

Commun. Theor. Phys. **72** (2020) 115003 (11pp) https://doi.org/10.1088/1572-9494/abb7c8

A deep learning method for solving third-order nonlinear evolution equations

Jun Li (李军)[1] **and Yong Chen** (陈勇)[2,3,4]

[1] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, 200062, China
[2] School of Mathematical Sciences, Shanghai Key Laboratory of PMMP, Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, 200062, China
[3] College of Mathematics and Systems Science, Shandong University of Science and Technology, Qingdao, 266590, China
[4] Department of Physics, Zhejiang Normal University, Jinhua, 321004, China

E-mail: ychen@sei.ecnu.edu.cn

Received 15 May 2020, revised 29 July 2020
Accepted for publication 29 July 2020
Published 20 October 2020

Abstract

It has still been difficult to solve nonlinear evolution equations analytically. In this paper, we present a deep learning method for recovering the intrinsic nonlinear dynamics from spatiotemporal data directly. Specifically, the model uses a deep neural network constrained with given governing equations to try to learn all optimal parameters. In particular, numerical experiments on several third-order nonlinear evolution equations, including the Korteweg–de Vries (KdV) equation, modified KdV equation, KdV–Burgers equation and Sharma–Tasso–Olver equation, demonstrate that the presented method is able to uncover the solitons and their interaction behaviors fairly well.

Keywords: deep learning, nonlinear evolution equations, soliton interaction, nonlinear dynamics

(Some figures may appear in colour only in the online journal)

1. Introduction

Nonlinear evolution equations, which depend on certain space–time signatures, have a multitude of important applications across broad disciplines including physics, finance and biology. Certain special solutions to such equations can exhibit soliton behaviors, that is, they do not disperse and thus conserve their original forms after the collision [1]. Moreover, interaction between solitons is one of the most fascinating features of many soliton phenomena [2].

While direct numerical solutions to some evolution equations are computationally expensive, with the revival of deep learning, it has attracted much interest on the development of more efficient data-driven solutions to nonlinear evolution equations [3–5]. As a direction of machine learning, deep learning methods are able to effectively learn the feature representations from raw data [6–10]. However, to our knowledge, previous works focus mainly on some simple solutions to the given equations, which could not uncover the soliton behaviors under some circumstances. Thus, we propose to combine a neural network framework with some underlying physical laws to reconstruct the soliton solutions.

For a certain amount of physical systems, some nonlinear and dispersive processes compete while the dissipation can be neglected. Therefore, in this paper, we will study nonlinear time-dependent partial differential equations where each contains the dispersive term in addition to other partial derivatives. These equations often play important roles in many scientific applications and physical phenomena. Specifically, we consider the $(1 + 1)$-dimensional third-order nonlinear evolution equations of the form

$$u_t = \mathcal{N}(u, u_x, u_{xx}, u_{xxx}), \tag{1}$$

in order to solve their soliton solutions, where the subscripts t and x denote the partial derivatives with respect to them and $\mathcal{N}$ is a nonlinear function of the solution u and its arbitrary-order partial derivatives with respect to the spatial variable x (concretely, in this work, the highest order is three).

Specifically, we approximate the latent solution u with a deep neural network [11–13] and then compute the derivatives of the network approximation u with respect to time t and space x with the help of automatic differentiation [14, 15].

Consequently, define the residual network

$$f := u_t - \mathcal{N}(u, u_x, u_{xx}, u_{xxx}), \tag{2}$$

and then the solution network is trained to satisfy the residual constraint (2), which plays a role of regularization and is embedded into the mean-squared objective function [16]

$$L = \frac{1}{N_u}\sum_{i=1}^{N_u}|u(t_u^i, x_u^i) - u^i|^2 + \frac{1}{N_f}\sum_{j=1}^{N_f}|f(t_f^j, x_f^j)|^2. \tag{3}$$

In this work, we choose the network architecture in a consistent fashion [17]. Specifically, we learn the unknown solution u by using a 13-layer feedforward network with 40 neurons per hidden layer. For the choice of activation functions, we have conducted many experiments for different functions such as tanh, sin, sigmoid (σ) and rectified linear units (ReLU) in different number of layers and neurons. We find that the tanh function is a little unstable. Moreover, the results indicate that the σ and ReLU functions could not represent the data in current settings. So we select sin as the activation function in most cases. In addition, we just tune all parameters of the objective (3) using the L-BFGS method [18]. More modern and efficient algorithms can be adopted for larger-scale data, for example, Adam [19], which is a variant of the stochastic gradient descent algorithm. All numerical examples reported here are run on a MacBook Pro computer with 2.4 GHz Dual-Core Intel Core i5 processor and 8 GB memory.

The outline of this paper follows. In section 2, we reconstruct the one-soliton and two-soliton solutions to the KdV equation from data collected from simulations. Consequently, we recover the one-soliton and breather solutions to the mKdV equation in section 3. In section 4, we then consider the kink solution to the KdV–Burgers equation. In section 5, we focus mainly on the soliton fusion and fission phenomena of the STO equation. Finally, some concluding discussion and remarks are contained in section 6.

2. The KdV equation

The KdV equation [20, 21] is a canonical model which describes the unidirectional propagation of shallow water waves with certain small amplitude and long wavelength. It also is one of the earliest equations with soliton solutions. The KdV equation can be regarded as a dispersive modification of the Burgers equation and converted by the Cole–Hopf transformation. The dispersion and nonlinearity of this equation balance each other which leads to the wave propagation without losing energy. However, The manifestation of the balance may vary from system to system, thus other evolution equations could have different soliton forms from the KdV equation whose soliton solutions are bell-shaped.

In this section, we consider the KdV equation along with Dirichlet boundary conditions [22–24] given by

$$\begin{cases} u_t + 6uu_x + u_{xxx} = 0, x \in [-20, 20], t \in [-5, 5], \\ u(t_0, x) = u_0(x), \\ u(t, -20) = u(t, 20) = 0, \end{cases} \tag{4}$$

where $u_0(x)$ is an arbitrary real-valued function. In this case, $\mathcal{N} = -6uu_x - u_{xxx}$.

Note that this equation and the mKdV equation which will be considered in the next section are both special cases of the generalized KdV equation

$$u_t + u_{xxx} + (u^p)_x = 0,$$

where the case $p = 2$ obviously corresponds to the KdV equation and $p = 3$ to the mKdV equation. By the way, these two equations are completely integrable.

2.1. One-soliton solution

Here, we first consider the one soliton problem. Some exact soliton solutions to such nonlinear evolution equations can be expressed in terms of elementary functions and then these solutions are very important for understanding the nonlinearity of these systems better. Meanwhile, they are also useful in testing the performance and accuracy of certain numerical methods. Applying some analytic methods [25, 26], one can show that the exact one-soliton solution to equation (4) admits the explicit expression given by

$$u(t, x) = \frac{c}{2}\operatorname{sech}^2\left(\frac{\sqrt{c}}{2}(x - x_0 - ct)\right).$$

Specifically, we just set $c = 3$ for convenience. Then, the corresponding initial condition is obtained with a specific initial displacement by

$$u_0(x) = \frac{3}{2}\operatorname{sech}^2\left(\frac{\sqrt{3}}{2}(x + 15)\right). \tag{5}$$

We simulate equation (4) using the conventional spectral method to obtain the data. Specifically, starting from the initial condition (5), we use the Chebfun package [27] with a Fourier spatial discretization with 512 modes and a 4th-order explicit Runge–Kutta (RK) integrator with time-step size 1×10^{-4}, and then integrate the equation up to the final instant $t = 5$. The solution is saved every $\Delta t = 0.05$ to give us totally 201 snapshots. We generate a smaller training dataset out of this data by randomly sub-sampling $N_u = 100$ initial-boundary data and $N_f = 10\,000$ collocation points which are generated by the Latin hypercube sampling method [28].

Figure 1 demonstrates our result for the data-driven one-soliton solution to the KdV equation (4). Specifically, given a set of initial and boundary data points, we try to learn the latent solution $u(t, x)$ by tuning all learnable parameters of the network using the loss function (3). The top panel of figure 1 compares between the exact solution and the predicted spatiotemporal solution. The model achieves a relative $\mathbb{L}_2$ error of size 3.44×10^{-3} in a runtime of approximately three and half a minute. We can see a more detailed assessment in the

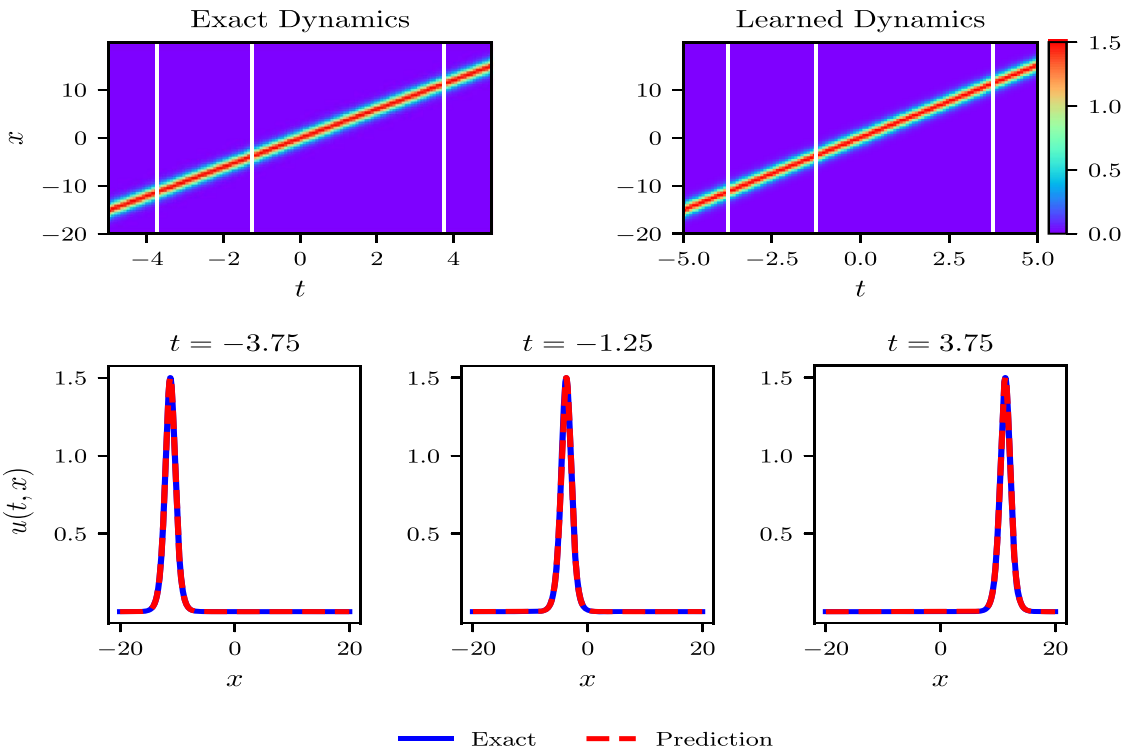

Figure 1. The KdV equation. Top: a one-soliton solution to the KdV equation (left panel) is compared to the corresponding predicted solution to the learned equation (right panel). The network correctly captures the dynamics behavior and accurately reproduces the soliton solution with a relative $\mathbb{L}_2$ error of 3.44×10^{-3}. Bottom: the comparison of the predicted and exact soliton solutions which correspond to the three temporal snapshots depicted by the white vertical lines in the top panel is presented.

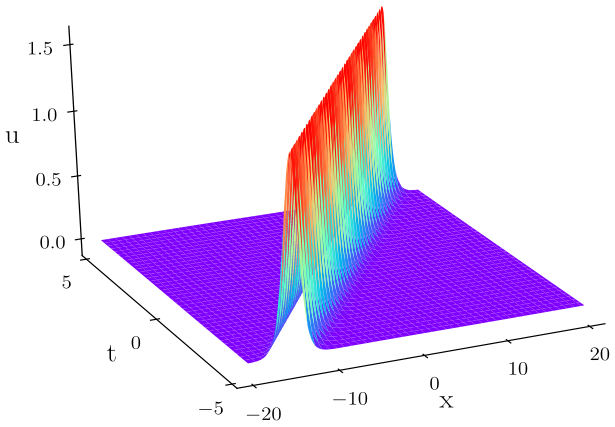

Figure 2. The spatiotemporal behavior of a one-soliton solution to the learned KdV equation.

bottom panel of figure 1. We particularly present a comparison between the exact solution and the predicted solutions at different times $t = -3.75, -1.25, 3.75$. The algorithm accurately reconstructs the one-soliton solution to the KdV equation.

From figure 2, we can observe the reconstructed single solitary wave motion better.

2.2. Two-soliton solutions

Many non-integrable equations also possess localized shape-preserving traveling waves that resemble soliton solutions. For example, it would be indistinguishable from a KdV soliton to single traveling wave solution of the wave equation expressed by

$$u_t + u_x = 0.$$

However, only integrable ones have the universal property of possessing several exact multi-soliton solutions which reflect perfectly nonlinear elastic interactions between individual solitons. Thus, we now consider the two-soliton problem [26] as an example. Using certain similar analytical methods, the exact two-soliton solution to equation (4) is given by

$$u(t, x) = \frac{c_1}{2}\,\mathrm{sech}^2\left(\frac{\sqrt{c_1}}{2}(x - x_1 - c_1 t)\right) + \frac{c_2}{2}\,\mathrm{sech}^2\left(\frac{\sqrt{c_2}}{2}(x - x_2 - c_2 t)\right),$$

where c_1 and c_2 denote the speeds of two individual solitons, respectively. From this expression, we know that the width of the soliton is inversely proportional to the square root of the wave speed for the KdV equation. Assuming $c_1 \gg c_2$ without loss of generality, if such two solitons are well separated with the taller (and thus narrower) to the left of the shorter, then the taller soliton travels faster to the right and would interact nonlinearly and collide elastically with the shorter one [1, 29, 30].

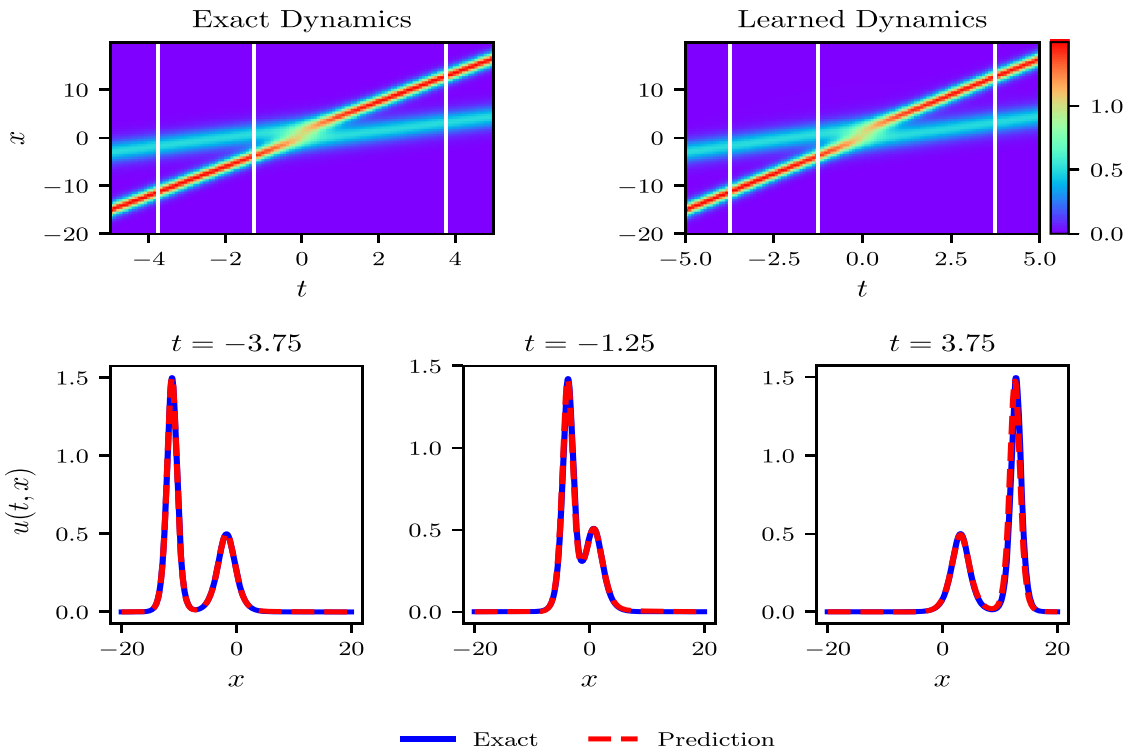

Figure 3. The KdV equation. Top: a two-soliton solution to the KdV equation is compared to the corresponding predicted solution to the learned equation (right panel). The model correctly exhibits the dynamics behavior and accurately reproduces the solution with a relative $\mathbb{L}_2$ error of 7.39×10^{-2}. Bottom: the comparison of the predicted solutions and exact solutions which correspond to the three temporal snapshots is presented.

As an example, an initial solution is given explicitly by

$$u_0(x) = \frac{3}{2}\,\mathrm{sech}^2\left(\frac{\sqrt{3}}{2}(x+15)\right) + \frac{1}{2}\,\mathrm{sech}^2\left(\frac{1}{2}(x+3)\right). \tag{6}$$

Using the above same spectral method, starting from the initial condition (6), we use the Chebfun package [27] with a Fourier spatial discretization with 512 modes and a 4th-order explicit RK integrator with time-step size 1×10^{-4}, and then integrate the equation up to the final instant $t = 5$. The solution is saved every $\Delta t = 0.05$ to give us totally 201 snapshots. We generate a smaller training dataset out of this data by randomly sub-sampling $N_u = 100$ initial-boundary data and $N_f = 10\,000$ collocation points.

Figure 3 demonstrates the evolution of two KdV solitons with different amplitudes, which enables the unique determination of the governing equation [31]. Specifically, given a set of initial and boundary data points, we try to learn the unknown solution $u(t, x)$ by training the network using the loss function (3). The top panel of figure 3 compares between the exact dynamics and the predicted solution. Initially, we have two clearly separated solitons. Then, they lose their identities in certain sort and merge into a composite structure during the interaction. Numerical simulations of the process show that a lower wave hump is formed in the interaction region. The result indicates that it is a nonlinear superposition of shifted counterparts which distinguishes from some other simple solitary traveling waves. After a while, these two solitons emerge from the interaction again. The model achieves a relative $\mathbb{L}_2$ error of size 7.39% in a runtime of approximately half an hour. We can see a more detailed assessment of the predicted solution in the bottom panel of figure 3. We present a comparison between the exact solutions and the predicted solutions at different instants $t = -3.75, -1.25, 3.75$. From the bottom of figure 3, we see that the wave patterns they produced match with the exact solutions well.

From figure 10, we can observe the elastic collision of two individual solitons with different amplitudes better.

In addition, if the speeds of these two solitons are close, i.e. $0 < \frac{c_1 - c_2}{c_1 + c_2} \ll 1$, the solitons will exchange their sizes and speeds at certain much long distance and consequently avoid the collision [32]. For instance, we consider the initial condition

$$u_0(x) = \frac{1.01}{2}\,\mathrm{sech}^2\left(\frac{\sqrt{1.01}}{2}(x+12)\right) + \frac{1}{2}\,\mathrm{sech}^2\left(\frac{1}{2}(x-2)\right), \tag{7}$$

and adopt the same data generation and sampling method.

Then, from figure 5, we can just observe that the two solitons never cross, but rather repulse each other at a long distance. However, the detailed process may be difficult to be observed numerically. See, e.g. [29, 33] for more analytical details.

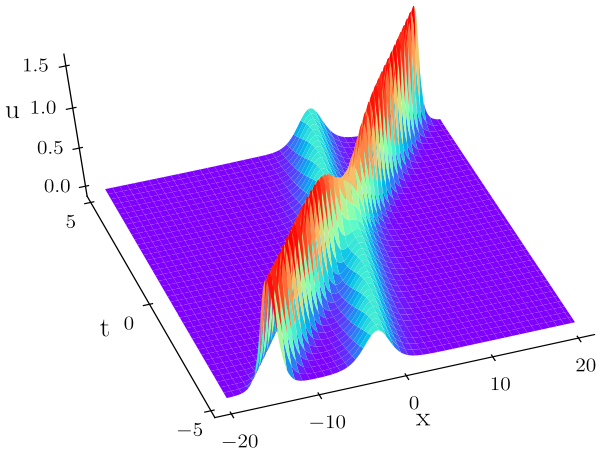

Figure 4. The spatiotemporal behavior of a two-soliton solution to the learned KdV equation.

3. The mKdV equation

The mKdV equation, which can be regarded as the KdV equation with a cubic nonlinearity, is also an integrable model that possesses most of the properties of the KdV equation [34–39] and even has a richer family of solutions including breathers. By the way, it can be obtained from the KdV equation by the Miura transformation.

3.1. One-soliton solution

First, we consider the one-soliton solution to the mKdV equation along with Dirichlet boundary conditions read as

$$\begin{cases} u_t + 6u^2u_x + u_{xxx} = 0,\ x \in [-20,\ 20],\ t \in [-5,\ 5], \\ u(t_0,\ x) = u_0(x) = \sqrt{3}\ \mathrm{sech}\,(\sqrt{3}\,(x + 15)), \\ u(t,\ -20) = u(t,\ 20) = 0. \end{cases} \tag{8}$$

Obviously, we know that $\mathcal{N} = -6u^2u_x - u_{xxx}$ in this case.

To obtain the training and testing data, we simulate equation (8) using the spectral method. Starting from the initial condition, we use the Chebfun package [27] with a Fourier spatial discretization with 512 modes and a 4th-order explicit RK integrator with time-step size 1×10^{-4}, and then integrate the equation up to the final instant $t = 5$. The solution is saved every $\Delta t = 0.05$ to give us totally 201 snapshots. We generate a smaller training data subset by randomly sub-sampling $N_u = 100$ initial and boundary data and $N_f = 10\,000$ collocation points.

Specifically, given a set of initial and boundary data, we attempt to parameterize the solution $u(t,\ x)$ by training the network using the loss function (3). In figure 6, we graphically show the wave profile of a one-soliton solution to the the mKdV equation (8). The top panel of figure 6 compares between the exact dynamics and the predicted solution. The model achieves a relative $\mathbb{L}_2$ error of size 4.57% in a runtime of about 13 minutes. From the viewpoint of training time, the mKdV equation is more complicated compared with the KdV equation obviously. We can see a more detailed assessment in the bottom panel of figure 6. We present a comparison between the exact solutions and the predicted solutions at different points $t = -3.75,\ -1.25,\ 3.75$.

3.2. Breather solution

Now, we consider the breather solution, which is not only spatially localized but also time periodic, to the mKdV equation:

$$\begin{cases} u_t + 6u^2u_x + u_{xxx} = 0,\ x \in [-20,\ 20],\ t \in [-0.3,\ 0.3], \\ u(t_0,\ x) = u_0(x), \\ u(t,\ -20) = u(t,\ 20) = 0. \end{cases} \tag{9}$$

One could obtain the exact breather solution using some analytical methods [40]:

$$\begin{aligned} u(t,\ x) &= 2\partial_x\left[\arctan\left(\frac{\beta}{\alpha}\frac{\sin(\alpha(x+\delta t))}{\cosh(\beta(x+\gamma t))}\right)\right] \\ &= 2\beta\ \mathrm{sech}(\beta(x+\gamma t)) \\ &\times\left[\frac{\cos(\alpha(x+\delta t)) - (\beta/\alpha)\sin(\alpha(x+\delta t))\tanh(\beta(x+\gamma t))}{1 + (\beta/\alpha)^2\sin^2(\alpha(x+\delta t))\mathrm{sech}^2(\beta(x+\gamma t))}\right], \end{aligned}$$

with $\delta = \alpha^2 - 3\beta^2$ and $\gamma = 3\alpha^2 - \beta^2$, where α and β are arbitrary constants.

When $\alpha = 1.5$ and $\beta = 1.0$, we generate the data of 201 snapshots directly on the regular space–time grid every $\Delta t = 0.003$. We generate a smaller training data subset scattered in space and time by randomly sub-sampling $N_u = 100$ initial data and $N_f = 10\,000$ collocation points. Specifically, given a set of initial and boundary data, we try to learn the solution $u(t,\ x)$ by training all learnable parameters of the network. Figure 7 demonstrates the evolution of the breather solution within about a time period (9). The top panel of figure 7 compares between the exact dynamics and the predicted solution. The model achieves a relative $\mathbb{L}_2$ error of size 1.05% in a runtime of about 2.2 h. We can see a more detailed assessment in the bottom panel of figure 7. We present a comparison between the exact solutions and the predicted solutions at different points $t = -0.22,\ 0,\ 0.23$. From figure 7, we observe that the model exactly reproduces the breather pattern.

4. The KdV–Burgers equation

The KdV–Burgers equation is often utilized for a large number of nonlinear systems because this model has damping and dispersion terms [41–43]. Specifically, we consider the KdV–Burgers equation with Dirichlet boundary conditions given by

$$\begin{cases} u_t + uu_x - \alpha u_{xx} - \beta u_{xxx} = 0,\ x \in [-40,\ 40],\ t \in [-5,\ 5], \\ u(t_0,\ x) = u_0(x), \\ u(t,\ -40) = u(t,\ 40) = 0, \end{cases} \tag{10}$$

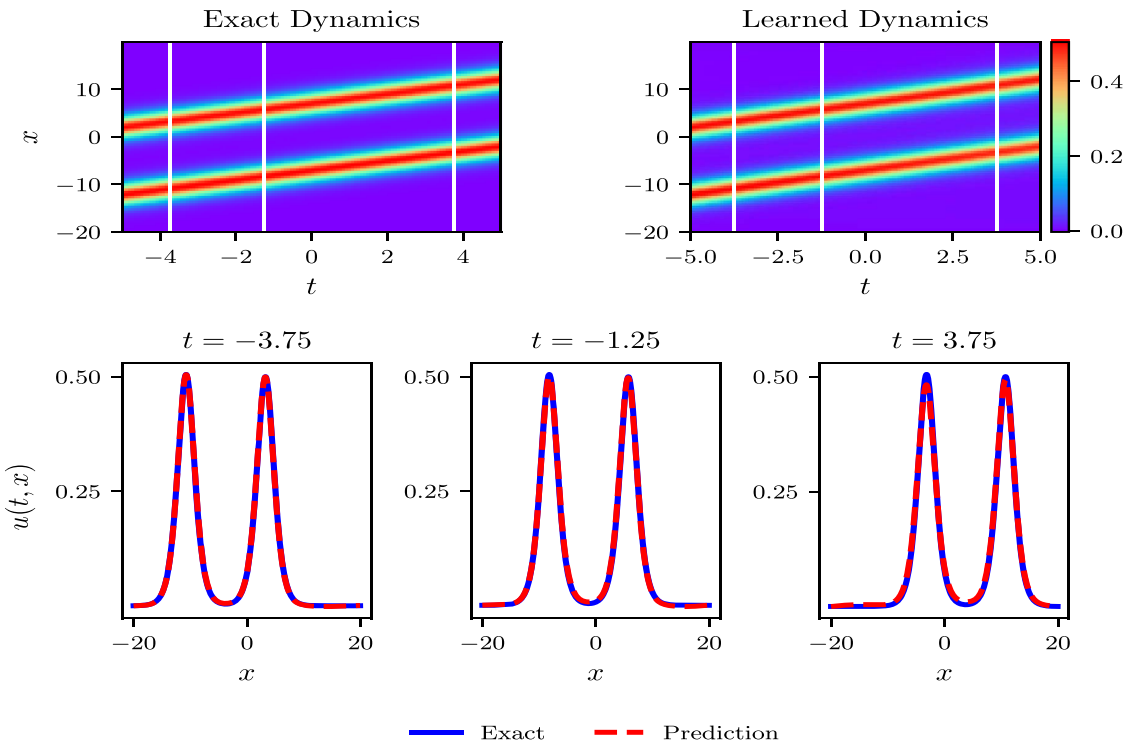

Figure 5. The KdV equation. Top: another two-soliton solution to the KdV equation (left panel) is compared to the predicted solution to the learned equation. The model correctly exhibits the dynamics behavior and accurately reproduces the solution with a relative $\mathbb{L}_2$ error of 2.53×10^{-2}. Bottom: the comparison of the predicted solutions and exact solutions is presented. The model training took about 7.5 min.

where α and β are constants. In this case, $\mathcal{N} = -uu_x + \alpha u_{xx} + \beta u_{xxx}$.

The exact one-soliton solution, that is actually a kink, is obtained:

$$u(t, x) = \frac{3\alpha^2}{25\beta}\left(2 + 2\tanh\frac{z}{2} - \operatorname{sech}^2\frac{z}{2}\right),$$

with $z = \frac{\alpha}{5\beta}\left(x - \frac{6\alpha^2}{25\beta}t\right)$, where α and β are constants.

When $\alpha = 1.0$ and $\beta = -1.0$, we generate the data. In this case, we just sample the data on the regular space–time grid every $\Delta t = 0.05$ and finally obtain totally 201 snapshots. Out of this data, we generate a smaller training data subset by randomly sub-sampling $N_u = 100$ initial data and $N_f = 10\,000$ collocation points. Figure 8 summarizes our result for the kink solution to the KdV–Burgers equation. The top panel of figure 8 compares between the exact dynamics and the predicted spatiotemporal solution and the resulting prediction error is measured at 8.08×10^{-3} in the relative $\mathbb{L}_2$-norm with a runtime of about one and half a minute. More detailed assessments are presented in the middle and bottom panels of figure 8. Moreover, we present a comparison between the exact solutions and the predicted solutions at different time instants $t = -3.75, -1.25, 3.75$. This model can accurately capture the kink dynamics behavior of the KdV–Burgers equation.

5. The STO equation

The STO equation has important applications in many scientific areas. It has been investigated using different analytic methods, such as the Cole–Hopf transformation and Hirota's bilinear method. Here, we consider the STO equation [44] with the Dirichlet boundary condition given by:

$$\begin{cases} u_t + 3\alpha u_x^2 + 3\alpha u^2 u_x + 3\alpha u u_{xx} + \alpha u_{xxx} = 0,\ x \in [-40, 40],\ t \in [-5, 5], \\ u(t_0, x) = u_0(x), \\ u(t, -40) = a,\ u(t, 40) = b, \end{cases} \tag{11}$$

where α is an arbitrary constant, and a, b are fixed values which can be easily obtained given an initial condition. In this case, $\mathcal{N} = -3\alpha u_x^2 - 3\alpha u^2 u_x - 3\alpha u u_{xx} - \alpha u_{xxx}$.

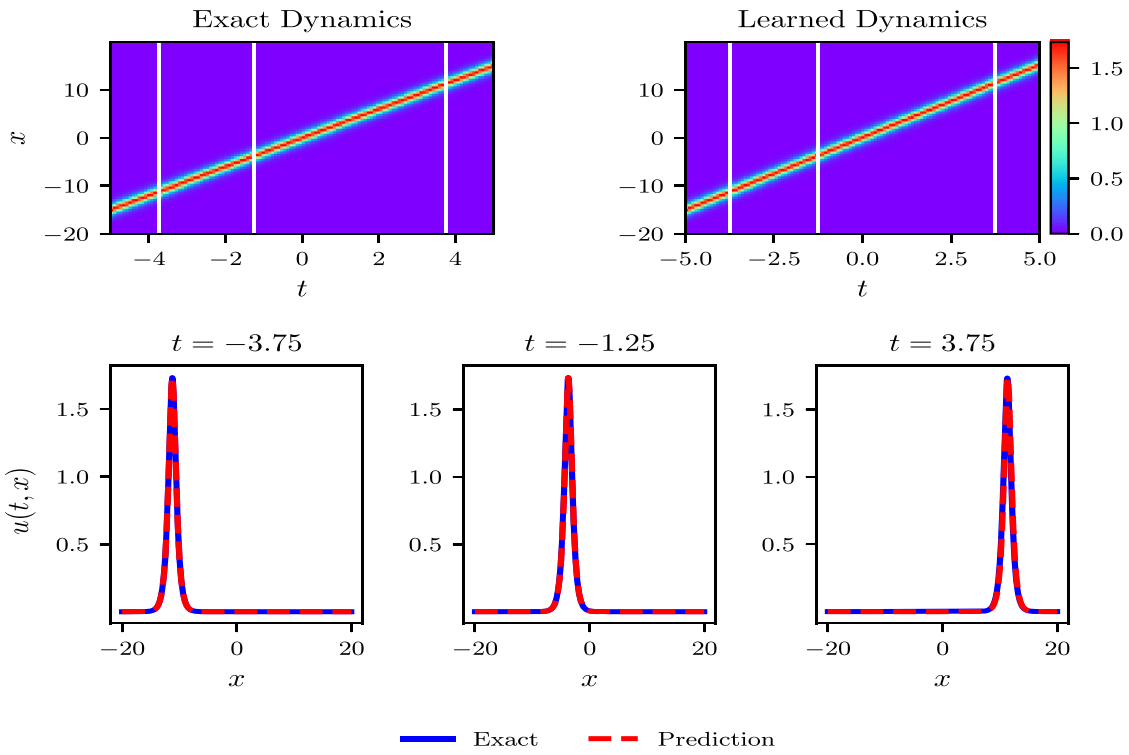

Figure 6. The mKdV equation. Top: a one-soliton solution to the mKdV equation (left panel) is compared to the predicted solution to the learned equation. The model correctly exhibits the dynamics behavior and accurately reproduces the solution with a relative $\mathbb{L}_2$ error of 4.57×10^{-2}. Bottom: the comparison of the predicted solutions and exact solutions corresponding to the three temporal snapshots is given.

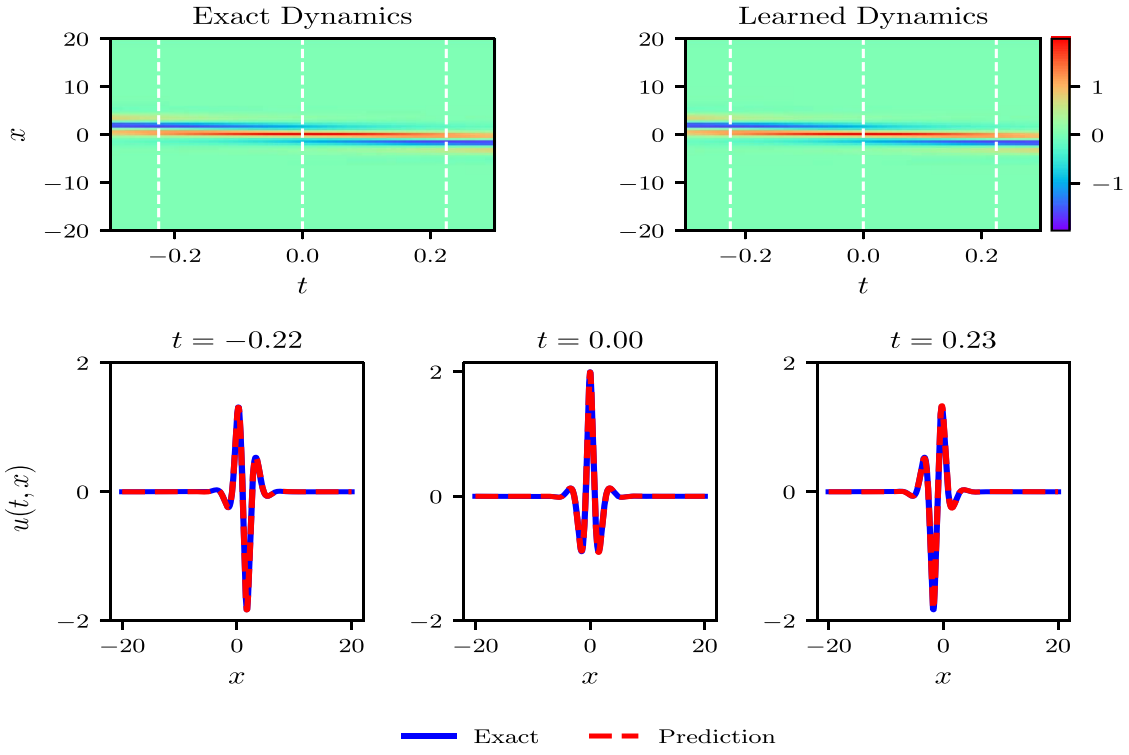

Figure 7. The mKdV equation. Top: a breather solution to the mKdV equation (left panel) is compared to the predicted solution to the learned equation. The model correctly exhibits the dynamics behavior and accurately reproduces the solution with a relative $\mathbb{L}_2$ error of 1.05×10^{-2}. Bottom: the comparison of the predicted solutions and exact solutions is presented.

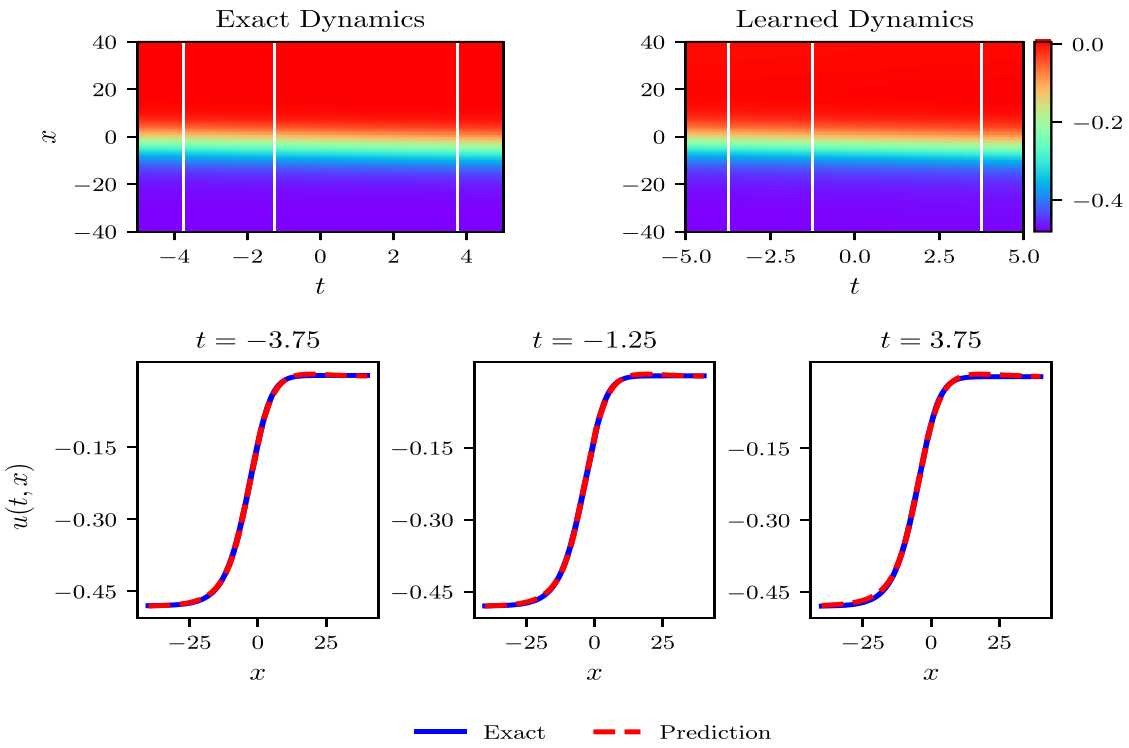

Figure 8. The KdV–Burgers equation. Top: a one-kink solution to the KdVB equation (left panel) is compared to the predicted solution to the learned equation. The model correctly exhibits the dynamics behavior and accurately reproduces the solution with a relative $\mathbb{L}_2$ error of 8.08×10^{-3}. Bottom: the comparison of the predicted solutions and exact solutions is presented.

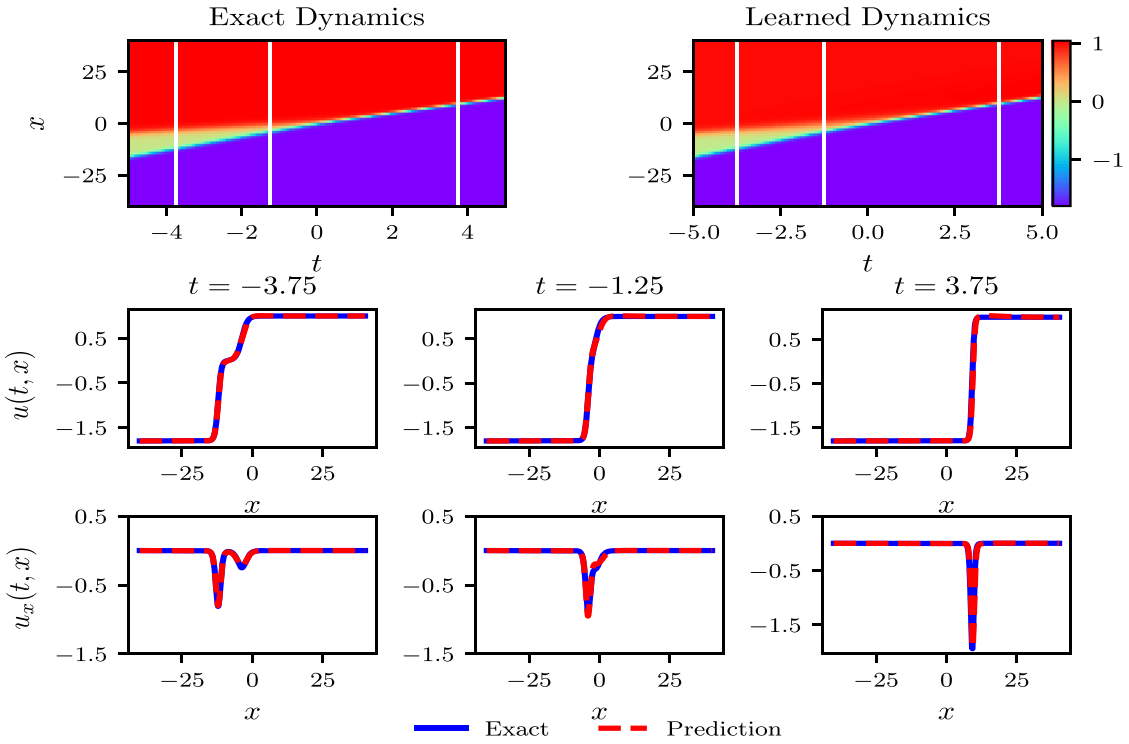

Figure 9. The soliton fusion phenomenon of the STO equation. Top: a solution to the STO equation (left panel) is compared to the predicted solution to the learned equation. The model correctly exhibits the dynamics behavior and accurately reproduces the solution with a relative $\mathbb{L}_2$ error of 1.61×10^{-2}. Middle: the comparison of the predicted solutions and exact solutions is presented. Bottom: the comparison of the corresponding predicted solutions and exact solutions of the potential $-u_x$ is also given.

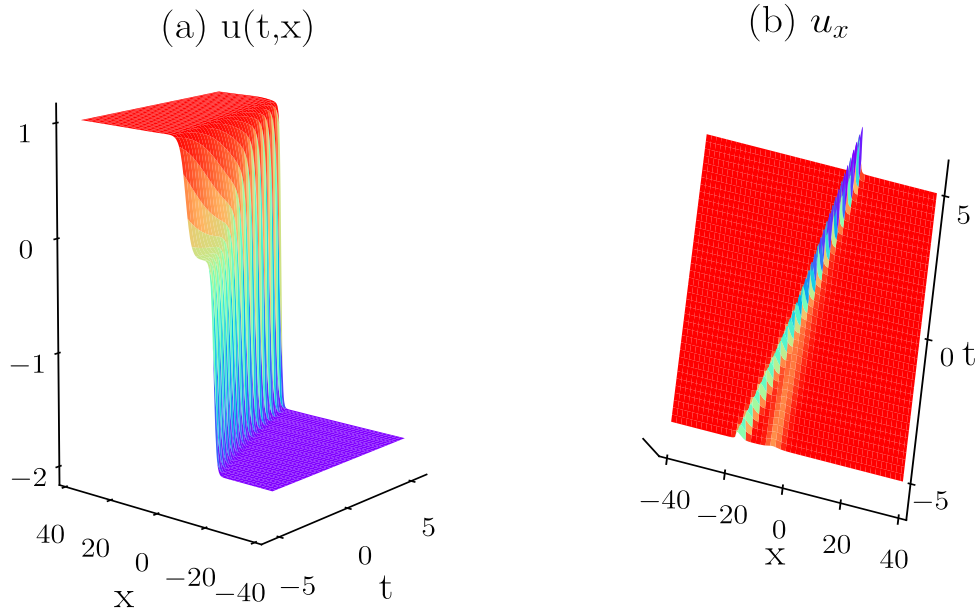

Figure 10. The soliton fusion pattern of the STO equation. (a) The spatiotemporal behavior of the reconstructed solution; (b) the spatiotemporal dynamics of the corresponding potential.

An exact soliton solution is obtained using certain method mentioned above:

$$u(t,x)=\frac{k_1e^{k_1(x-\alpha k_1^2t)}+k_2e^{k_2(x-\alpha k_2^2t)}}{1+e^{k_1(x-\alpha k_1^2t)}+e^{k_2(x-\alpha k_2^2t)}}.$$

5.1. Soliton fusion

The soliton fusion phenomenon is a resonance-like inelastic interaction where two or more solitons fuse into one single structure or less solitons, that is to say, the total number of solitons is not conserved.

Specifically, when $\alpha=1.0$ and $k_1=-1.8$, $k_2=1.0$, we obtain the solution data. In this case, we sample the data on the regular space–time grid every $\Delta t=0.05$ and finally obtain totally 201 snapshots. Out of this data, we generate a smaller training data subset by randomly sub-sampling $N_u=100$ initial-boundary data and $N_f=10\,000$ collocation points. Specifically, given a set of initial and boundary data, we try to learn the solution $u(t,x)$ by tuning all parameters of the network. Figure 9 graphically shows the evolution of the soliton fusion phenomena of the the STO equation (11). The top panel of figure 9 compares between the exact dynamics and the predicted spatiotemporal solution. The model achieves a relative $\mathbb{L}_2$ error of size 1.61% in a runtime of approximately 10 minutes. More detailed assessments are presented in the middle and bottom panels of figure 9. We present a comparison between the exact solutions and the predicted solutions at different time points $t=-3.75$, -1.25, 3.75.

From figure 4, we can more clearly observe that two solitons with different speeds fuse into a single soliton with a larger amplitude.

5.2. Soliton fission

Now, we consider a sort of inverse of the fusion process, namely, one or several solitons may crack into two or more solitons.

Note that, we reset $x\in[-60,20]$ and $t\in[0,4]$ in this case. When $\alpha=-1.0$ and $k_1=-1.8$, $k_2=-1.0$, we obtain the data. In this case, we just sample the data on the regular grid every $\Delta t=0.008$ from $t=0$ up to the final instant $t=4$ and finally obtain totally 501 snapshots. Out of this data, we generate a smaller training data subset by randomly sub-sampling $N_u=200$ initial-boundary data and $N_f=20\,000$ collocation points.

For this soliton fission case, the $\sin(x)$ activation is often not good, thus we choose the $\tanh(x)$ function as the activation. Specifically, given a set of initial and boundary data, we try to fit the solution $u(t,x)$ by training the network using the loss function (3). Figure 11 graphically shows the evolution of the soliton fission process of the the STO equation (11). The top panel of figure 11 compares between the exact dynamics and the predicted spatiotemporal solution. The model achieves a relative $\mathbb{L}_2$ error of size 2.41% in a runtime of approximately 11 min. More detailed assessments are presented in the middle and bottom panels of figure 11. We present a comparison between the exact solutions and the predicted solutions at different points $t=0.5$, 1.5, 3.5.

This model approximately reconstructs the exact solution from the coarse-grained sampled data. However, from the middle and bottom panels of figure 11, it obviously can not exhibit the vicinity of wave humps well. One could devise more sophisticated sampling strategies to enable adaptive refinement, for instance, by tracking the curvature of the solution. This will be further investigated in the future research.

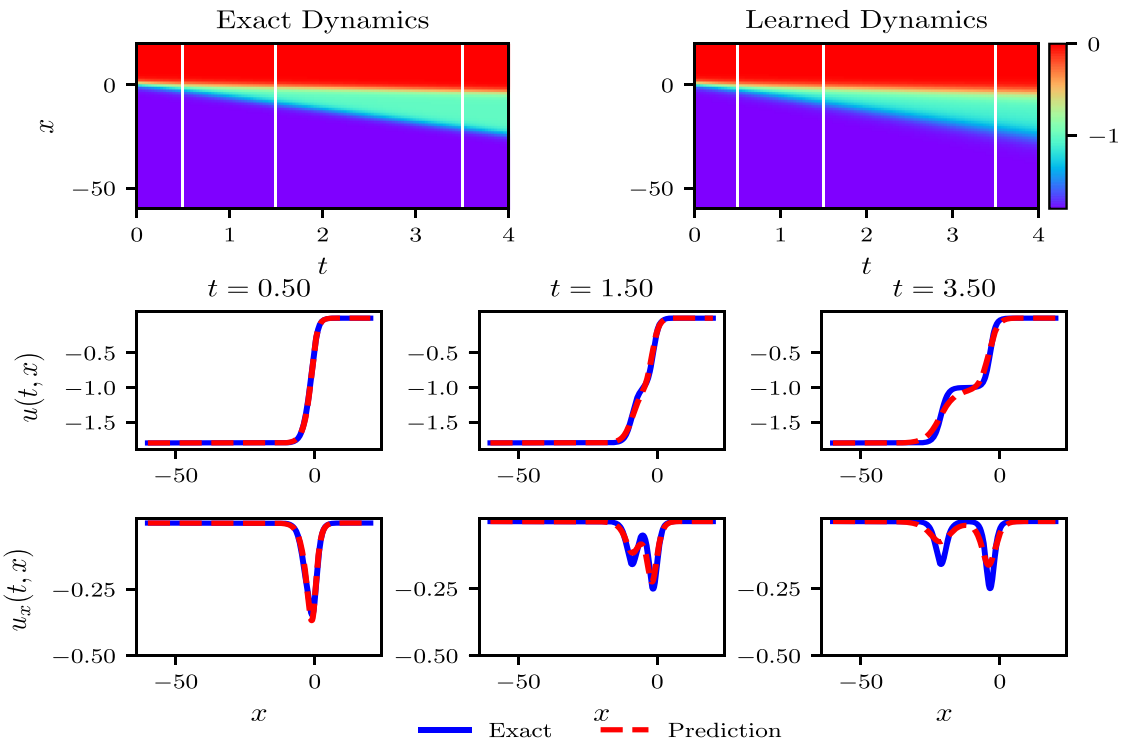

Figure 11. The soliton fission phenomenon of the STO equation. Top: a solution to the STO equation (left panel) is compared to the predicted solution to the learned equation. The model approximately exhibits the dynamics behavior and reproduces the solution with a relative $\mathbb{L}_2$ error of 2.41×10^{-2}. Middle: the comparison of the predicted solutions and exact solutions is presented. Bottom: the comparison of the corresponding predicted solutions and exact solutions of the potential is also given.

6. Remarks and discussion

Deep learning offers a quite different approach for modeling these dynamical behaviors by using the training data to parameterize the solution manifold itself; in other words, it learns both the intrinsic features and their interactions from data collected from experiments and simulations. In this paper, we present a neural network framework for extracting soliton dynamics of evolution equations from the spatio-temporal data. The framework provides a universal treatment of $(1 + 1)$-dimensional third-order nonlinear evolution equations. Specifically, we outline how different categories of soliton solutions (e.g. general soliton solutions, breathers and kinks) to the equations come about due to different choices of initial and boundary data. The results show that the model could recover the different soliton behaviors of these equations fairly well.

Note that a low loss value is a necessary but not sufficient condition for stable training and accurate prediction. For the soliton fission case in the previous section, in particular, the model with low training loss exhibits relatively poor stability and prediction result. In addition, soliton behaviors under certain small perturbations have been studied to some extent [30, 45, 46]. Correspondingly, it is very interesting to extend to the stability of solitons with training the neural network with noisy data. These remain important areas of exploration for future work.

Acknowledgments

The first author would like to express his sincere thanks to Tao Xu for his valuable comments and excellent suggestions on this work. The authors gratefully acknowledge the support of the National Natural Science Foundation of China (No. 11675054), the Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (Grant No. ZF1213) and the Science and Technology Commission of Shanghai Municipality (No. 18dz2271000).

References

[1] Zabusky N J and Kruskal M D 1965 *Phys. Rev. Lett.* **15** 240–3
[2] Craig W, Guyenne P, Hammack J, Henderson D and Sulem C 2006 *Phys. Fluids* **18** 057106
[3] Bongard J and Lipson H 2007 *Proc. Natl Acad. Sci. USA* **104** 9943–8
[4] Raissi M, Perdikaris P and Karniadakis G E 2017 *J. Comput. Phys.* **348** 683–93
[5] Raissi M and Karniadakis G E 2018 *J. Comput. Phys.* **357** 125–41
[6] Lagaris I E, Likas A and Fotiadis D I 1998 *IEEE Trans. Neural Networks* **9** 987–1000
[7] Yadav N, Yadav A and Kumar M 2015 *An Introduction to Neural Network Methods for Differential Equations* (Berlin: Springer)
[8] Sirignano J and Spiliopoulos K 2018 *J. Comput. Phys.* **375** 1339–64

[9] Han J, Jentzen A and Weinan E 2018 *Proc. Natl Acad. Sci. USA* **115** 8505–10
[10] Bar-Sinai Y, Hoyer S, Hickey J and Brenner M P 2019 *Proc. Natl Acad. Sci. USA* **116** 15344–9
[11] Raissi M 2018 *J. Mach. Learn. Res.* **19** 932–55
[12] Raissi M, Perdikaris P and Karniadakis G E 2019 *J. Comput. Phys.* **378** 686–707
[13] Lu L, Meng X, Mao Z and Karniadakis G E 2019 DeepXDE: A deep learning library for solving differential equations (arXiv:1907.04502)
[14] Abadi M *et al* 2016 *12th USENIX Symp. on Operating Systems Design and Implementation* vol 16, pp 265–83
[15] Baydin A G, Pearlmutter B A, Radul A A and Siskind J M 2018 *J. Mach. Learn. Res.* **18** 1–43
[16] Choromanska A, Henaff M, Mathieu M, Arous G B and Lecun Y 2015 *Proc. 18 Int. Conf. on Artificial Intelligence and Statistics, PMLR* vol 38 pp 192–204
[17] Raghu M, Poole B, Kleinberg J, Ganguli S and Sohl-Dickstein J 2017 *Proc. 34th Int. Conf. on Machine Learning, PMLR* vol 70 pp 2847–54
[18] Liu D C and Nocedal J 1989 *Math. Program.* **45** 503–28
[19] Kingma D P and Ba J 2015 *Int. Conf. on Learning Representations (ICLR)*
[20] Korteweg D J and de Vries G 1895 *Phil. Mag.* **539** 422–43
[21] Gardner C S, Greene J M, Kruskal M D and Miura R M 1967 *Phys. Rev. Lett.* **19** 1095–7
[22] Hirota R 1971 *Phys. Rev. Lett.* **27** 1192–4
[23] Bona J L and Smith R 1975 *Phil. Trans. R. Soc.* A **278** 555–601
[24] Eckhaus W and Schuur P 1983 *Math. Methods Appl. Sci.* **5** 97–116
[25] Wadati M and Toda M 1972 *J. Phys. Soc. Japan* **32** 1403–11
[26] Gardner C S, Greene J M, Kruskal M D and Miura R M 1974 *Commun. Pure Appl. Math.* **27** 97–133
[27] Driscoll T A, Hale N and Trefethen L N 2014 *Chebfun Guide* (Oxford: Pafnuty Publications)
[28] Stein M L 1987 *Technometrics* **29** 143–51
[29] Lax P D 1968 *Commun. Pure Appl. Math.* **21** 467–90
[30] Tao T 2009 *Bull. Am. Math. Soc.* **46** 1–33
[31] Rudy S H, Brunton S L, Proctor J L and Kutz J N 2017 *Sci. Adv.* **3** e1602614
[32] Martel Y 2019 *Proc. Int. Congress of Mathematicians (ICM 2018)* pp 2439–66
[33] LeVeque R 1987 *SIAM J. Appl. Math.* **47** 254–62
[34] Wadati M 1972 *J. Phys. Soc. Japan* **32** 1681
[35] Hirota R 1972 *J. Phys. Soc. Japan* **33** 1456–8
[36] Wadati M 1973 *J. Phys. Soc. Japan* **34** 1289–96
[37] Fonseca G, Linares F and Ponce G 1999 *Commun. PDE* **24** 683–705
[38] Hayashi N and Naumkin P 2001 *Math. Phys. Anal. Geom.* **4** 197–201
[39] Germain P, Pusateri F and Rousset F 2016 *Adv. Math.* **299** 271–330
[40] Alejo M A and Muñoz C 2013 *Commun. Math. Phys.* **324** 233–62
[41] Johnson R S 1970 *J. Fluid Mech.* **42** 49–60
[42] Canosa J and Gazdag J 1977 *J. Comput. Phys.* **23** 393–403
[43] Ahmad H, Seadawy A R and Khan T A 2020 *Phys. Scr.* **95** 045210
[44] Wang S, Tang X and Lou S 2004 *Chaos Solitons Fractals* **21** 231–9
[45] Benjamin T B 1972 *Proc. R. Soc.* A **328** 153–83
[46] Bona J 1975 *Proc. R. Soc.* A **344** 363–74