entropy

MDPI

Article

Learning Traveling Solitary Waves Using Separable Gaussian Neural Networks

Siyuan Xing [1,*] and Efstathios G. Charalampidis [2,*]

[1] Department of Mechanical Engineering, California Polytechnic State University, San Luis Obispo, CA 93407-0403, USA
[2] Mathematics Department, California Polytechnic State University, San Luis Obispo, CA 93407-0403, USA
* Correspondence: sixing@calpoly.edu (S.X.); echarala@calpoly.edu (E.G.C.)

Abstract: In this paper, we apply a machine-learning approach to learn traveling solitary waves across various physical systems that are described by families of partial differential equations (PDEs). Our approach integrates a novel interpretable neural network (NN) architecture, called Separable Gaussian Neural Networks (SGNN) into the framework of Physics-Informed Neural Networks (PINNs). Unlike the traditional PINNs that treat spatial and temporal data as independent inputs, the present method leverages wave characteristics to transform data into the so-called co-traveling wave frame. This reformulation effectively addresses the issue of propagation failure in PINNs when applied to large computational domains. Here, the SGNN architecture demonstrates robust approximation capabilities for single-peakon, multi-peakon, and stationary solutions (known as "leftons") within the (1+1)-dimensional, *b*-family of PDEs. In addition, we expand our investigations, and explore not only peakon solutions in the *ab*-family but also compacton solutions in (2+1)-dimensional, Rosenau-Hyman family of PDEs. A comparative analysis with multi-layer perceptron (MLP) reveals that SGNN achieves comparable accuracy with fewer than a tenth of the neurons, underscoring its efficiency and potential for broader application in solving complex nonlinear PDEs.

Keywords: traveling waves; solitons; peakons; compactons; separable gaussian neural networks; physics-informed neural networks

Citation: Xing, S.; Charalampidis, E.G. Learning Traveling Solitary Waves Using Separable Gaussian Neural Networks. *Entropy* **2024**, *26*, 396. https://doi.org/10.3390/e26050396

Academic Editors: Lars English and Faustino Palmero

Received: 19 March 2024
Revised: 26 April 2024
Accepted: 26 April 2024
Published: 30 April 2024

1. Introduction

Physics-informed Neural Networks (PINNs) [1,2] have emerged as a promising data-driven approach to solving partial differential equations (PDEs) by synthesizing data and physical laws. Moreover, they have received considerable traction because they can be efficiently adapted to solving PDEs defined on domains with arbitrary geometry. Remarkable results with PINNs have been achieved across multiple domains and physical situations, such as heat transfer [3], Navier-Stokes [4] and Euler equations [5], nonlinear dynamical lattices [6,7], and medical image processing [8], to name a few.

However, many examples of PINNs are limited to "toy" problems situated in low-dimensional spaces with small spatio-temporal, i.e., computational domains. It has been observed that PINNs often converge to incorrect or trivial solutions across a broad spectrum of problems [9–11] (see also [6] for a case where they fail to respect symmetries). This issue becomes more pronounced in problems with larger domains, where a phenomenon known as propagation failure [12] frequently occurs. This challenge arises because PINNs utilize an unsupervised learning scheme to solve PDEs by minimizing the residual errors of the underlying governing equations. The presence of propagation failure does not ensure convergence to a faithful solution of the physical system at hand, as numerous trivial solutions can also exhibit zero residuals. Therefore, as the learning process attempts to extend the solution from the initial and/or boundary conditions to the interior points, it often becomes "trapped" in regions of solution spaces that contain trivial solutions only. This phenomenon is particularly common when PINNs are applied to solve problems

with large domains. Indicatively, Figure 1 highlights this issue in the Camassa-Holm (CH) equation [13] for a single-peakon solution (Note that the loss function of this example is modified from [14], removing the termination condition at $t = T$).

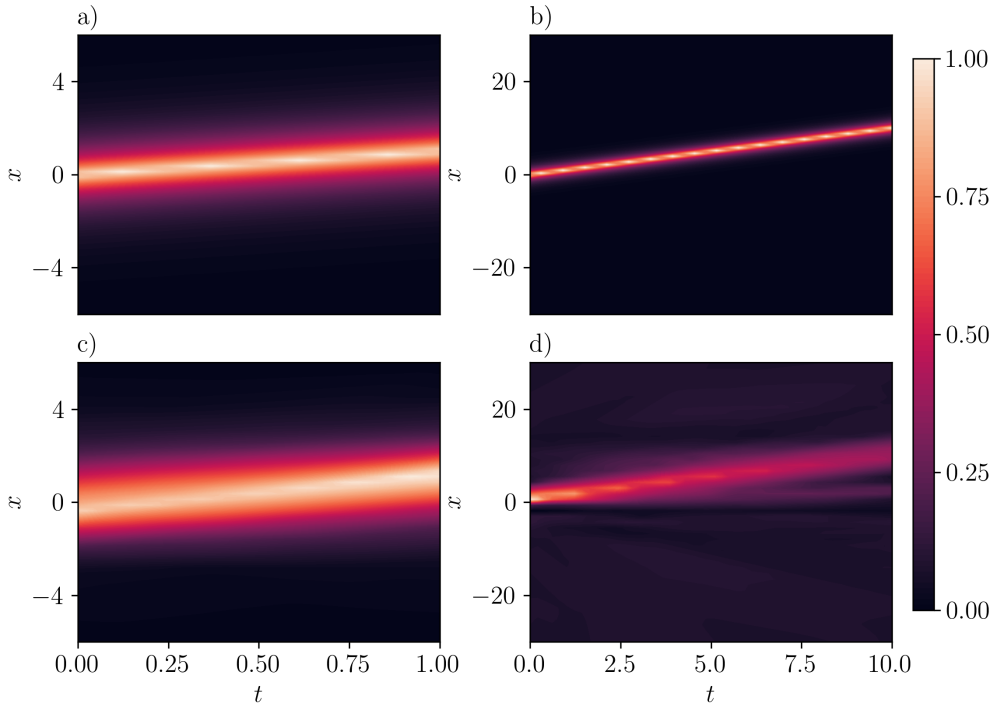

Figure 1. Inference of the spatio-temporal evolution of a peakon in the Camassa-Holm (CH) equation by PINNs. Panels (**a**,**b**) correspond to ground truth, whereas panels (**c**,**d**) to the NN approximation. For small domains, e.g., $[-5, 5] \times [0, 1]$ (panel (**c**)), PINNs are able to roughly capture the correct solution. However, the propagation failure of PINNs occurs when a large spatio-temporal domain, e.g., $[-30, 30] \times [0, 10]$ is utilized, see, panel (**d**). In this case, PINNs converge to a trivial solution.

To address the pathologies of PINNs, multiple methods have been developed including ones that consider embedding Fourier features [15], adaptive sampling [12,16], and those respecting causality [17]. In fact, besides the physical laws embodied in PDEs themselves, the mathematical properties of their solutions can be leveraged too. For example, traveling waves (TWs) to PDEs are solutions of the form $u(x \pm ct)$, where c is their speed (the "$-$" and "+" signs correspond to TWs moving, i.e., traveling to the right and left, respectively). However, a few efforts have been devoted to embedding such mathematical properties of solutions into PINNs (see, [6] for the development of symmetry-preserving PINNs) such that the output of neural networks (NNs) will automatically respect the corresponding features of the solution. This is expected to improve the efficiency in training and increase the opportunity for NNs to converge to correct solutions.

In this paper, we aim to enhance PINNs by pursuing this route. More specifically, we will focus on seeking TW solutions to nonlinear PDEs using PINNs with input transformed into a frame that co-moves with the solution, i.e., co-traveling frame. This idea has been explored in the recent work in [18] in which the characteristics of hyperbolic PDEs are encoded in the network by adding a characteristic layer. Herein, we will use this structure to learn TWs, i.e., solitary waves in multiple families of one and two-dimensional nonlinear PDEs. Those include the b- and ab-families of peakon equations [19,20] which contain the (completely integrable) Camassa-Holm (CH) and Degasperis-Procesi equations [13,21] (see, also [22]), and the Rosenau-Hyman compacton equations [23] (see, also [24]). In addition, a novel interpretable NN—Separable Gaussian Neural Networks (SGNNs) [25]—will be employed to extract solution forms in the sense of generalized Gaussian radial-basis

functions. The description of this network will be deferred to Section 2, along with the discussion about its advantages.

The rest of the paper is organized as follows. In Section 2, we introduce the architecture of SGNN with its input transformed to a co-traveling frame. Our aim here is to integrate the mathematical description of TWs into the framework of PINNs in order to reliably identify TWs to physically relevant PDEs. In Sections 3 and 4, we demonstrate the applicability of the method to the study of peakons in the *b*- and *ab*-families of peakon equations, respectively. Then, we extend this approach in Section 5 to identify 2D compacton configurations. We mention in passing that the architecture can easily predict such higher-dimensional solutions which have not been studied in the realm of PINNs, to the best of our knowledge. At last, we perform an extensive comparison of the two different architectures of PINNs with different network structures in Section 6, where the advantages and disadvantages of SGNN are discussed. We conclude our findings in Section 7, and present future research directions.

2. Methods

2.1. Architecture of SGNN for Traveling Waves

Inspired by [18], a d-dimensional (in space) TW is mapped into a frame that co-moves with it by performing the following coordinate transformation

$$\zeta_i = x_i - c_i t, \;\; i = 1, 2, \ldots, d, \tag{1}$$

where c_i is its (constant) velocity in the i-th dimension. Under such a transformation, a TW becomes a stationary wave in the co-traveling frame. As shown in Figure 2, the coordinates ζ_i $(i = 1, 2, \ldots, d)$ become the input of the SGNN. The coordinates are then divided according to their dimensions, and sequentially fed to the feedforward layers. This results in a number of layers that is equal to the number of spatial dimensions. The neurons of each layer we consider are expressed in terms of generalized univariate Gaussian functions

$$\varphi(\zeta_i, \mu, \sigma) = exp\Big(-|\zeta_i - \mu|^\alpha / \sigma^2\Big), \tag{2}$$

where $\alpha \in \mathbb{R} - \{0\}$. When $\alpha = 2$, φ is the regular univariate radial-basis Gaussian function. In this paper, we will adopt $\alpha = 1$ for peakon solutions, and $\alpha = 2$ for other solutions.

The first hidden layer of SGNN receives a single input: the partial coordinate ζ_1. Subsequent hidden layers take two inputs - the output from the preceding hidden layer, and a coordinate in the traveling frame. The network culminates in a dense output layer, which aggregates the outputs from the final hidden layer. The mathematical representation of SGNN [25] is defined in the form

$$\mathcal{N}_i^{(1)} = \varphi_i^{(1)}(\zeta_1, \mu_i^{(1)}, \sigma_i^{(1)}), \; 1 \le i \le N_1, \tag{3}$$

$$\mathcal{N}_i^{(\ell)} = \varphi_i^{(\ell)}(\zeta_\ell, \mu_i^{(\ell)}, \sigma_i^{(\ell)}) \sum_{j=1}^{N_l} W_{ij}^{(\ell)} \mathcal{N}_j^{(\ell-1)}, \;\; 2 \le \ell \le d, \;\; 1 \le i \le N_l, \tag{4}$$

$$\bar{u}(\mathbf{x}) = \mathcal{N}(\mathbf{x}) = \sum_{j=1}^{N_d} W_j^{(d)} \mathcal{N}_j^{(d)}, \tag{5}$$

where $\mathcal{N}_i^{(l)}$ represents the output of the i-th neuron of the l-th layer, N_l stands for the number of neurons of the l-th layer, and $\bar{u}$ is the output of SGNN. When $d > 2$, the weights of the output layer are set to 1.

Thanks to the separable property of Gaussian radial-basis functions, the forward propagation of such univariate Gaussian functions yields the summation of multiple chains of univariate Gaussian functions, equivalent to the summation of high dimensional

Gaussian radial-basis functions. In other words, the output of an SGNN equals the output of a Gaussian-Radial-Basis-Function Neural Network (GRBFNN) [26] in the form of

$$\bar{u}(\mathbf{x}) = \sum_{j=1} \mathcal{W}_j G_j, \tag{6}$$

where G_j is a d-dimensional Gaussian radial-basis function.

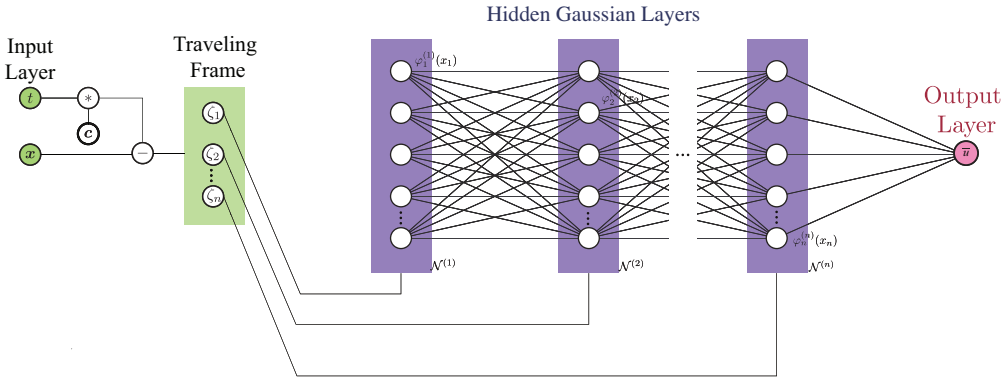

Figure 2. The architecture of SGNN with input transformed into the co-traveling frame whose coordinates (in vector form) are $\boldsymbol{\zeta} = \boldsymbol{x} - \boldsymbol{c}t$, where $\boldsymbol{c}$ represents the velocity of the wave. Different from multi-layer perceptron (MLP), the transformed input is then split and fed sequentially to hidden layers of an SGNN that consist of univariate functions. The multiplication and addition of such univariate functions in feedforward propagation can eventually lead to the summation of a set of multivariate functions used to approximate the solution of a PDE.

The SGNN offers several advantages. Firstly, it is interpretable. The parameters of a neuron depict its local geometrical information (center and width). Without the composition of nonlinear activation functions, a human-interpretable explicit output form of Equation (6) can be obtained, in the sense of Gaussian radial-basis functions. Secondly, SGNN is easier to tune than MLP. This is because the depth of SGNN is identical to the number of dimensions; therefore, the only tunable hyperparameter is the width of each layer. Lastly, it can achieve several-order-of-magnitude more accurate results than MLP when approximating complex functions. The interested reader can consult [25] for detailed comparisons between SGNN and MLP.

2.2. Physics-Informed Machine Learning

The SGNN is adopted to approximate the solution $u(x,t)$ of PDEs in the form

$$u_t + \mathcal{F}[u, u_x, u_{xt}, u_{xx}, \dots] = 0, \tag{7}$$

which is subject to boundary and initial conditions (abbreviated hereafter as BCs and ICs, respectively)

$$\mathcal{B}[u] = 0, \tag{8}$$
$$u(x,0) = f(x). \tag{9}$$

The loss function is defined as

$$\mathcal{L} = \lambda_r \mathcal{L}_r + \lambda_{bc} \mathcal{L}_{bc} + \lambda_{ic} \mathcal{L}_{ic}, \tag{10}$$

where

$$\mathcal{L}_r = \frac{1}{N_r}\sum_{i=1}^{N_r}\left|\mathcal{R}(\boldsymbol{x}_r^i, t_r^i)\right|^2, \tag{11}$$

$$\mathcal{L}_{bc} = \frac{1}{N_{bc}}\sum_{i=1}^{N_{bc}}\left|\mathcal{B}[u](\boldsymbol{x}_{bc}^i, t_{bc}^i)\right|^2, \tag{12}$$

$$\mathcal{L}_{ic} = \frac{1}{N_{ic}}\sum_{i=1}^{N_{ic}}\left|u(\boldsymbol{x}_{ic}^i, 0) - f(\boldsymbol{x}_{ic}^i)\right|^2, \tag{13}$$

and $\lambda_r, \lambda_{bc}, \lambda_{ic}$ are scaling factors. Here, $\mathcal{L}_r$, $\mathcal{L}_{bc}$, and $\mathcal{L}_{ic}$ represent the MSE (mean-squared error) of PDEs, BCs, and ICs, respectively. The collocation points denoted as $\{\boldsymbol{x}_r^i, t_r^i\}$ and $\{\boldsymbol{x}_{bc}^i, t_{bc}^i\}$ are randomly sampled in the domain and on the boundary, respectively. In addition, $\{\boldsymbol{x}_{ic}^i\}$ are spatial points sampled at $t = 0$. Throughout this paper, λ_r is fixed to 1.

2.3. Training Scheme

In the 1D problems presented next, we employ a two-stage training process, initially using the ADAM optimizer [27] followed by the L-BFGS algorithm [28]. This approach allows us to leverage the L-BFGS algorithm's capability to enhance convergence accuracy after the preliminary optimization with ADAM. In contrast, and for the 2D problems at hand, we solely rely on the ADAM optimizer due to the computational demands of running L-BFGS with larger datasets. The training dataset is randomly sampled using the 'Sobol' method, which empirically can yield better results [16]. The validation dataset is created through a method of even partitioning across the domain and boundaries, ensuring comprehensive coverage and testing of the model's predictive capabilities. Throughout the training phase, the SGNNs' weights are initialized based on a uniform random distribution. Additionally, the initial centers of the univariate Gaussian neurons are distributed evenly across the respective dimensions, with their initial widths defined by the distance between adjacent centers.

3. Peakons in *b*-Family

The first model-PDE we consider in this work is the well-known b-family of peakon equations:

$$u_t - u_{xxt} + (b+1)uu_x = bu_xu_{xx} + uu_{xxx}, \tag{14}$$

that was introduced in [19]. It has been proposed as a model for the propagation of shallow water waves [19] with the parameter b related to the Kodama transformation group of shallow water water equations [29,30]. Moreover, Equation (14) contains two completely integrable models for $b = 2$ and $b = 3$, known as the Camassa-Holm equation [13,31] (see, also [22]) and Degasperis-Procesi equation [21], respectively.

The striking feature of the b-family of Equation (14) is that it possesses explicit single-peakon

$$u(x,t) = ce^{-|x-ct|}, \tag{15}$$

and multi-peakon solutions

$$u(x,t) = \sum_{j=1}^{N} p_j(t)e^{-|x-q_j(t)|}, \tag{16}$$

for all values of b, where q_j and p_j are the position and amplitude of j-th peakon with N representing the number of peakons, i.e., $j = 1, \ldots, N$. The peakon solution of Equation (15) (and similarly, its multi-peakon version) is not differentiable at its center, rendering its analytical and numerical study (from the PDE point of view) an extremely challenging task (see [32] for the spectral stability analysis of peakons). It should be noted in passing

that alongside the existence of peakon solutions, the b-family possesses explicit stationary solutions known as "leftons" [32] (and references therein) given by

$$u = A(\cosh(\gamma(x - x_0)))^{-\frac{1}{\gamma}}, \quad \gamma = -\frac{b+1}{2}, \tag{17}$$

where A and x_0 are their amplitude and center, respectively. These solutions also emerge numerically upon propagating Gaussian initial data to Equation (14) for $b < -1$. Even more, the propagation of Gaussian initial data to the b-family with $-1 < b < 1$ results in the emergence of self-similar solutions known as "ramp-cliffs", see [32], and references therein for details.

Having introduced the model of interest, we will use the SGNN to approximate both one-peakon and multi-peakon solutions in the next section.

3.1. Single Peakon

3.1.1. Camassa-Holm ($b = 2$)

We first inspect a one-peakon/one-antipeakon solution in the Camassa-Holm (CH) equation. The computational domain we consider is $\Omega = \{(x, t) : [-20, 20] \times [0, 10]\}$. We adopt periodic BCs, and ICs of the form $u(x, 0) = e^{-|x|}$ (i.e., $c = 1$). For our analysis, we employ a one-layer SGNN with 60 neurons. As both centers and widths are trainable, the total number of trainable parameters is 180.

The data collection process involves the sampling of $2^{12} = 4096$ points within the specified domain. Additionally, we use the 'Sobol' sampling scheme to gather $2^9 = 512$ boundary points, and another 512 spatial points satisfying the initial condition. It should be noted that the number of samples is larger than the number reported in the literature. This increase in sample size is attributed to the comparatively larger domain size in our analysis. We train SGNN for 5000 epochs using ADAM [27], followed by L-BFGS [28] to refine the results. The dataset is divided into 8 mini-batches. The learning rate of ADAM is $1e-2$ for the first 100 epochs, and $1e-3$ for the rest. We report that the mean-squared loss is $8.43e-3$ when training finishes. It should also be noted that this value is scaled by a relatively large scaling factor ($\lambda_{ic} = 1000$) that is selected using trial and error. On the other hand, the mean-squared validation error is much smaller, with a value of $7.21e-6$. The maximum absolute validation error is $3.90e-2$. As illustrated in Figure 3b, the inferred peakon solution with $c = 1.0$ accurately approximates the exact solution with the error getting maximized at the crest of the peakon. The good agreement is also demonstrated in Figure 3c, where the "x" markers stand for the exact solution [cf. Equation (15)], and line for the predicted solution by SGNN at two different instant of times (see, the legend in the figure).

A case corresponding to an anti-peakon solution with $c = -1.0$ is represented in Figure 4. The prediction by SGNN yields a mean-squared loss of $1.94e-11$. This means that the inference of SGNN precisely matches the exact solution. The largest error occurs on the characteristic curve $x + t = -10$, with the magnitude level of $1e-5$.

3.1.2. Other Values of b

We next investigate the emergence of peakons using SGNN for different values of b. Indeed, Figure 5a presents a peakon solution predicted by SGNN with $b = 0.8$ and $c = 1.5$. While the temporal domain remains as $[0, 10]$, the spatial domain is enlarged to $[-30, 30]$ in order to accommodate the rise of velocity, and thus the peakon "fits" in the computational domain over its propagation. As shown in Figure 5b, the prediction matches very well with the exact solution. The mean-squared validation error is $4.09e-6$, and the maximum absolute error is 0.0274. The maximum absolute error appears at the region where the $u(x, t)$ reaches its peak value. The training loss after 5000 epochs is reduced to $9.18e-2$. The waveforms at $t = 0$ and $t = 10$ are depicted in Figure 5c, where lines represent SGNN's prediction, and "x" markers represent the exact solution, respectively. The predicted peakon solution with $b = -1, c = 0.8$ is presented in Figure 6. Likewise, a good

agreement between inference by SGNN and the exact solution $u(x,t) = 0.8e^{-|x-0.8t|}$ is achieved, with a training loss of $3.1e-2$, a mean-squared validation loss of $3.5e-5$, and a maximum absolute validation loss of $5.92e-3$.

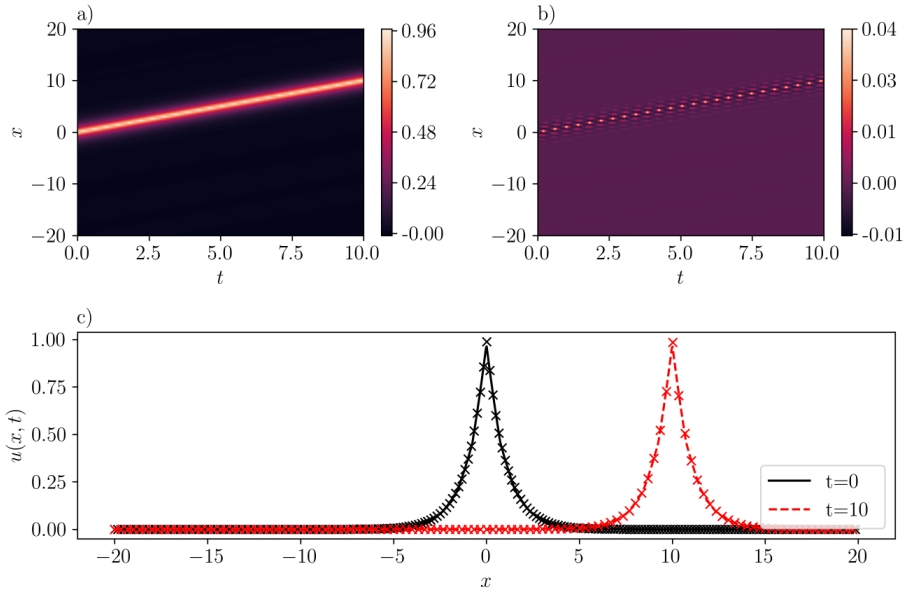

Figure 3. A one-peakon solution in the CH equation ($b = 2$) with $c = 1$. (**a**): $\bar{u}(x,t)$ inferred by SGNN; (**b**): error $e(x,t) = u(x,t) - \bar{u}(x,t)$; (**c**): $\bar{u}(x,t)$ at two time instants. In (**c**), "x" markers represent the exact solution while lines represent the prediction by SGNN. The training loss is $8.43e-3$, with $\lambda_{ic} = 1000$, $\lambda_{bc} = 1$. Validation error: $\|e\|_\infty = 3.90e-2$, $\|e\|_2 = 7.21e-6$.

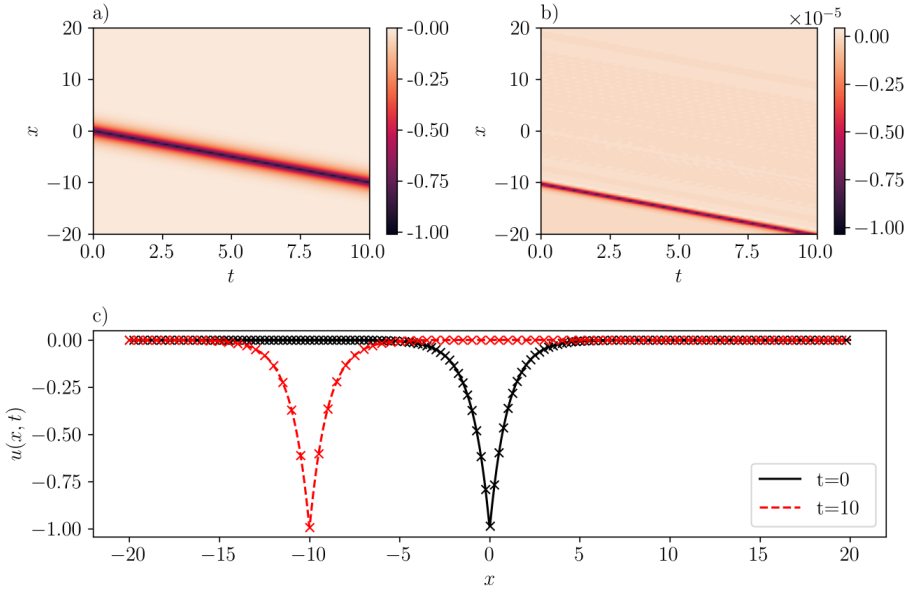

Figure 4. Same as Figure 3 but for an one-antipeakon solution in the CH equation ($b = 2$) with $c = -1$. (**a**): $\bar{u}(x,t)$ inferred by SGNN; (**b**): error $e(x,t) = u(x,t) - \bar{u}(x,t)$; (**c**): $\bar{u}(x,t)$ at two time instants. In (**c**), "x" represents the exact solution while lines represent the prediction by SGNN. The training loss is $1.94e-11$, with $\lambda_{ic} = 1000$, $\lambda_{bc} = 1$. Validation error: $\|e\|_\infty = 1.02e-5$, $\|e\|_2 = 9.59e-13$.

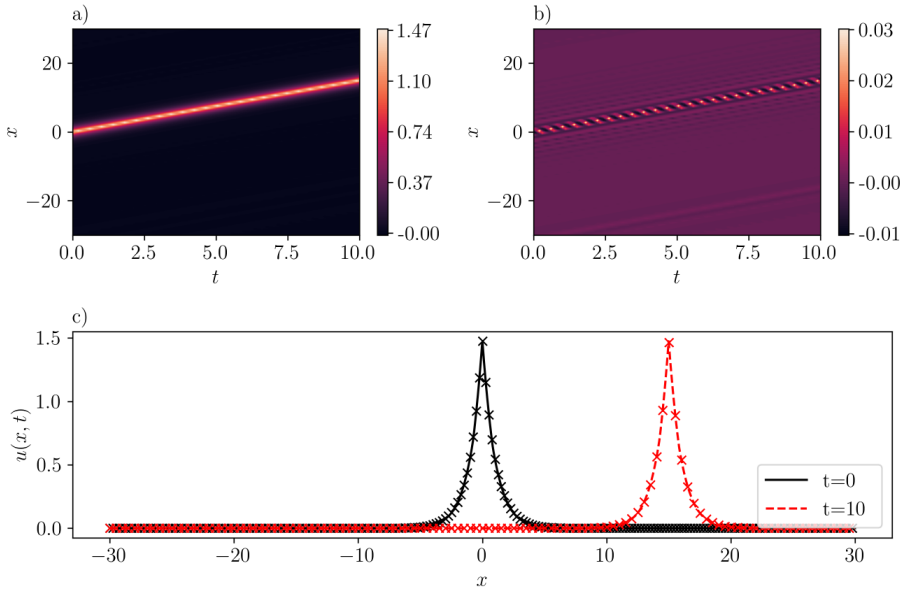

Figure 5. Same as Figure 4 but for a one-peakon solution of the b-family with $b = 0.8$ and $c = 1.5$. (**a**): $\bar{u}(x,t)$ inferred by SGNN; (**b**): error $e(x,t) = u(x,t) - \bar{u}(x,t)$; (**c**): $\bar{u}(x,t)$ at two time instants. In (**c**), "x" represents the analytical solution while curves represent the prediction by SGNN. The training loss is $9.18e-2$, with $\lambda_{ic} = 1000$, $\lambda_{bc} = 1$. Validation error: $\|e\|_\infty = 2.74e-2$, $\|e\|_2 = 4.09e-6$.

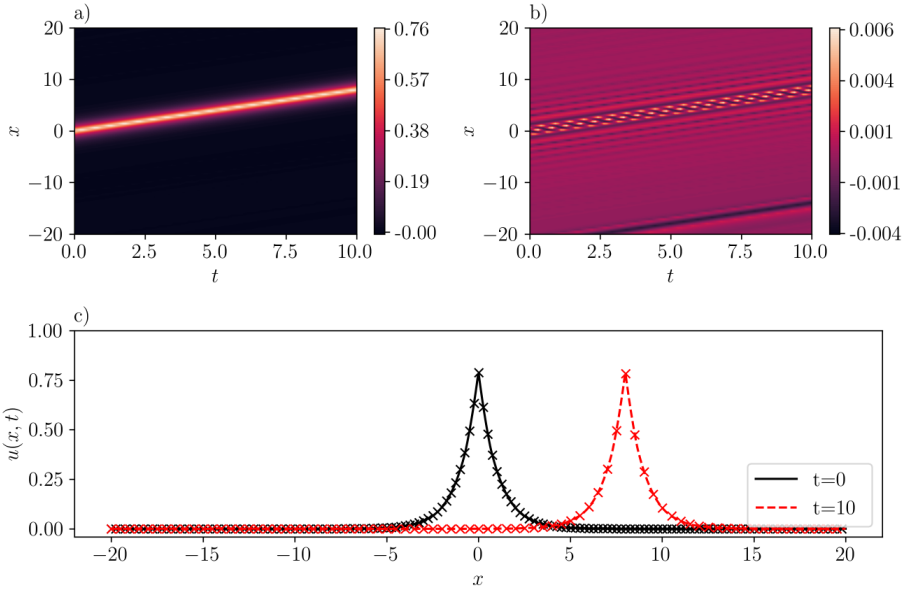

Figure 6. Same as Figure 5 but with $b = -1.0$ and $c = 0.8$. (**a**): $\bar{u}(x,t)$ inferred by SGNN; (**b**): error $e(x,t) = u(x,t) - \bar{u}(x,t)$; (**c**): $\bar{u}(x,t)$ at two time instants. The format in panel (**c**) is the same as in panel (**c**) of Figure 5. Here, the training loss is $3.10e-2$, with $\lambda_{ic} = 10,000$, and $\lambda_{bc} = 1$. Validation error: $\|e\|_\infty = 5.92e-3$, $\|e\|_2 = 3.50e-5$.

3.1.3. Interacting Peakons

Having discussed the prediction of single-peakon (and anti-peakon) solutions in the b-family, we now turn our focus to cases involving two-peakon configurations in the CH equation ($b = 2$), thus emulating their interactions. In particular, we focus on the following three specific scenarios: (1) peakons traveling along the same direction with identical speed, (2) peakons traveling in the same direction but at different velocities, and (3) peakons moving in opposite directions. Given that peakons can travel at varying speeds

and in distinct directions in space (i.e., either left or right), we employ multiple SGNNs to approximate these peakons, allocating one SGNN per peakon. The sum of such SGNNs produces the NN-solution of Equation (14), and the NN structure in this case is shown in Figure 7. During the training stage, the loss functions associated with the PDE and BCs are identical to those in Equations (11) and (12). However, it is necessary to modify the loss function of ICs such that the output of each SGNN at $t = 0$ accurately reflects the corresponding peakon solution at $t = 0$.

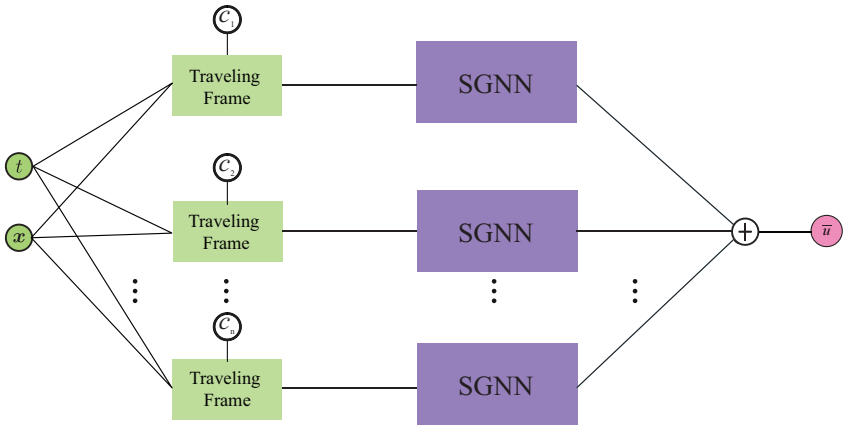

Figure 7. The NN architecture used for the study of multi-peakon configurations.

We inspect the response from $t = 0$ to $t = 10$, within a spatial domain $[-30, 30]$. Two one-layer SGNNs with 40 neurons are used. Each training dataset is generated by randomly sampling $2^{13} = 8192$ collocation points within the domain, and $2^{10} = 1024$ points on the boundary. The dataset is then divided into 8 mini-batches. The results are obtained with 5000 training epochs by ADAM, followed by refinement by L-BFGS (as before). The validation set is generated by uniformly sampling a 50×100 grid in the domain including BCs and ICs.

In Figure 8, two peakons traveling towards the right with identical speed $c = 1$ are presented. The ICs employed here are $u(x, 0) = e^{-|x+2|} + e^{-|x-2|}$, which forms a bi-nominal shape. The training error is $4.27e-3$, with scaling factors $\lambda_{ic} = \lambda_{bc} = 100$. As shown in Figure 8a, the peakons maintain their distance during propagation. Moreover, it can be discerned from Figure 8b that SGNN is capable of making very good predictions of such configurations. Indeed, the mean-squared and maximum absolute errors are $2.99e-5$ and $5.22e-2$, respectively for this case. The good agreement between SGNN and exact solutions is further demonstrated in Figure 8c, where "x" makers are for exact solutions and solid lines are predictions by SGNN.

The complementary case corresponding to the interaction of two anti-peakons traveling at different speeds is presented in Figure 9. In particular, we consider a configuration involving two anti-peakons: one centered at $x = -5$ with speed 0.8, and another one whose center is (symmetrically) placed at $x = 5$ and travels with velocity of -2.2. The respective IC that describes this configuration is $u(x, 0) = -0.8e^{-|x+5|} - 2.2e^{-|x-5|}$. The training error is 0.0188, with scaling factors $\lambda_{ic} = \lambda_{bc} = 100$. On the validation dataset, the mean-squared error is $2.99e-5$. In addition, the maximum absolute error is $5.22e-2$, which is reflected in Figure 9b. The interactions of these two anti-peakons are shown in Figure 9a. The second anti-peakon (in darker red), possessing a higher velocity, will eventually overtake the first one (in orange), despite initially lagging behind. A good agreement between SGNN prediction and exact solution is demonstrated in Figure 9c. The second one catches the first one at $t = 7.14$, where their peaks add up, as shown in Figure 9d.

Finally, Figure 10 shows a more realistic scenario: the (elastic) collision between a peakon and an anti-peakon. In this case, the IC considered is given by $u(x, 0) = -e^{-|x-2|} + e^{-|x+2|}$, where the training error is $6.12e-3$, with scaling factors $\lambda_{ic} = \lambda_{bc} = 100$. The mean-squared validation error is $2.11e-5$ while the maximum absolute validation error is

$5.54e-2$, as illustrated in Figure 10b. As expected, the peakon (light red) and anti-peakon (in darker red) move towards each other with same velocity as shown in Figure 10a until they collide at $t = 2$. Indeed, Figure 10d showcases the predicted solution at the time of collision where the waveforms cancel each other. Then, at later times, i.e., $t > 2$, the anti-peakon and peakon re-emerge, and they can maintain their shape after collision, as shown in Figure 10c.

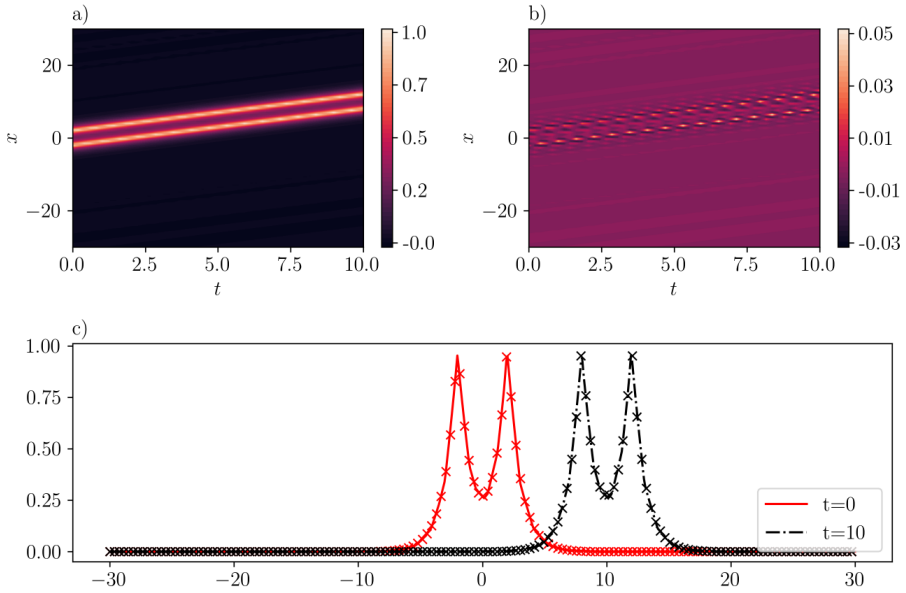

Figure 8. Two peakons with identical traveling speed ($c = 1$) in the CH equation. (**a**): $\bar{u}(x,t)$ inferred by SGNN; (**b**): error $e(x,t) = u(x,t) - \bar{u}(x,t)$; (**c**): $\bar{u}(x,t)$ at two time instants. In (**c**), "x" represents the exact solution while lines represent the prediction by SGNN. The training loss is $4.27e-3$, with $\lambda_{ic} = \lambda_{bc} = 100$. Validation error: $\|e\|_\infty = 5.22e-2$, $\|e\|_2 = 2.99e-5$.

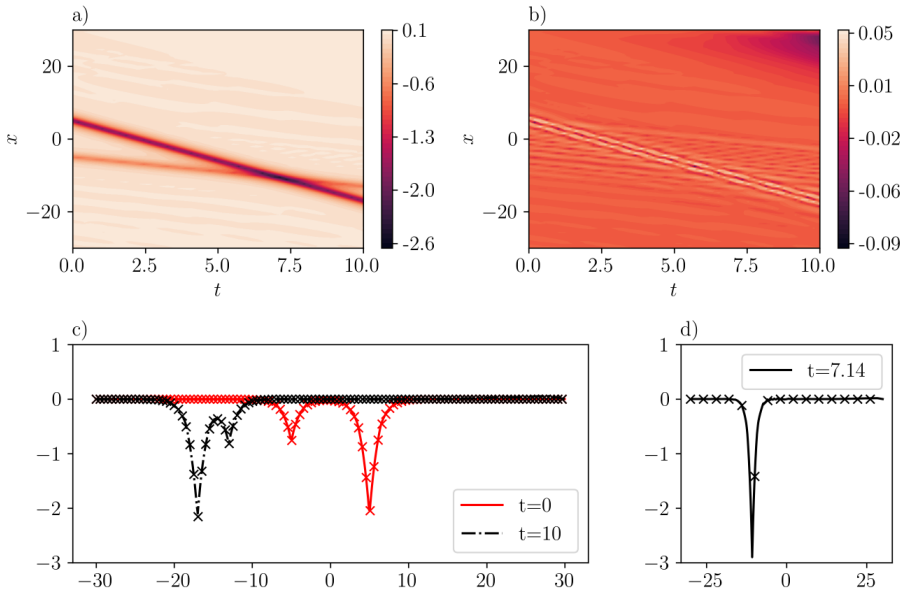

Figure 9. Same as Figure 8 but for the case corresponding to the interaction of two anti-peakons ($c_1 = -0.8$, $c_2 = -2.2$) in the CH equation. (**a**): $\bar{u}(x,t)$ inferred by SGNN; (**b**): error $e(x,t) = u(x,t) - \bar{u}(x,t)$; (**c**): $\bar{u}(x,t)$ at two time instants; (**d**): $\bar{u}(x,t)$ when two peakons collide. The format of panel (**c**) is the same as the one in Figure 8. The training loss is $1.88e-2$, with $\lambda_{ic} = \lambda_{bc} = 100$. Validation error: $\|e\|_\infty = 9.0e-2$, $\|e\|_2 = 9.12e-5$.

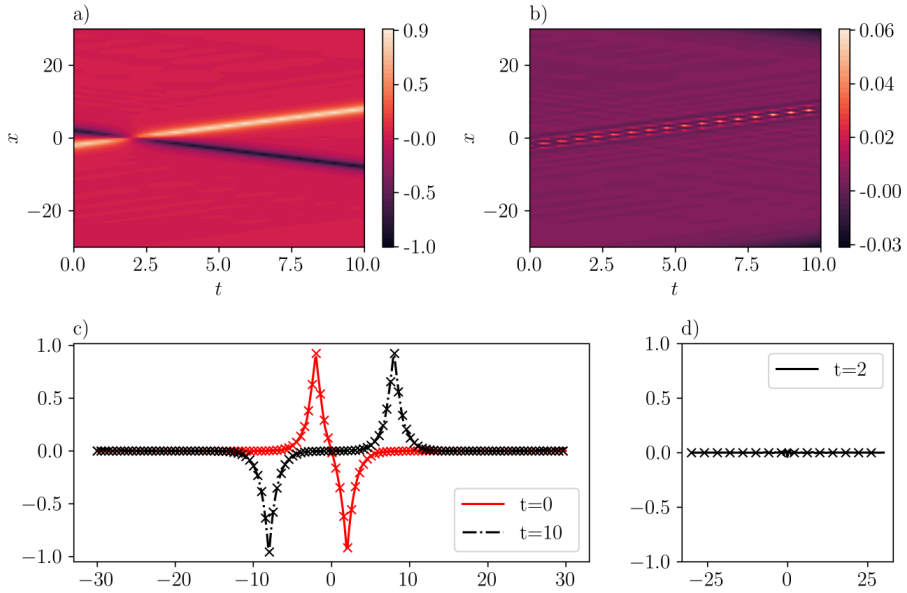

Figure 10. Same as Figure 9 but for the case corresponding to the "head-on" collision of a peakon and an anti-peakon (with $c_1 = c_2 = -1$) in the CH equation. (**a**): $\bar{u}(x,t)$ inferred by SGNN; (**b**): error $e(x,t) = u(x,t) - \bar{u}(x,t)$; (**c**): $\bar{u}(x,t)$ at two time instants; (**d**): $\bar{u}(x,t)$ when two peakons collide. The format of panel (**c**) is the same as the one in Figure 8. The training loss is $6.12e-3$, with $\lambda_{ic} = \lambda_{bc} = 100$. Validation error: $\|e\|_\infty = 5.54e-2$, $\|e\|_2 = 2.11e-5$.

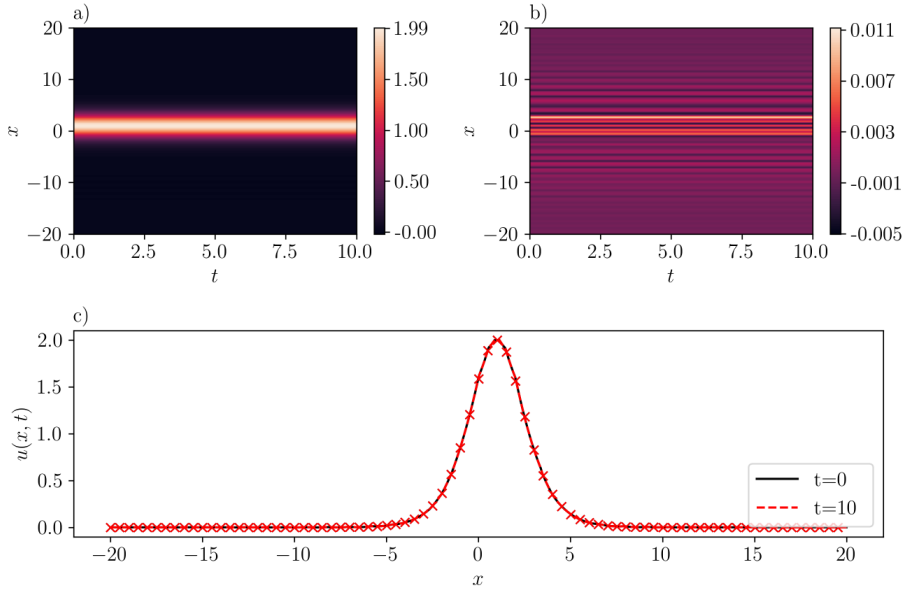

Figure 11. Same as Figure 10 but for a stationary solution, i.e., "lefton" of the b-family with $b = -2.0$. (**a**): $\bar{u}(x,t)$ inferred by SGNN; (**b**) $e(x,t) = u(x,t) - \bar{u}(x,t)$; (**c**) $u(x,t)$ at two time instants. The format of panel (**c**) is the same as the one in Figure 9. The training loss is 0.054, with $\lambda_{ic} = 1{,}000$, $\lambda_{bc} = 1$. Validation loss: $\|e\|_\infty = 1.12e-2$, $\|e\|_2 = 4.62e-6$.

3.1.4. Lefton Solutions

The last case that we consider using SGNNs is the lefton regime, i.e., $b < -1$ whose explicit solution form is given by Equation (17). Herein, we study such solutions at $b = -2$. For our training dataset, we randomly select $2^{12} = 4096$ points within the domain alongside an additional $2^9 = 512$ points, subsequently dividing this dataset into 8 mini-batches. The chosen time domain is set at $t \in [-10, 10]$, and the spatial domain at $x \in [-20, 20]$. As

depicted in Figure 11, there is a high degree of concordance between the SGNN predictions and the exact solutions. The training loss, adjusted by scaling factors $\lambda_{ic} = 1000$ and $\lambda_{bc} = 1$, is recorded at 0.054. The mean-squared error for the validation loss stands at $4.62e-6$, with the maximum absolute validation loss reaching $1.12e-2$.

4. Peakons in *ab*-Family

In this section, we turn our focus on the applicability of SGNN to the so-called *ab*-family [20]

$$u_t + u^2 u_x - a u_x^3 + D^{-2}\partial_x \left[\frac{b}{3}u^3 + \frac{6-6a-b}{2} u u_x^2\right] + D^{-2}\left[\frac{2a+b-2}{2} u_x^3\right] = 0 \quad (18)$$

of peakon equations where D^{-2} stands for the nonlocal operator $(1-\partial_x^2)^{-1}$. The *ab*-family is a generalization of the *b*-family [cf. Equation (14)] in the sense that it corresponds to cubic (in its nonlinearity) CH-type equations unlike the quadratic CH-type equations of the *b*-family [20]. Interestingly, the *ab*-family admits the one-peakon solution taking the form

$$u(x,t) = \pm\sqrt{c}e^{-|x-(1-a)ct|}. \quad (19)$$

For the applicability of SGNN, we inspect the peakon solution in the spatial domain $x \in [-20, 20]$ and time domain $t \in [0, 10]$. A SGNN with 80 neurons is used to approximate the one-peakon solution in the *ab*-family. To generate the training dataset, we randomly generate $2^{13} = 8192$ collection samples within the domain and $2^{10} = 1024$ samples on the boundary. The training dataset is evenly split into 8 mini-batches. In the loss function, $\lambda_{ic} = 1000$ and $\lambda_{bc} = 100$ are applied to penalize ICs and BCs. Same as before, the ADAM method is then used to train the SGNN, followed by the refinement by L-BFGS.

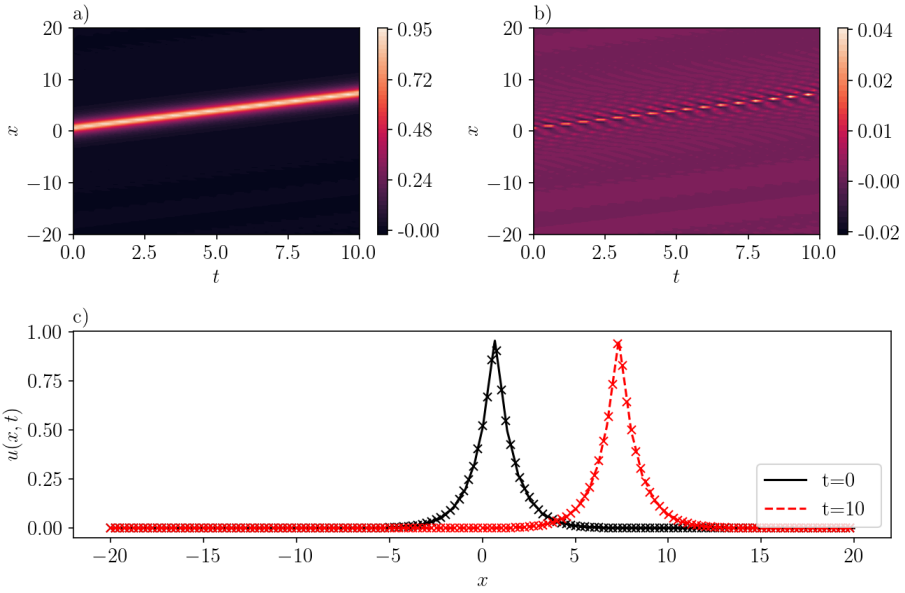

Figure 12. A peakon in the *ab*-family with $b = 2.0, a = 1/3$. The wave speed is $c = 1$. (**a**): $\bar{u}(x,t)$ inferred by SGNN; (**b**) $e(x,t) = u(x,t) - \bar{u}(x,t)$; (**c**) $u(x,t)$ at two time instants. In (**c**), "x" markers represent the exact solution while lines depict the prediction by SGNN. The training loss is 0.0141, with $\lambda_{ic} = 1000$, $\lambda_{bc} = 100$. Validation loss: $\|e\|_\infty = 3.70e-2$, $\|e\|_2 = 8.63e-6$.

Distinct from the members of the *b*-family, both peakons and anti-peakons of the *ab*-family propagate in the same direction. This behavior is confirmed using parameters $b = 2.0$, $a = 1/3$, and $c = 1$, as illustrated in Figures 12 and 13. The training losses for the peakon and anti-peakon solutions are recorded at 0.0141 and 0.0138, respectively. For the peakon solution, the mean-squared error across the validation set is measured at $8.63e-6$,

with the maximum absolute error reaching $3.7e-2$. Similarly, the anti-peakon solution exhibits a mean-squared error of $8.24e-6$ over the validation set, and its maximum absolute error is noted as $4.5e-2$.

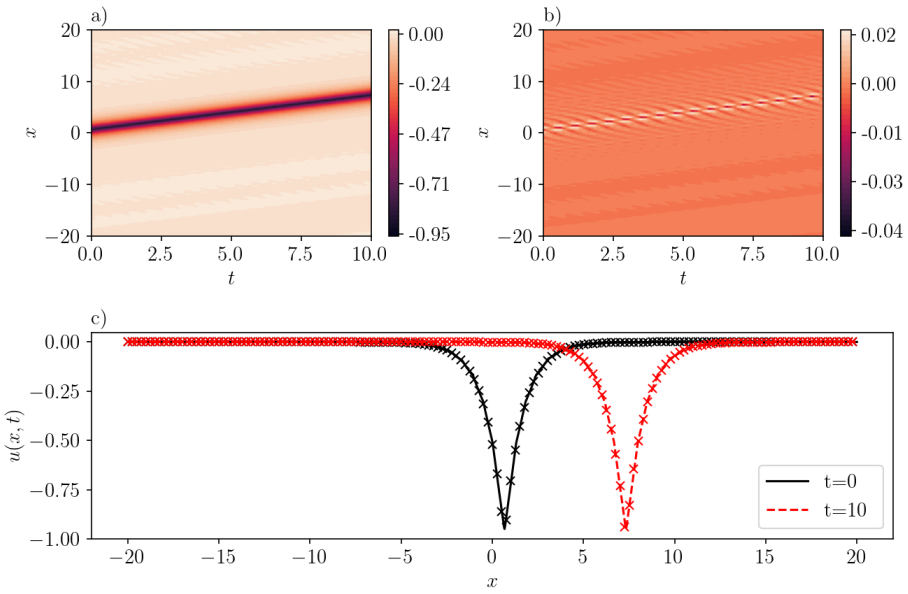

Figure 13. Same as in Figure 12, but for an anti-peakon in the ab-family with $b = 2.0, a = 1/3$, and wave speed $c = -1$. (**a**): $\bar{u}(x,t)$ inferred by SGNN; (**b**) $e(x,t) = u(x,t) - \bar{u}(x,t)$; (**c**) $u(x,t)$ at two time instants. The format of panel (**c**) is the same as the one in Figure 12. Here, the training loss is 0.0138, with $\lambda_{ic} = 1000$, $\lambda_{bc} = 100$. Validation error: $\|e\|_\infty = 4.05e-2$, $\|e\|_2 = 8.24e-6$.

5. 2D Compactons

In this section, we depart from the previous one-dimensional (in space) studies, and apply SGNN in order to predict TWs in two-dimensional nonlinear wave equations. More specifically, we focus on TWs that have compact support which are referred to as compactons, and introduced in [24]. Following the notation in [24], there exists a family of PDEs denoted as $C_N(m, a+b)$ given by

$$u_t + (u^m)_x + \frac{1}{b}[u^a(\nabla^2 u^b)]_x = 0, \tag{20}$$

that possesses compacton TWs with $m \geq max(1, a-1), b > 0$. Here, $C_N(m, a+b)$ represents a N-dimensional compacton (with $N = 1, 2, 3$) with a parameter set $\{m, a, b\}$. In the following, we restrict ourselves to $N = 2$, and concentrate on the single compacton case. In other words, the network structure of Figure 2 will be used. According to [23], Equation (20) supports traveling compactons traversing in the x direction. In this case, we have

$$s = x - \lambda t, \tag{21}$$

where λ is the velocity of the compacton. The case with $C_2(m = 1+b, 1+b)$ yields an explicit solution in the form

$$u = \lambda^{1/b}\left[1 - \frac{F(R)}{F(R^*)}\right]^{1/b}, \; 0 < R \leq R^*, \tag{22}$$

where u vanishes elsewhere (i.e., compact support). In Equation (22), $R = \sqrt{s^2 + y^2}$, and $F(R) = J_0(\sqrt{b}R)$ where J_0 is the zeroth-order Bessel function, and $\sqrt{b}R^*$ is the root of the first-order Bessel function.

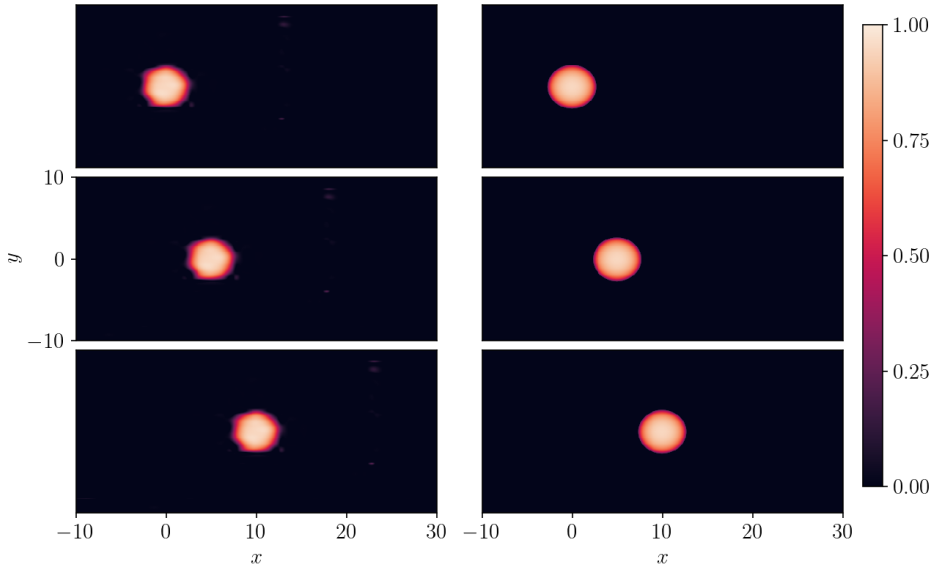

Figure 14. A 2D compacton $C_2(3, 1+2)$ of Equation (22) with $\lambda = 1$. Left panel: SGNN prediction; right panel: ground truth. Top panel: $t = 0$; middle panel: $t = 5$; bottom panel: $t = 10$. The training loss:$9.97e-3$:$\lambda_{ic} = 100$, $\lambda_{bc} = 10$. Validation error: $\|e\|_\infty = 0.371$, $\|e\|_2 = 1.58e-4$.

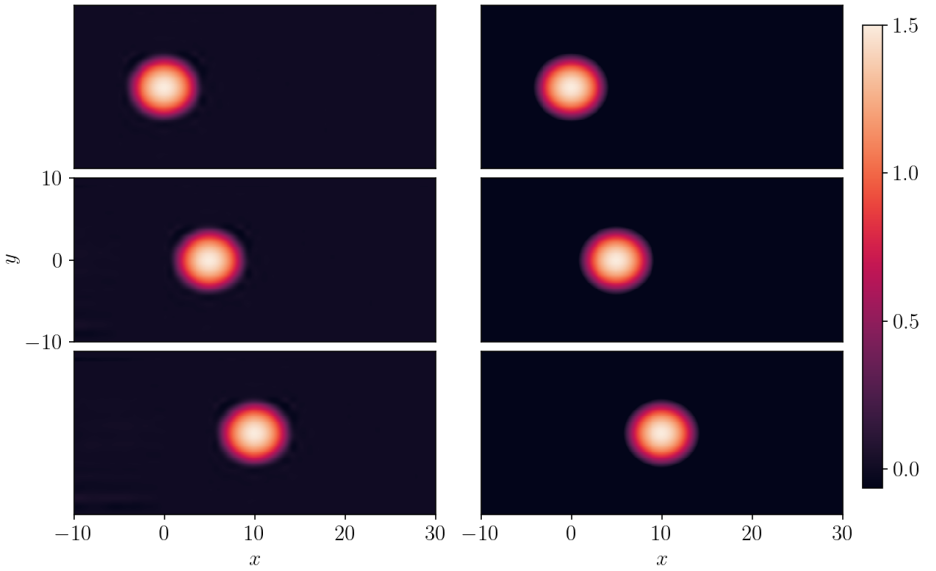

Figure 15. Same as Figure 14 but for the 2D compacton $C_2(2, 0+3)$ of Equation (24) with $\lambda = 1$. Left panel: SGNN prediction; right panel: ground truth. Top panel: $t = 0$; middle panel $t = 5$, bottom panel: $t = 10$. The training loss:$1.23e-3$:$\lambda_{ic} = 100$, $\lambda_{bc} = 10$. Validation error: $\|e\|_\infty = 0.135$, $\|e\|_2 = 8.28e-5$.

We use a SGNN to approximate the compacton $C_2(3, 1+2)$. The SGNN has two layers, with 50 neurons per layer, and the approximation is performed in the spatial domain $x \in [-10, 30]$ and time domain $t \in [0, 10]$. To generate the training dataset, we randomly sampled $2^{16} = 65,536$ collocation points within the domain, along with $2^{12} = 4096$ points on the boundary. The dataset is then evenly split into 8 mini-batches. The mini-batch ADAM is used to SGNN, with loss functions to minimize the residual error of the PDE, ICs, and BCs.

In Figure 14, the SGNN's prediction (left column) is presented against the exact (right column) compacton solution $C_2(3, 1+2)$. The training loss is stopped at $9.97e-3$, with $\lambda_{ic} = 100$, and $\lambda_{bc} = 10$. The mean-squared validation error is $1.58e-4$ while the maximum absolute error is 0.371. The compacton travels along x-axis with a velocity of $\lambda = 1$. At

$t = 0$ (top panel), the compacton commences with a center placed at $x = 0$. Middle and bottom panels present snapshots of compactons at $t = 5$, and $t = 10$, respectively. We report that the SGNN's prediction captures the main characteristics of the exact solution although minor errors appear around the edges of compacton.

As a last case, we consider the compacton $C_2(m = 2, a + b = 3)$ whose explicit solution is given by

$$u = \kappa_N[\lambda A_N - bR^2], 0 < R \leq R_* \equiv \sqrt{\lambda A_N / b} \tag{23}$$

with u vanishing elsewhere. According to Ref. [23], we pick $N = 2$, $m = 0$, $a = 0$, and $b = 3$, and thus we have

$$C_N(2, 0 + 3) : A_2 = \frac{3}{2}(4 + 2)^2, \kappa_2^{-1} = 6(4 + 2). \tag{24}$$

The SGNN's prediction and exact solution of $C_2(2, 0 + 3)$ compacton solutions are presented in the left and right columns of Figure 15. Snapshots of the solutions are shown at $t = 0$ (top), $t = 5$ (middle), and $t = 10$ (bottom) therein. In this case, we report that the predictions precisely match the exact solution at these times. The scaling factors in the loss function are $\lambda_{ic} = 100$ and $\lambda_{bc} = 10$, and the training loss is stopped at $1.23e - 3$ after 300 epochs. The mean-squared validation error is $8.28e - 5$, while the maximum absolute error is 0.135.

In summary, we present the training losses and validation errors of all previous results (see, also the reference Figure in the left column) in Table 1.

Table 1. The training losses and validation errors of the presented results in Sections 2–5.

Figure	Training loss	Validation error	
		$\|e\|_2$	$\|e\|_\infty$
3	$8.43e - 3$	$7.21e - 6$	$3.90e - 2$
4	$1.94e - 11$	$9.59e - 13$	$1.02e - 5$
5	$9.18e - 2$	$4.09e - 6$	$2.74e - 2$
6	$3.10e - 2$	$3.50e - 5$	$5.92e - 3$
8	$4.27e - 3$	$2.99e - 5$	$5.22e - 2$
9	$1.88e - 2$	$9.12e - 5$	$9.00e - 2$
10	$6.12e - 3$	$2.11e - 5$	$5.54e - 2$
11	$5.40e - 2$	$4.62e - 6$	$1.12e - 2$
12	$1.40e - 2$	$8.63e - 6$	$3.70e - 2$
13	$1.38e - 2$	$8.24e - 6$	$4.05e - 2$
14	$9.97e - 3$	$1.58e - 4$	$3.71e - 1$
15	$1.23e - 3$	$8.28e - 5$	$1.35e - 1$

6. Comparison and Discussion

To compare the performance of the traditional and new structures of PINNs, we use them to approximate a peakon solution in the CH equation with $c = 1$. The spatial domain employed is $[-20, 20]$, while the temporal domain is $[0, 10]$. The size of the training set is $2^{12} = 4096$, with $2^9 = 512$ samples for the ICs and BCs. The selection of width and depth for models is informed by the configurations reported in existing literature. Additionally, we also compare the performance of SGNN vs MLP. As shown in Table 2, in the traditional PINN framework, neither SGNN nor MLP can successfully converge to a TW on a large spatial and temporal domain. Despite small training losses, all NN structures get stuck at the trivial solution as it is very difficult to overcome propagation failure when dealing with enlarged domains. By introducing a training method that respects causality [17] or performs adaptive sampling [16], one may be able to address this failure.

On the other hand, as illustrated in Table 3, with the TW coordinate transformation, identical NN structures of SGNN and MLP with ReLu function all converge to the correct solution. Although the training losses of MLP with hyperbolic tangent and sigmoid functions are relatively large, they all capture the characteristics of the TW. Sigmoid and hyperbolic functions have difficulties approximating the non-differentiable peak of the

peakon, with about 0.3 error in the peak value. Notably, these results can be improved by modifying the sampling method, training scheme, and loss functions. With the increase of depth and width, MLP with ReLU and sigmoid functions can further reduce loss values. The loss values with SGNN also gradually drop as width increases. SGNN excels at the compact structure that only requires less than 1/10 of training parameters. In addition, SGNN can give an explicit solution form of the PDEs in the sense of Gaussian radial-basis functions. However, increasing further the number of neurons in SGNN does not dramatically result in loss-value reduction. This could be remedied by modifying the training and sampling schemes.

Table 2. Comparison of SGNN and MLP with the traditional PINNs. Despite small loss values, no network structure can converge to the correct solution. Spatial domain: $[-30, 30]$, time domain: $[0, 10]$. GRB: generalized Gaussian radial-basis function.

Network	Activation	Depth	Width	Loss	Trivial solution?
SGNN	GRBF	1	40	$(2.69 \pm 5.47)e-7$	Yes
MLP	ReLu	2	40	$(1.13 \pm 0.60)e-3$	Yes
	ReLu	4	40	$(6.11 \pm 4.10)e-4$	Yes
	ReLu	6	40	$(1.05 \pm 0.66)e-3$	Yes
	ReLu	8	20	$(1.06 \pm 0.77)e-3$	Yes
	sigmoid	2	40	$(7.37 \pm 2.10)e-6$	Yes
	sigmoid	4	40	$(1.28 \pm 0.62)e-5$	Yes
	sigmoid	6	40	$(1.08 \pm 0.82)e-6$	Yes
	sigmoid	8	20	$(1.91 \pm 0.91)e-5$	Yes
	tanh	2	40	$(1.26 \pm 0.24)e-6$	Yes
	tanh	4	40	$(2.11 \pm 0.54)e-6$	Yes
	tanh	6	40	$(2.24 \pm 0.69)e-6$	Yes
	tanh	8	20	$(3.59 \pm 1.46)e-6$	Yes

Table 3. Comparison of SGNN and MLP with PINNs incorporating with traveling frame. Spatial domain: $[-30, 30]$, time domain: $[0, 10]$. GRB: generalized Gaussian radial-basis function.

Network	Activation	Depth	Width	Parameters	Loss	Trivial solution?
SGNN	GRBF	1	20	60	$(1.42 \pm 0.08)e-2$	No
	GRBF	1	40	120	$(1.02 \pm 0.17)e-2$	No
	GRBF	1	60	180	$(8.65 \pm 1.22)e-3$	No
MLP	ReLu	2	40	1640	$(2.95 \pm 0.12)e-2$	No
	ReLu	4	40	4840	$(2.93 \pm 0.11)e-2$	No
	ReLu	6	40	8040	$(5.34 \pm 2.10)e-4$	No
	ReLu	8	20	2820	$(8.60 \pm 2.93)e-4$	No
	sigmoid	2	40	1640	$0.72 \pm 5.62e-4$	No
	sigmoid	4	40	4840	$0.72 \pm 6.80e-3$	No
	sigmoid	6	40	8040	$0.71 \pm 7.61e-3$	No
	sigmoid	8	20	2820	$0.72 \pm 5.33e-3$	No
	tanh	2	40	1640	$0.72 \pm 7.00e-3$	No
	tanh	4	40	4840	$0.71 \pm 4.65e-3$	No
	tanh	6	40	8040	0.70 ± 0.05	No
	tanh	8	20	2820	0.56 ± 0.15	No

Why can the modified structure of PINNs avoid propagation failure and lead to better results? We attempt to answer this question next. By mathematically transforming the NN

input to the traveling coordinate $x - ct$, we inherently produce an output in the form of $u(x - ct)$. This representation naturally aligns with the solution form of TWs, maintaining the integrity of the solution's structure. From a physical perspective, this transformation converts a dynamic problem into a static one (i.e., a TW becomes stationary in a frame that co-moves with the solution), thus simplifying the problem considerably. Algorithmically, this transformation effectively reduces the input dimension by one, which can lead to a decrease in the required data size for training. Furthermore, the functional form of TWs of $u(x - ct)$ ensures that any combination of spatial and temporal coordinates resulting in the same traveling frame coordinate will produce identical outcomes. This characteristic automatically propagates initial and/or boundary conditions along the characteristic path $x - ct$, significantly reducing the challenge of extending solutions from ICs and BCs to interior points.

7. Conclusions

In this work, we introduce a modified structure of PINNs that incorporates the mathematical description of TWs to nonlinear PDEs. In particular, we integrated a novel neural network architecture, called SGNN, into the PINNs framework. Our approach demonstrated a significant improvement in overcoming the propagation failure of PINNs, particularly in large-domain applications. Utilizing this enhanced network, we successfully generated interpretable predictions for TWs across various PDE families including the b- and ab-families of peakon equations. To the authors' best knowledge, this is also the first study on applying PINNs to identify 2D TWs, such as compactons, as well as to study the collisions of 1D multiple peakons. This work opens up new directions for future studies that we plan to undertake herein. Specifically, and on the one hand, there exist solutions to nonlinear dispersive PDEs that self-similarly blow-up in finite (or infinite) time [33]. Under a stretching transformation [33], such solutions can appear as steady ones in a frame that "co-explodes" with the solution [34,35], thus enabling the applicability of the present NN architecture for the identification and prediction of self-similar collapse. On the other hand, the present NN structure could be expanded in order to model and predict the transient behavior of TWs. In addition, regularization techniques can be incorporated to refine the model to capture the essential features of the solutions more succinctly.

Author Contributions: Conceptualization, methodology, software, validation, writing—original draft preparation: S.X. Research design, writing—review and editing: E.G.C. All authors have read and agreed to the published version of the manuscript.

Funding: S. Xing is supported by the Donald E. Bently Center for Engineering Innovation and Lockheed Endowed Professorship in the College of Engineering at Cal Poly. This work has been supported by the U.S. National Science Foundation under Grant No. DMS-2204782 (EGC).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: We dedicate this work to the memory of our esteemed colleague and friend, Joseph Callenes-Sloan (Electrical and Computer Engineering Department, Cal Poly San Luis Obispo).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Raissi, M.; Perdikaris, P.; Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
2. Karniadakis, G.; Kevrekidis, I.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L. Physics-informed machine learning. *Nat. Rev. Phys.* **2021**, *3*, 422–440. [CrossRef]
3. Cai, S.; Wang, Z.; Wang, S.; Perdikaris, P.; Karniadakis, G. Physics-informed neural networks for heat transfer Problems. *J. Heat Transf.* **2021**, *143*, 060801. [CrossRef]
4. Jin, X.; Cai, S.; Li, H.; Karniadakis, G. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *J. Comput. Phys.* **2021**, *426*, 109951. [CrossRef]

5. Wang, Y.; Lai, C.Y.; Gómez-Serrano, J.; Buckmaster, T. Asymptotic self-similar blow-up profile for three-dimensional axisymmetric euler equations Using Neural Networks. *Phys. Rev. Lett.* **2023**, *130*, 244002. [CrossRef]
6. Zhu, W.; Khademi, W.; Charalampidis, E.G.; Kevrekidis, P.G. Neural networks enforcing physical symmetries in nonlinear dynamical lattices: The case example of the Ablowitz–Ladik model. *Phys. D* **2022**, *434*, 133264. [CrossRef]
7. Saqlain, S.; Zhu, W.; Charalampidis, E.G.; Kevrekidis, P.G. Discovering governing equations in discrete systems using PINNs. *Commun. Nonlinear Sci. Numer. Simul.* **2023**, *126*, 107498. [CrossRef]
8. Van Herten, R.L.; Chiribiri, A.; Breeuwer, M.; Veta, M.; Scannell, C.M. Physics-informed neural networks for myocardial perfusion mri quantification. *Med. Image Anal.* **2022**, *78*, 102399. [CrossRef]
9. Wang, S.; Teng, Y.; Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM J. Sci. Comput.* **2021**, *43*, A3055–A3081. [CrossRef]
10. Wang, S.; Yu, X.; Perdikaris, P. When and why PINNs fail to train: A neural tangent kernel perspective. *J. Comput. Phys.* **2022**, *449*, 110768. [CrossRef]
11. Krishnapriyan, A.S.; Gholami, A.; Zhe, S.; Kirby, R.; Mahoney, M.W. Characterizing possible failure modes in physics-informed neural networks. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 26548–26560.
12. Daw, A.; Bu, J.; Wang, S.; Perdikaris, P.; Karpatne, A. Mitigating Propagation Failures in Physics-informed Neural Networks using Retain-Resample-Release (R3) Sampling. In Proceedings of the ICML'23: Proceedings of the 40th International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023; pp. 7264–7302.
13. Camassa, R.; Holm, D.D. An integrable shallow water equation with peaked solitons. *Phys. Rev. Lett.* **1993**, *71*, 1661–1664. [CrossRef]
14. Wang, L.; Yan, Z. Data-driven peakon and periodic peakon solutions and parameter discovery of some nonlinear dispersive equations via deep learning. *Physica D* **2021**, *428*, 133037. [CrossRef]
15. Tancik, M.; Srinivasan, P.P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J.T.; Ng, R. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7537–7547.
16. Wu, C.; Zhu, M.; Tan, Q.; Kartha, Y.; Lu, L. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Comput. Methods Appl. Mech. Eng.* **2023**, *403*, 115671. [CrossRef]
17. Wang, S.; Sankaran, S.; Perdikaris, P. Respecting causality is all you need for training physics-informed neural networks. *Comput. Methods Appl. Mech. Eng.* **2024**, *421*, 116813. [CrossRef]
18. Braga-Neto, U. Characteristics-informed neural networks for forward and inverse hyperbolic problems. *arXiv* **2022**, arXiv:2212.14012.
19. Holm, D.D.; Staley, M.F. Nonlinear balance and exchange of stability in dynamics of solitons, peakons, ramps/cliffs and leftons in a 1+1 nonlinear evolutionary PDE. *Phys. Lett. A* **2003**, *308*, 437–444. [CrossRef]
20. Himonas, A.A.; Mantzavinos, D. An ab-family of the equation with peakon traveling waves. *Proc. Am. Math. Soc.* **2016**, *144*, 3797–3811. [CrossRef]
21. Degasperis, A.; Holm, D.D.; Hone, A.N.W. A new integrable equation with peakon solutions. *Theor. Math. Phys.* **2002**, *133*, 1463–1474. [CrossRef]
22. Fuchssteiner, B.; Fokas, A. Symplectic structures, their Bäcklund transformations and hereditary symmetries. *Phys. D Nonlinear Phenom.* **1981**, *4*, 47–66. [CrossRef]
23. Rosenau, P.; Hyman, J.M.; Staley, M. Multidimensional Compactons. *Phys. Rev. Lett.* **2007**, *98*, 024101. [CrossRef] [PubMed]
24. Rosenau, P. Compact and noncompact dispersive patterns. *Phys. Lett. Sect. A Gen. At. Solid State Phys.* **2000**, *275*, 193–203. [CrossRef]
25. Xing, S.; Sun, J.Q. Separable Gaussian Neural Networks: Structure, analysis, and function approximations. *Algorithms* **2023**, *16*, 453. [CrossRef]
26. Park, J.; Sandberg, I.W. Universal approximation using Radial-Basis-Function Networks. *Neural Comput.* **1991**, *3*, 246–257. [CrossRef] [PubMed]
27. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the ICLR (Poster), San Diego, CA, USA, 7–9 May 2015.
28. Liu, D.C.; Nocedal, J. On the limited memory BFGS method for large scale optimization. *Math. Program.* **1989**, *45*, 503–528. [CrossRef]
29. Kodama, Y. Normal forms for weakly dispersive wave equations. *Phys. Lett. A* **1985**, *112*, 193–196. [CrossRef]
30. Kodama, Y. On integrable systems with higher order corrections. *Phys. Lett. A* **1985**, *107*, 245–249. [CrossRef]
31. Camassa, R.; Holm, D.D.; Hyman, J.M. *A New Integrable Shallow Water Equation*; Volume 31: Advances in Applied Mechanics; Elsevier: Amsterdam, The Netherlands, 1994; pp. 1–33.
32. Charalampidis, E.; Parker, R.; Kevrekidis, P.; Lafortune, S. The stability of the b-family of peakon equations. *Nonlinearity* **2023**, *36*, 1192–1217. [CrossRef]
33. Sulem, C.; Sulem, P. *The Nonlinear Schrödinger Equation*; Springer: New York, NY, USA, 1999.

34. Chapman, S.; Kavousanakis, M.; Charalampidis, E.; Kevrekidis, I.; Kevrekidis, P. A spectral analysis of the nonlinear Schrödinger equation in the co-exploding frame. *Phys. D Nonlinear Phenom.* **2022**, *439*, 133396. [CrossRef]
35. Chapman, S.; Kavousanakis, M.; Charalampidis, E.; Kevrekidis, I.; Kevrekidis, P. Self-similar blow-up solutions in the generalized Korteweg-de Vries equation: Spectral analysis, normal form and asymptotics. *arXiv* **2023**, arXiv:2310.13770 .