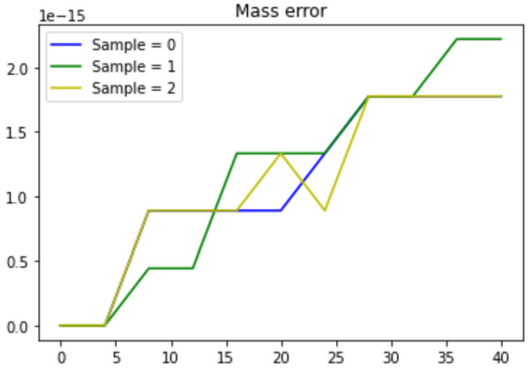

Fig. 3.11. Time histories of the deviation of the total mass of three test samples for the transport equation with a variable coefficient in Example 3.3.

Fig. 3.12 shows 1D cuts of solutions at $y = 0.5$ for three test examples at several instances of time during the forward integration with CFL=0.6. It is observed that the proposed method significantly outperforms WENO5 in resolving shocks sharply without introducing spurious oscillations. Furthermore, even after conducting simulations over a long period of 2560 time steps, the proposed method still produces highly accurate results. For a more effective comparison, we also provide the 2D plots of the solutions at time step 256 in Fig. 3.13. It can be seen that the solution produced by WENO5 exhibits noticeable smeared shocks, whereas the solution by the proposed method exhibits much sharper shock resolution.

Example 3.5. In this example, we simulate the 2D deformational flow proposed in [29], governed by the following 2D transport equation

$$u_t + (a(x, y, t)u)_x + (b(x, y, t)u)_y = 0, \quad (x, y) \in [0, 1]^2, \tag{3.3}$$

with the velocity field is a periodic swirling flow

$$\begin{aligned} a(x, y, t) &= \sin^2(\pi x)\sin(2\pi y)\cos(\pi t/T), \\ b(x, y, t) &= -\sin^2(\pi y)\sin(2\pi x)\cos(\pi t/T). \end{aligned} \tag{3.4}$$

It is a widely recognized benchmark example for transport solvers. The solution profile is deformed over time. At $t = \frac{1}{2}T$ the direction of this flow reverses, while the solution returns to the initial state at $t = T$, completing a full cycle of the evolution.

We set the period $T = 2$ and the initial condition to be a cosine bell centered at $[c_x, c_y]$:

$$\begin{aligned} u(x, y) &= \frac{1}{2}[1 + \cos(\pi r)] \\ r(x, y) &= \min\left[1, 6\sqrt{(x - c_x)^2 + (y - c_y)^2}\right]. \end{aligned} \tag{3.5}$$

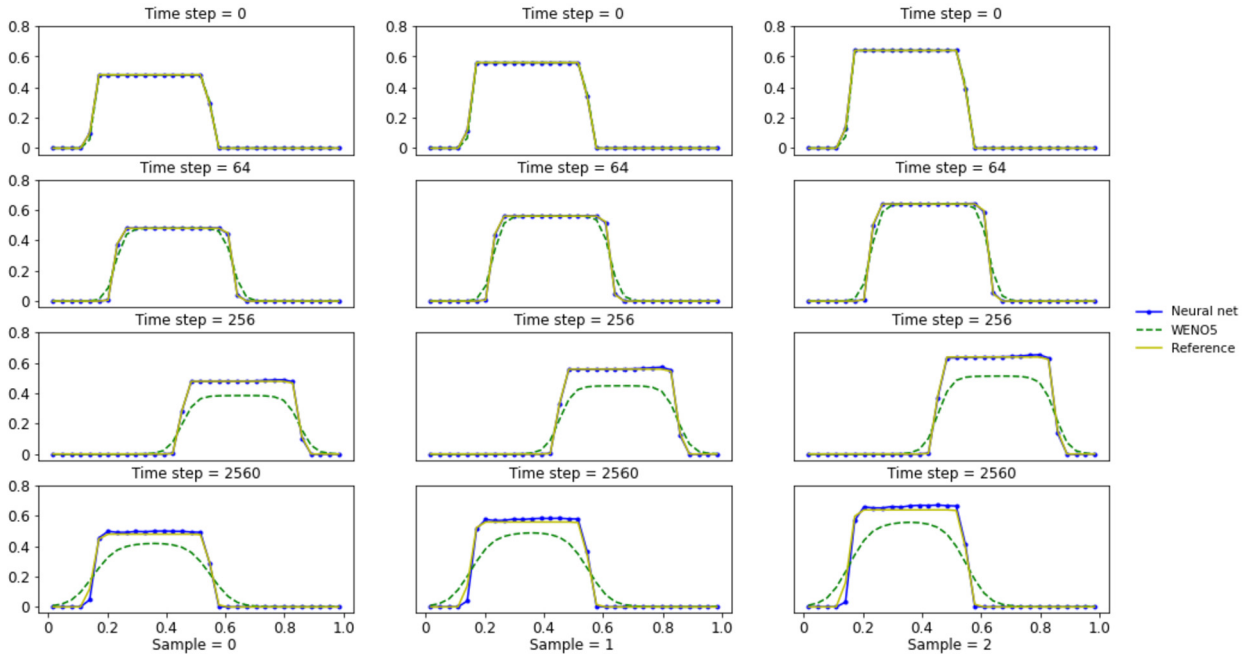

Fig. 3.12. 1D cuts at $y = 0.5$ of three test samples for 2D transport equation with constant coefficients in Example 3.4. CFL=0.6.

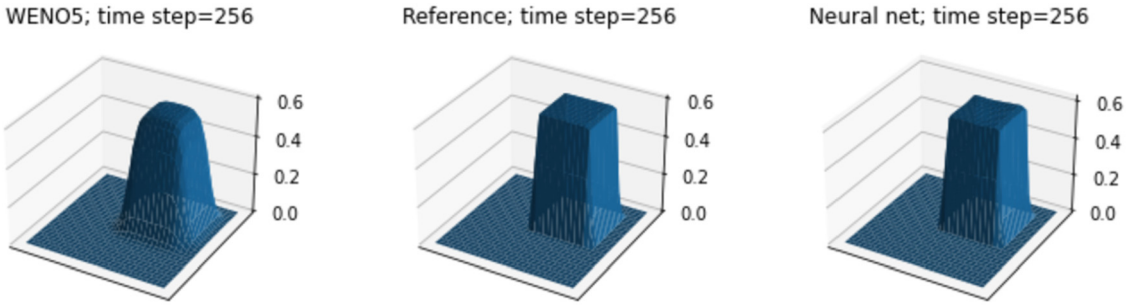

Fig. 3.13. One test samples for the 2D transport equation with constant coefficients in Example 3.4. 2D plots of the solutions at time step 256. CFL=0.6.

We initialize 30 trajectories with c_x and c_y randomly sampled from $[0.25, 0.75]$ using a high-resolution mesh of 256×256 cells, which are coarsened by a factor of 8 in each dimension as the training data. Each solution trajectory in the training data contains a sequence of time steps from $t = 0$ to $t = T$, with the CFL number ranging in [0.3,1.8]. We set the stencil size to be 5×5. During testing, the initial conditions are sampled from the same distribution as the training data.

In Fig. 3.14, we show the contour plots of the numerical solutions computed by the proposed method and WENO5 along with the reference solution for one test sample. The CFL number is taken as 1.8. Note that the solution is significantly distorted at $t = T/2$ and returns to its initial state at $t = T$. It is observed that our method produces a result that is in good agreement with the reference solution. However, the solution obtained using WENO5 noticeably deviates from the reference solution due to a large amount of numerical diffusion.

Furthermore, the model which was trained using data from solution trajectories featuring a single bell can generalize to simulate problems with an initial condition containing two bells, as shown in Fig. 3.15. The observation is similar to the single bell case. Last, we compare the errors of the numerical solutions by our method and by WENO5 at $t = T$ in Table 1. It is observed that the error of the ML-based SL FV method over the mesh of 32×32 cells is much smaller that of WENO5 with the same mesh size and is comparable with that of WENO5 over the finer mesh of 128×128 cells, demonstrating the efficiency of the proposed method.

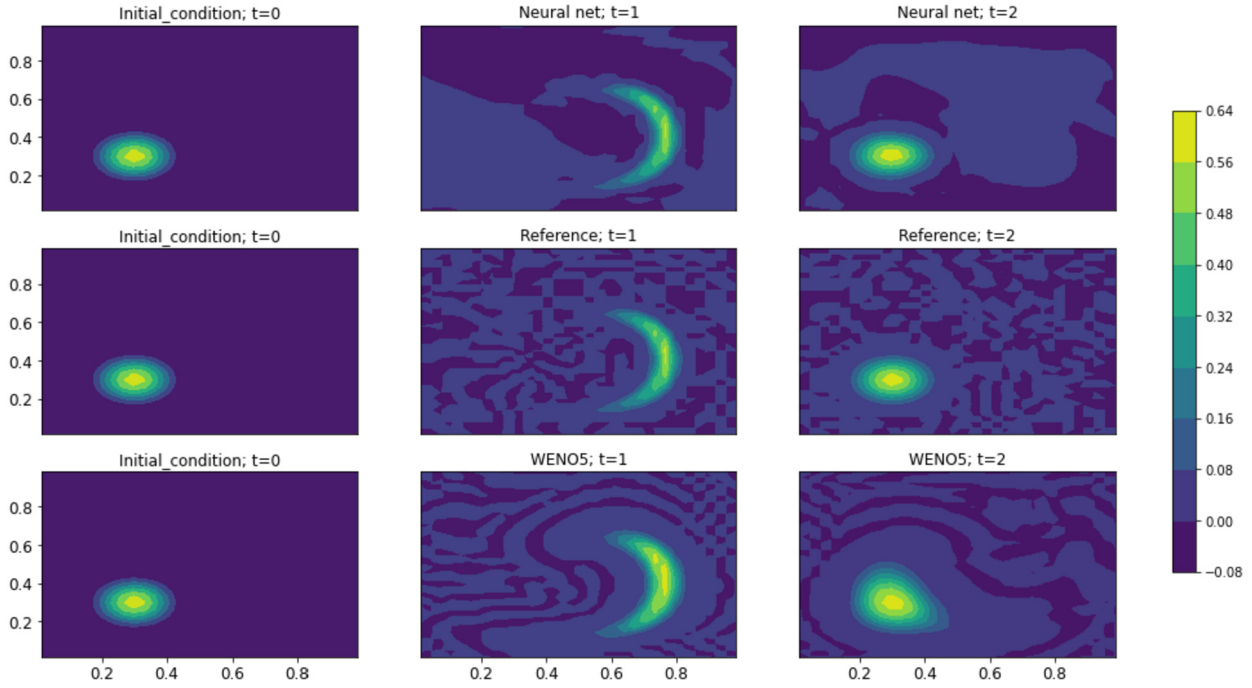

Fig. 3.14. Contour plots of the numerical solutions of the 2D deformational flow at $t = 0$, 1, 2 in Example 3.5 for one test sample. The solution profiles are significantly distorted at $t = 1$, and then the flow reverses. At $t = 2$ the solutions return to the initial states. CFL=1.8.

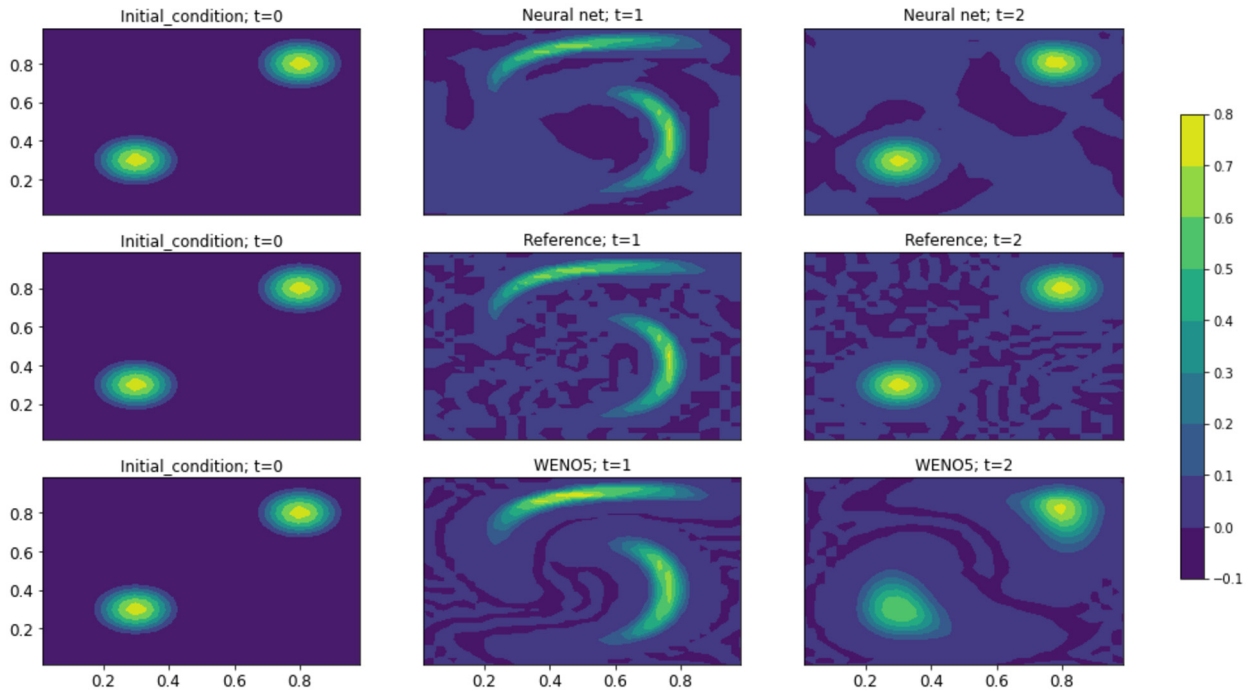

Fig. 3.15. Contour plots of the numerical solutions of the 2D deformational flow at $t = 0$, 1, 2 in Example 3.5 with two cosine bells. The proposed ML model is trained with a data set for which each trajectory only contains a single cosine bell. CFL=1.8.

Table 1
Accuracy comparison for the ML-based SL FV method and WENO5 of three test samples for the 2D deformational flow in Example 3.5.

Samples \ Method	WENO5 (128×128)	WENO5 (32×32)	ML-based SL FV method
Sample = 0	9.269E-06	1.173E-03	1.955E-05
Sample = 1	1.069E-05	1.982E-03	9.753E-06
Sample = 2	9.302E-06	1.623E-03	1.107E-05

4. Conclusion

In this paper, we proposed a machine-learning-assisted semi-Lagrangian (SL) finite volume (FV) scheme for efficient simulations of transport equations. Our method leverages a convolutional neural network to optimize SL discretization using high-resolution data, eliminating the need for costly upstream cell tracking. With a fixed 5-cell stencil, the CFL number can reach as large as 1.8. The inclusion of a constraint layer in the network ensures total mass conservation to machine precision. Numerical experiments show superior performance compared with the WENO methods. Future work includes extending the method to nonlinear transport equations such as the Vlasov system and investigating the use of graph neural networks for accommodating unstructured meshes, adaptivity, complex geometries, among many others.

CRediT authorship contribution statement

Yongsheng Chen: Methodology, Implementation, Computational study, Writing; **Wei Guo:** Conceptualization, Formal analysis, Methodology, Writing; **Xinghui Zhong:** Conceptualization, Formal analysis, Methodology, Writing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

Research work of W. Guo is partially supported by the NSF grant NSF-DMS-2111383, Air Force Office of Scientific Research FA9550-18-1-0257. Research work of X. Zhong is partially supported by the NSFC Grant 12272347.

References

[1] Y. Bar-Sinai, S. Hoyer, J. Hickey, M.P. Brenner, Learning data-driven discretizations for partial differential equations, Proc. Natl. Acad. Sci. 116 (31) (2019) 15344–15349.
[2] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (2018) 1–43.
[3] K. Bhattacharya, B. Hosseini, N.B. Kovachki, A.M. Stuart, Model reduction and neural networks for parametric PDEs, SMAI J. Comput. Math. 7 (2021) 121–157.
[4] J. Brandstetter, D. Worrall, M. Welling, Message passing neural PDE solvers, arXiv preprint, arXiv:2202.03376, 2022.
[5] Y. Chen, J. Yan, X. Zhong, Cell-average based neural network method for third order and fifth order KdV type equations, Front. Appl. Math. Stat. 8 (2022).
[6] Z. Chen, V. Churchill, K. Wu, D. Xiu, Deep neural network modeling of unknown partial differential equations in nodal space, J. Comput. Phys. 449 (2022) 110782.
[7] C.-Z. Cheng, G. Knorr, The integration of the Vlasov equation in configuration space, J. Comput. Phys. 22 (3) (1976) 330–351.
[8] V. Churchill, D. Xiu, Deep learning of chaotic systems from partially-observed data, arXiv:2205.08384 [nlin], 2022.
[9] B. Cockburn, C.-W. Shu, Runge–Kutta discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comput. 16 (2001) 173–261.
[10] T. Cohen, M. Welling, Group equivariant convolutional networks, in: International Conference on Machine Learning, PMLR, 2016, pp. 2990–2999.
[11] R. Courant, E. Isaacson, M. Rees, On the solution of nonlinear hyperbolic differential equations by finite differences, Commun. Pure Appl. Math. 5 (3) (1952) 243–255.
[12] S. Cuomo, V.S. di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics-informed neural networks: where we are and what's next, arXiv:2201.05624 [physics], 2022.
[13] C. Erath, P.H. Lauritzen, H.M. Tufo, On mass conservation in high-order high-resolution rigorous remapping schemes on the sphere, Mon. Weather Rev. 141 (6) (2013) 2128–2133.
[14] S. Gottlieb, C.-W. Shu, E. Tadmor, Strong stability-preserving high-order time discretization methods, SIAM Rev. 43 (1) (2001) 89–112.
[15] D. Greenfeld, M. Galun, R. Basri, I. Yavneh, R. Kimmel, Learning to optimize multigrid PDE solvers, in: International Conference on Machine Learning, PMLR, 2019, pp. 2415–2423.
[16] Tong Qin, Kailiang Wu, Dongbin Xiu, Data driven governing equations approximation using deep neural networks, J. Comput. Phys. 395 (2019) 620–635.
[17] Kailiang Wu, Dongbin Xiu, Data-driven deep learning of partial differential equations in modal space, J. Comput. Phys. 408 (2020) 109307.
[18] W. Guo, R.D. Nair, J.-M. Qiu, A conservative semi-Lagrangian discontinuous Galerkin scheme on the cubed sphere, Mon. Weather Rev. 142 (1) (2014) 457–475.

[19] J.-T. Hsieh, S. Zhao, S. Eismann, L. Mirabella, S. Ermon, Learning neural PDE solvers with convergence guarantees, in: International Conference on Learning Representations, 2019.
[20] A.D. Jagtap, E. Kharazmi, G.E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems, Comput. Methods Appl. Mech. Eng. 365 (2020) 113028.
[21] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1) (1996) 202–228.
[22] G. Kissas, J. Seidman, L.F. Guilhoto, V.M. Preciado, G.J. Pappas, P. Perdikaris, Learning operators with coupled attention, arXiv:2201.01032 [physics], 2022.
[23] D. Kochkov, J.A. Smith, A. Alieva, Q. Wang, M.P. Brenner, S. Hoyer, Machine learning–accelerated computational fluid dynamics, Proc. Natl. Acad. Sci. 118 (21) (2021) e2101784118.
[24] L.Á. Larios-Cárdenas, F. Gibou, Error-correcting neural networks for semi-Lagrangian advection in the level-set method, J. Comput. Phys. 471 (2022) 111623.
[25] P.H. Lauritzen, R.D. Nair, P.A. Ullrich, A conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid, J. Comput. Phys. 229 (5) (2010) 1401–1424.
[26] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, in: The Handbook of Brain Theory and Neural Networks, vol. 3361, 1995, p. 1995.
[27] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel, Handwritten digit recognition with a back-propagation network, Adv. Neural Inf. Process. Syst. 2 (1989).
[28] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[29] R.J. LeVeque, High-resolution conservative algorithms for advection in incompressible flow, SIAM J. Numer. Anal. 33 (2) (1996) 627–665.
[30] Z. Li, N.B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A.M. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, in: International Conference on Learning Representations, 2020.
[31] Z. Li, N.B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A.M. Stuart, A. Anandkumar, Neural operator: graph kernel network for partial differential equations, in: ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations, 2020.
[32] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: International Conference on Learning Representations, 2018.
[33] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (Mar. 2021) 218–229.
[34] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, G.E. Karniadakis, A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data, Comput. Methods Appl. Mech. Eng. 393 (2022) 114778.
[35] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis DeepXDE, A deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228.
[36] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S.G. Johnson, Physics-informed neural networks with hard constraints for inverse design, SIAM J. Sci. Comput. 43 (6) (2021) B1105–B1132.
[37] L. McClenny, U. Braga-Neto, Self-adaptive physics-informed neural networks using a soft attention mechanism, arXiv:2009.04544 [cs, stat], 2022.
[38] G. Pang, L. Lu, G.E. Karniadakis, fPINNs: fractional physics-informed neural networks, SIAM J. Sci. Comput. 41 (4) (2019).
[39] J. Pathak, M. Mustafa, K. Kashinath, E. Motheau, T. Kurth, M. Day, Using machine learning to augment coarse-grid computational fluid dynamics simulations, arXiv preprint, arXiv:2010.00072, 2020.
[40] C. Qiu, J. Yan, Cell-average based neural network method for hyperbolic and parabolic partial differential equations, arXiv:2107.00813, 2021.
[41] J.-M. Qiu, A. Christlieb, A conservative high order semi-Lagrangian WENO method for the Vlasov equation, J. Comput. Phys. 229 (4) (2010) 1130–1149.
[42] J.-M. Qiu, C.-W. Shu, Conservative high order semi-Lagrangian finite difference WENO methods for advection in incompressible flow, J. Comput. Phys. 230 (4) (2011) 863–889.
[43] J.-M. Qiu, C.-W. Shu, Positivity preserving semi-Lagrangian discontinuous Galerkin formulation: theoretical analysis and application to the Vlasov–Poisson system, J. Comput. Phys. 230 (23) (2011) 8386–8409.
[44] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[45] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (Part I): data-driven solutions of nonlinear partial differential equations, arXiv:1711.10561 [cs, math, stat], 2017.
[46] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (Part II): data-driven discovery of nonlinear partial differential equations, arXiv:1711.10566 [cs, math, stat], 2017.
[47] D. Ray, J.S. Hesthaven, An artificial neural network as a troubled-cell indicator, J. Comput. Phys. 367 (2018) 166–191.
[48] A. Robert, A stable numerical integration scheme for the primitive meteorological equations, Atmos.-Ocean 19 (1) (1981) 35–46.
[49] J.A. Rossmanith, D.C. Seal, A positivity-preserving high-order semi-Lagrangian discontinuous Galerkin scheme for the Vlasov–Poisson equations, J. Comput. Phys. 230 (16) (2011) 6203–6232.
[50] E. Sonnendrücker, J. Roche, P. Bertrand, A. Ghizzo, The semi-Lagrangian method for the numerical resolution of the Vlasov equation, J. Comput. Phys. 149 (2) (1999) 201–220.
[51] A. Staniforth, J. Côté, Semi-Lagrangian integration schemes for atmospheric models—a review, Mon. Weather Rev. 119 (9) (1991) 2206–2223.
[52] Z. Sun, S. Wang, L.-B. Chang, Y. Xing, D. Xiu, Convolution neural network shock detector for numerical solution of conservation laws, Commun. Comput. Phys. 28 (5) (2020) 2075–2108.
[53] N. Trask, R.G. Patel, B.J. Gross, P.J. Atzberger, GMLS-Nets: a framework for learning from unstructured data, arXiv:1909.05371 [physics, stat], 2019.
[54] Y. Wang, Z. Shen, Z. Long, B. Dong, Learning to discretize: solving 1D scalar conservation laws via deep reinforcement learning, Commun. Comput. Phys. 28 (5) (2020) 2158–2179.
[55] J. Yu, L. Lu, X. Meng, G.E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Comput. Methods Appl. Mech. Eng. 393 (2022) 114823, arXiv:2111.02801 [physics].
[56] X. Yu, C.-W. Shu, Multi-layer perceptron estimator for the total variation bounded constant in limiters for discontinuous Galerkin methods, Matematica 1 (1) (2022) 53–84.
[57] J. Zhuang, D. Kochkov, Y. Bar-Sinai, M.P. Brenner, S. Hoyer, Learned discretizations for passive scalar advection in a two-dimensional turbulent flow, Phys. Rev. Fluids 6 (6) (2021) 064605.