entropy

Article

Learning the Nonlinear Solitary Wave Solution of the Korteweg–De Vries Equation with Novel Neural Network Algorithm

Ying Wen [1,*,†] and Temuer Chaolu [2,†]

1 College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China
2 College of Sciences and Arts, Shanghai Maritime University, Shanghai 201306, China; tmchaolu@shmtu.edu.cn
* Correspondence: 201840310002@stu.shmtu.edu.cn
† These authors contributed equally to this work.

Abstract: The study of wave-like propagation of information in nonlinear and dispersive media is a complex phenomenon. In this paper, we provide a new approach to studying this phenomenon, paying special attention to the nonlinear solitary wave problem of the Korteweg–De Vries (KdV) equation. Our proposed algorithm is based on the traveling wave transformation of the KdV equation, which reduces the dimensionality of the system, enabling us to obtain a highly accurate solution with fewer data. The proposed algorithm uses a Lie-group-based neural network trained via the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization method. Our experimental results demonstrate that the proposed Lie-group-based neural network algorithm can simulate the behavior of the KdV equation with high accuracy while using fewer data. The effectiveness of our method is proved by examples.

Keywords: KdV equation; Lie groups; deep learning; BFGS; partial differential equations

Citation: Wen, Y.; Chaolu, T. Learning the Nonlinear Solitary Wave Solution of the Korteweg–De Vries Equation with Novel Neural Network algorithm. *Entropy* **2023**, *25*, 704. https://doi.org/10.3390/e25050704

Academic Editor: Nikolay Kolev Vitanov

Received: 17 March 2023
Revised: 20 April 2023
Accepted: 22 April 2023
Published: 24 April 2023

1. Introduction

Nonlinear science is an interdisciplinary research field that spans a wide range of scientific domains, including life science, mathematical science, spatial science, and geographic science. In recent decades, the study of solitons has become increasingly important and widespread. Solitons are relevant in many fields of science, including fluid mechanics, quantum mechanics, biology, ocean engineering, and more. Therefore, the solution of soliton equations is theoretically and practically critical and has become an important area of theoretical and application-based research. The investigation of soliton equation solution techniques has not halted since the soliton theory was first put forward. Numerous equations have verified several mature solution techniques, including the Painlevé analysis [1], the Backlund transform method [2], the Darboux transform method [3], the inverse scattering transform method [4], the Lie group and Lie algebra method [5], the Hamiltonian structure method [6], etc. There are numerous approaches for finding the exact solution to the soliton problem due to its complexity; however, these approaches cannot be unified.

The ability to answer such scientific computing issues in conjunction with numerical analysis has significantly improved with the development of machine learning [7]. The most well-known area of machine learning research is deep learning. Researchers originally proved a general approximation theorem [8] for neural networks, and deep learning was later used in a variety of fields, including image recognition [9], natural language processing [10], and optimization problems [11]. The neural networks have an advantage over the dimensional catastrophe problem that traditional numerical approaches must deal with because of the increased dimensionality and the linear rise in computational effort. Differential equations and machine learning methods combined have been proven to be advantageous. The combination of differential equations and machine learning methods is beneficial in solving soliton problems. The performance of neural network-based algorithms in predicting soliton solutions has been investigated, and these algorithms have demonstrated promising results in terms of accuracy and the amount of required data. With ongoing developments in machine learning, there is potential for further advancements in the field of soliton equation solutions. Overall, the integration of machine learning and soliton theory is a promising approach that can lead to significant progress in various fields of science.

Recent advancements in deep neural networks have enabled researchers to solve differential and partial differential equations (PDEs) with fewer data points while maintaining high accuracy, making deep learning techniques an increasingly popular alternative to traditional numerical methods. In the beginning, Lee et al. [12] employed Hopfield neural networks to solve ordinary differential equation (ODE) models. Lagaris et al. [13] obtained the trial solution within the error range by continuously optimizing the parameters of the neural network and replacing the solution of the equation with the sum of the initial and boundary value and the neural network function. Methods that require fewer data and produce quick results while maintaining high accuracy are gaining popularity. Chen et al. [14] used neural networks to parameterize the derivatives of the hidden states rather than directly parameterizing the hidden states, auto memory tuning, and adaptive computation to compress the ODEs into the neural network. Raissi et al. [15,16] suggested the novel loss function form to physical priors into the neural network architecture. Differential equations and machine learning techniques are combined in the scientific machine learning that Rackauckas et al. [17] proposed. Continuous convolutional neural networks were employed by Habiba et al. [18] to learn PDE systems. Deep learning in machine learning is increasingly being used as a framework to analyze PDEs, including the common nonlinear wave model KdV equation. Cellular neural networks can imitate KdV behavior, as demonstrated experimentally by Bilotta et al. [19]. To forecast the solutions and parameters of the KdV equation, Fang et al. [20] integrated conservation laws into neural networks. Higher-order nonlinear soliton equations were solved by Cui et al. [21] using deep learning techniques. A two-stage physics-informed neural network (PINN) approach was suggested by Lin et al. [22] to more accurately and generally simulate the local wave solutions of the productive equations. Lin et al. [23] followed up by using the Miura transform and PINN to propose a PINN scheme based on Miura transform to solve the KdV equation. Wu et al. [24] conducted a comprehensive study on the sampling method of PINN sampling and tested its performance in the KdV equation, guiding researchers on the sampling method in subsequent research. Applications of deep learning in analyzing PDEs, including the common nonlinear wave model KdV equation, are gaining popularity. The literature [25] provides a review of the broad application potential of deep neural networks for solving PDEs. This approach is expected to be more widely adopted in future research, facilitating the progress of scientific areas such as physics, biology, and finance.

While neural networks have been used to solve PDE problems, the development of efficient algorithms that utilize minimal resources whilst still effectively addressing the underlying properties of solutions remains an ongoing issue. In this paper, we propose a novel approach that utilizes a Lie-group-based neural network algorithm for solving the KdV equation. Our new method boasts good learning performance, which we affirm by comparing its numerical results with those obtained from the true solution. Specifically, inspired by the unique form of Lie group theory for solving first-order differential equations, we developed a novel method to address PDE problems by constructing a solution consisting of a neural network function and a Lie-group-based solution. In our approach, the sum of these two parts approximates the solution of the differential equation. To effectively apply this approach to PDE problems, we first convert the PDEs into an ODEs. We observed that constructing the solution in this manner eliminates the need to increase the initial value item in the loss function whilst still fully satisfying the initial value. Moreover, using only a small number of neural network parameters can improve fitting ability, all thanks to the Lie-group-based solution. Our proposed approach is highly efficient since the Lie-group-based solution captures the nonlinear characteristics of the KdV equation well before training the neural network. As a result, the cost of the subsequent neural network calculations is reduced. Our approach not only delivers precise predictions but also highlights essential characteristics of the KdV solution such as the constancy of solitary waves over time. By leveraging our method, we can better understand and analyze complex physical phenomena described by the KdV equation, a feat that has remained challenging using other techniques. The ability to capture these key features is crucial in advancing our understanding of nonlinear dynamics and provides a significant boost to the predictive power of our model.

Encouragingly, our investigation revealed that this new method can efficiently and accurately capture complex phenomena in nonlinear waves. We developed all implementations using PyCharm 2021.2.3 and conducted simulations on a Lenovo laptop with a 2.60 GHz 2-core Intel(R) Core(TM) i5-3230M CPU and $8GB$ memory. The proposed approach could serve as a foundation for future research exploring more general forms of PDEs while reflecting upon the properties of the solutions. The code used in this study is made publicly available to support reproducibility and ease further analyses.

The remainder of this essay is structured as follows. The algorithm presented in this paper and its precise steps are shown in Section 2. The approach is used to solve the KdV equation in Section 3, and this section goes into great depth about how it was accomplished and how accurate the results were. We also study and evaluate our results. Concluding comments and future research work are offered in Section 4.

2. The Main Idea of the Lie-Group-Based Neural Network Algorithm

2.1. Illustration of the Algorithms

Consider the general form of PDE as follows:

$$u_t + N(x, u, u_x, u_{xx}, \cdots) = 0, \quad x \in \Omega, \quad t \in [0, T]. \tag{1}$$

The independent variables x, t, the solution u to be solved, and the partial derivatives of u with respect to the space variable x make up the nonlinear function N. The equation is subject to boundary or initial conditions.

The following autonomous system of ODEs is obtained by transforming [26] the PDE:

$$\frac{du_i}{da} = f_i(u_1, u_2, \cdots, u_n), \ u_i(0) = \alpha_i \in \mathbb{R}^1, i = 1, 2, \cdots, n. \tag{2}$$

where $a \in \mathcal{O} \subset \mathbb{R}^1$ is independent variable x or t and $u_i = u_i(a)$ is u in (1) and f_i are differential functions of own arguments after the variable t or x has been eliminated. α_i is the initial condition.

From [27], the solution of (2) can be written as Lie group solution $\hat{u}(a) = e^{aD}\alpha$, where D is the differential operator. According to theorem 2, D can be split into $D_1 + D_2$, $e^{aD} = e^{aD_1} + \sum_{\alpha=1}^{\infty}\sum_{k=\alpha}^{\infty} \frac{a^k}{k!} D_1^{k-\alpha} D_2 D^{\alpha-1}$. The first part e^{aD_1} is obtained from the equation $\frac{d\bar{u}}{da} = D_1\bar{u}, \bar{u}(0) = \alpha$, the second integral calculation should be replaced with the neural network function form $a\mathcal{N}(a;\theta)$, which offers superior simplicity and ease of computation. The advantage of $\hat{u}(a) = \bar{u}(a) + a\mathcal{N}(a;\theta)$ for approximating the solution u of the (2) is that the first part can easily capture the nonlinear nature of the equation which can accelerate the convergence of the second part of the neural computation, while the second part uses a simple neural network structure with fewer resources and less memory consumption, and the sum of the two parts can effectively model the behavior of (1).

In our study, we use a fully connected neural network $\mathcal{N}$ with one input, one output, m units in the hidden layer, and an activation function σ. The outputs of the network $\mathcal{N}$ can be written as

$$\begin{aligned}\mathcal{N}(a;\theta) &= Z_L \\ &= \sigma(W^{(L)} \cdots \sigma(W^{(l)} \cdots \sigma(W^{(2)} \cdot \sigma(W^{(1)} \cdot a + b^{(1)}) + b^{(2)})) + b^{(L)}).\end{aligned} \tag{3}$$

The output of the layer l is $Z_l = \sigma(W^{(l)} \cdot Z_{l-1} + b^{(l)})$, $\{a_\tau\}_{\tau=1}^{\lambda}$ is the training point in the definition domain, $\theta = \{W^{(l)}, b^{(l)}\}_{l=1}^{L}$ is the parameters of the neural network, where $W^{(l)}$ is the weight of layer l with respect to layer $l-1$, b^l is the bias of layer l, and $w_{jk}^{(l)}$ is the weight from units k in the layer $l-1$ to units j in the layer l. By adjusting the parameter θ, we can enhance the approximation of $u(a)$ via the network solution $\hat{u}(a)$ where σ is a nonlinear activation function $\tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

$$W^{(l)} = \begin{pmatrix} w_{11}^l & \cdots & w_{1m_{k-1}}^l \\ \vdots & \ddots & \vdots \\ w_{m_k 1}^l & \cdots & w_{m_k m_{k-1}}^l \end{pmatrix}, \tag{4}$$

$$b^{(l)} = \begin{pmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_{m_k}^l \end{pmatrix}. \tag{5}$$

2.2. Details of the Algorithm

The unconstrained optimization process of (2) is measured by the following mean square error equation

$$\mathcal{L}(\theta) = \frac{1}{\lambda}\left(\sum_{\tau=1}^{\lambda}\sum_{i=1}^{n}\left(\hat{u}_i'(a_\tau) - f_i\right)^2\right), \tag{6}$$

The trial solution $\hat{u}(a)$ is substituted into (2) so that the loss function (6) of the neural network is minimized at the training points, and the parameter set $\{W, b\}$ is found using the optimization algorithm. λ is the number of training points and n denotes the number of equations. When the number of equations increases, the number of training points can be increased. The method can successfully approximate the solution u of (1) when $\mathcal{L}(\theta)$ is small enough. In addition to using the mean square error mentioned above to create the loss function, we also used the average root mean square error L_{RMSE} to evaluate the superiority of the method.

$$L_{RMSE} = \frac{1}{2}\sum_{\mu} L_\mu(\theta), \tag{7}$$

where $L_\mu(\theta)$ is the mean square error between the trial solution $\hat{u}_i(a)$ and the exact solution $u_i(a)$. When the exact solution is not available, the numerical solution $u_i(a)$ is employed, μ is the number of dependent variables, where $L_1(\theta) = \sqrt{\frac{1}{\lambda}\left(\sum_{\tau=1}^{\lambda}(\hat{u}(a_\tau,\theta) - u(a_\tau))^2\right)}$.

3. Example for Korteweg–De Vries Equation

In this study, we present our novel method for identifying solitons of the KdV equation. The KdV equation represents a fundamental model in mathematical physics and is typically formulated as a PDE given by $u_t + 6uu_x + u_{xxx} = 0$. This equation is commonly used to describe water waves and has been extensively studied in the previous literature [28].

To detect the soliton of the equation, we first perform a traveling transformation denoted by $\xi = x - vt$, thereby enabling us to transform the PDE into an ODE. Specifically, this transformation allows us to rewrite the equation in terms of the new variable ξ as

$$u''' + 6uu' - vu' = 0, \tag{8}$$

with $u = u(\xi)$. We seek the soliton to the (8) with properties $u(0) = u_{max}, u'(0) = 0, u''(0) = u_0 < 0$ and $u(\pm\infty) = 0$. Specifically, when $\xi = 0$, the wave value reaches its peak at $u(0) = 1$. In our particular case, we take $v = 2$, $u_{\max} = 1$, $u_0 = -1$ and consider the variable ξ over the interval $[-3, 3]$.

This ODE formulation can be solved using our proposed method, which efficiently detects soliton solutions in the equation. We transform the problem (8) to the standard form in our method as

$$\dot{u}_1 = u_2,\ \dot{u}_2 = u_3, \dot{u}_3 = 2u_2 - 6u_1u_2, \tag{9}$$

with initial values $u_1(0) = 1, u_2(0) = 0, u_3(0) = -1$ by introducing variables $(u_1, u_2, u_3) = (u, \dot{u}, \ddot{u})$. It corresponds to operator $D = D_1 + D_2$ with a selection $D_1 = u_2\partial_{u_1} + u_3\partial_{u_2} + 2u_2\partial_{u_3}$.

The associated initial value problem yields solutions

$$\begin{aligned} &\bar{u}_1(\xi) = -\frac{1}{2}\Big(\cosh(\sqrt{2}\xi) - 3\Big); \bar{u}_2(\xi) = -\frac{1}{\sqrt{2}}\sinh(\sqrt{2}\xi); \\ &\bar{u}_3(\xi) = -\cosh(\sqrt{2}\xi). \end{aligned}$$

Therefore, we have trial solution $\hat{u} = \bar{u} + \xi\mathcal{N}(\xi, \theta)$. The parameters of the neural network $\mathcal{N}$ can be learned by minimizing the mean squared error loss (6)

$$\mathcal{L}(\xi, \theta) = \frac{1}{\lambda}\left(\sum_{\tau=1}^{\lambda}\big(\hat{u}_1'(\xi_\tau) - f_1\big)^2 + \big(\hat{u}_2'(\xi_\tau) - f_2\big)^2 + \big(\hat{u}_3'(\xi_\tau) - f_3\big)^2\right), \tag{10}$$

where $f_1 = \hat{u}_2(\xi_\tau), f_2 = \hat{u}_3(\xi_\tau), f_3 = 2\hat{u}_2(\xi_\tau) - 6\hat{u}_1(\xi_\tau)\hat{u}_2(\xi_\tau), \tau = 1, 2, \cdots, \lambda$.

The comparisons of our solution $\bar{u}_1$ and exact solution $u = \operatorname{sech}^2\Big(\frac{\xi}{\sqrt{2}}\Big)$ to (8) are given in the Figure 1. Our proposed method has been able to accurately approach the true solution in the range of interest $[-1, 1]$, indicating the first part of our solution is effective. This not only improves the accuracy of our solution but also speeds up the computation of the second part of our neural network, which ensures rapid convergence.

In this study, we investigate the ability of our proposed method to learn from sparse training data. To accomplish this, we obtained a limited set of 250 training data within the interval $[-3, 3]$, which were used to train a feed-forward neural network with a single hidden layer consisting of 30 units. The network solution $\hat{u}$ for the initial value problem (8) was obtained by minimizing the mean square error of (6) using Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [29]. The comparison of the network solution $\hat{u}$ with the exact solution u on the training set is presented in the left panel of Figure 2. Our method achieves an accurate approximation of the exact solution u even when trained with small amounts of data. The right panel of Figure 2 shows the comparison of the network solution $\hat{u}$ with the exact solution u within the test set $[-3, 3.3]$. We observe that the network maintains good generalization ability outside of the interval $[-3, 3]$ and continues to provide accurate approximations of the solution in the absence of training points. Our results suggest that the proposed approach is capable of learning from limited training data, which is particularly important in practical scenarios where data acquisition may be difficult or expensive.

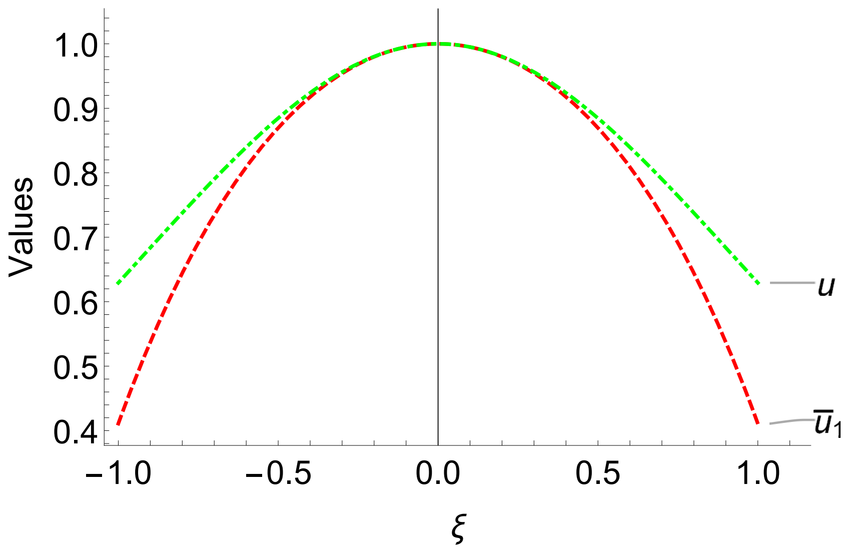

Figure 1. Comparison of the efficiency of the first part $\bar{u}_1$ of the network solution $\hat{u}_1$ with the exact solution $u = \text{sech}^2\left(\xi/\sqrt{2}\right)$ of problem (8).

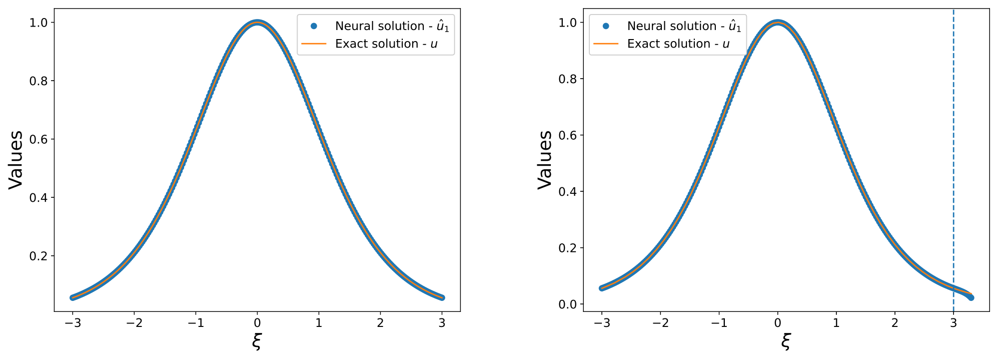

Figure 2. (**Left**): Comparison of our solution $\hat{u}$ with the exact solution $u = \text{sech}^2\left(\xi/\sqrt{2}\right)$ of problem (8) in the training set. (**Right**): Comparison of our solution $\hat{u}$ with the exact solution $u = \text{sech}^2\left(\xi/\sqrt{2}\right)$ of problem (8) in the test set.

The blue line in Figure 3 shows the relationship between the number of iterations during training and the loss function $\mathcal{L}(\xi, \theta)$, an error entirely attributed to the Lie group method and the neural network's ability to approximate u. When the number of iterations is around 2000, $\mathcal{L}(\theta) = 2.8378 \times 10^{-7}$. $L_{RMSE} = 7.6167 \times 10^{-5}$. To evaluate the effectiveness of our proposed method, we compare it with existing PINN approaches. We conduct experiments using 250 training points in the interval $x \in [-1,3], t \in [0,1]$ and a single hidden layer structure with 30 neurons. The loss function is set to $L = \frac{1}{\lambda}\sum_{\tau=1}^{\lambda}\left(L_f + L_u\right)$, where $L_f = (\hat{u}_t + 6\hat{u}\hat{u}_x + \hat{u}_{xxx})^2$, $L_u = (\hat{u}(0, x^\tau) - u(0, x^\tau))^2 + (\hat{u}(t^\tau, -1) - u(t^\tau, -1))^2 + (\hat{u}(t^\tau, 3) - u(t^\tau, 3))^2$. The objective is to approximate the exact solution u with higher accuracy under the same conditions as our proposed method. We compare the performance of our method with other PINN algorithms by the number of iterations versus the loss value. Our experimental results show that the PINN method produces better loss function values until the number of iterations is 100, and our proposed method outperforms existing methods in terms of convergence speed and accuracy after 100 iterations. As described in Figure 3, the loss function L of PINN method reaches 10^{-3} after about 2000 iterations.

We investigate the prediction accuracy of several neural network architectures using the same training points to analyze the performance of our proposed method in more detail. We study the loss function $\mathcal{L}(\theta)$ for different numbers of hidden layers and different numbers of neurons per layer. Table 1 presents the results of our analysis, demonstrating the impact of varying the architecture of the neural network on prediction accuracy. Here, the training points are fixed to the range $[-3, 3]$ of 250 uniformly spaced points. As expected, we observe that as the number of layers and neurons increases, the prediction accuracy systematically improves. This is in line with the general notion that larger and deeper neural networks have greater expressive power and are better equipped to approximate complex functions. It is worth noting that while increasing the number of neurons and layers in a neural network can improve its performance, it also comes at a cost of increased computational complexity and potentially slower training times. Therefore, in practical settings, it is important to balance the trade-off between model complexity and computational efficiency. Our findings suggest that, given sufficient training data, our proposed method can be used to build highly accurate models, but careful consideration must be given to the size of the neural network when implementing it in practical applications.

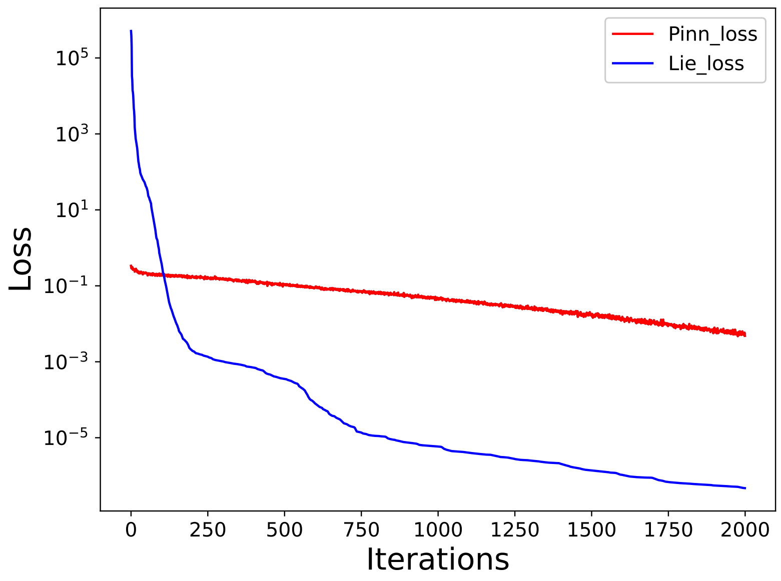

Figure 3. The variation curve of the loss function with the number of iterations for the PINN approach and the Lie-group-based neural network algorithm for problem (8).

Table 1. The loss function $\mathcal{L}(\theta)$ for different number of hidden layers and different number of neurons per layer. Here, the training points are fixed to the range $[-3, 3]$ of 250 uniformly spaced points.

Neurons $\mathcal{L}(\theta)$ **Layers**	**30**	**40**	**50**
1	2.3×10^{-7}	1.9×10^{-7}	2.0×10^{-7}
2	3.1×10^{-8}	3.5×10^{-8}	2.7×10^{-8}
3	1.6×10^{-8}	1.8×10^{-8}	1.5×10^{-8}

The results of this experiment are summarized in Figure 4. Specifically, the top left panel shows the true solution $u(t, x)$ of the KdV equation, while the right panel displays the spatiotemporal solution $\hat{u}(t, x)$ predicted according to the chosen optimal parameter θ. We observe that our approach is highly accurate in approximating the true solution. The bottom panel of Figure 4 gives a more detailed evaluation of the predicted solution $\hat{u}(t, x)$. For different times $t = 0.3$, 0.5, and 0.8, we compare the exact and predicted solutions in particular at the bottom of Figure 4. Our experimental results demonstrate that our approach can produce highly accurate predictions even for complex spatiotemporal problems such as the KdV equation.

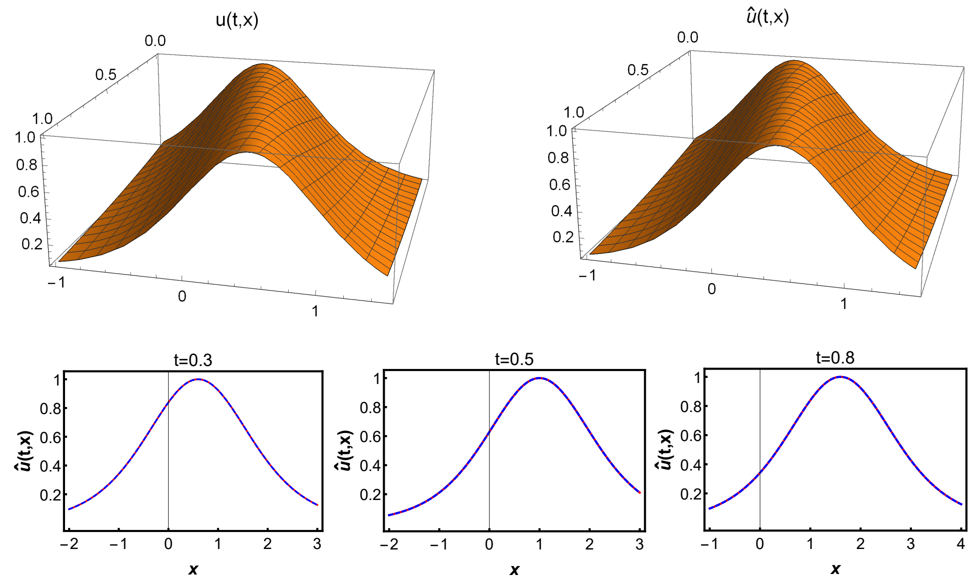

Figure 4. (**Top**): The true solution $u = \text{sech}^2\left((x-2t)/\sqrt{2}\right)$ of the KdV equation is on the left, the predicted solution $\hat{u}(t,x)$ is on the right. (**Bottom**): Comparison of predicted and exact solutions at time $t = 0.3, 0.5$, and 0.8. (The dashed blue line indicates the exact solution $u(t,x)$, and the solid red line indicates the predicted solution $\hat{u}(t,x)$).

To further investigate the effectiveness of the algorithm in approximating the performance of the true solution of the KdV equation, with the true solution $u = 2/3 - 2\tanh^2(\xi)$ [30] , the solitary wave with wave peak $u(\xi) = 2/3$ is sought under the traveling wave transform $\xi = x + 4t$ with the initial conditions $v = -4, u_{max} = 2/3, u_0 = -4$ and consider the interval $[-3,3]$ for variable ξ.

The initial value problem $u_1(0) = 2/3, u_2(0) = 0, u_3(0) = -4$ for problem (9). Choose the operator D_1 that $u_2\partial_{u_1} + u_3\partial_{u_2} - 4u_2\partial_{u_3}$. The associated initial value problem yields solutions, $\bar{u}_1(\xi) = -\frac{1}{3}\cos(2\xi)$, $\bar{u}_2(\xi) = -2\sin(2\xi)$ and $\bar{u}_3(\xi) = -4\cos(2\xi)$. The decomposition part of the trial solution constructed from the Lie group can capture the nonlinear nature of the problem (8), as illustrated by a comparison between u and $\bar{u}_1$ in Figure 5.

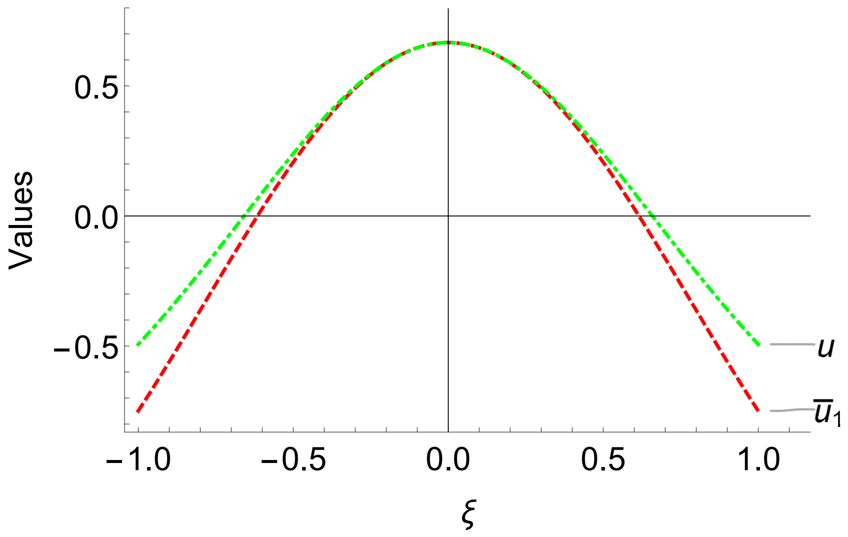

Figure 5. Comparison of the efficiency of the first part $\bar{u}_1$ of the network solution $\hat{u}_1$ with the exact solution $u = 2/3 - 2\tanh^2(\xi)$ of problem (8).

Using the BFGS algorithm, the same network structure with a single hidden layer containing 30 neurons, training data equally spaced at 250 training points in the range

$[-3, 3]$, and the same test data are learned u. The results are presented in Figure 6, where it is evident that our approach yields remarkable accuracy in predicting u.

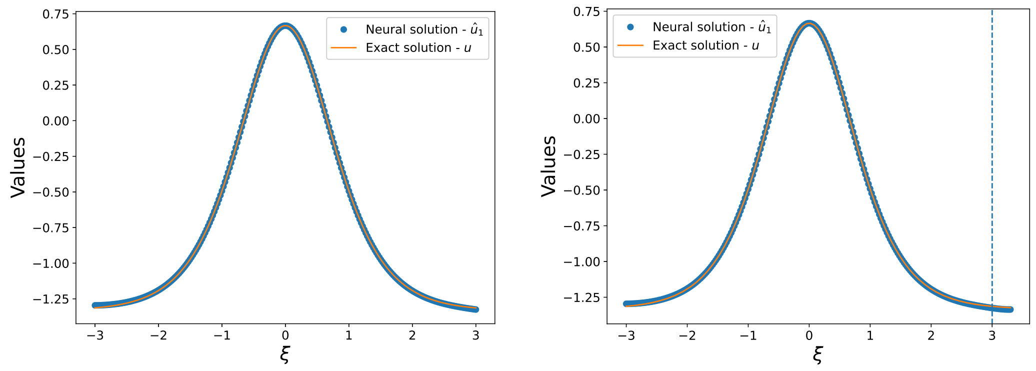

Figure 6. (**Left**): Comparison of our solution $\hat{u}$ with the exact solution $u = 2/3 - 2\tanh^2(\xi)$ of problem (8) in the training set. (**Right**): Comparison of our solution $\hat{u}$ with the exact solution $u = 2/3 - 2\tanh^2(\xi)$ of problem (8) in the test set.

The number of iterations and the loss function $\mathcal{L}(\xi, \theta)$ are shown in Figure 7, where we observe that in this case, $\mathcal{L}(\theta) = 5.4765 \times 10^{-7}$, indicating the high accuracy of our approach.

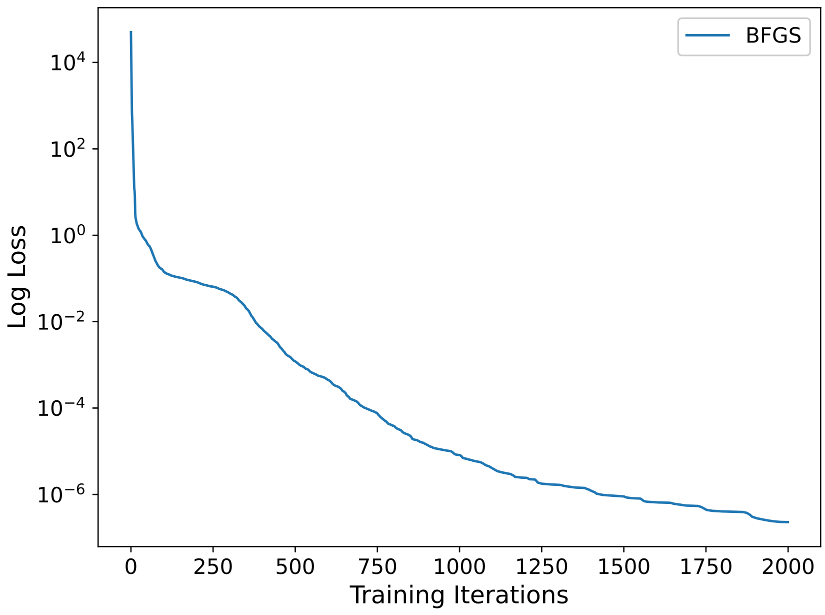

Figure 7. The variation curve of the Loss function with the number of iterations for the Lie-group-based neural network algorithm.

In Figure 8, we present a comparison between the true solution $u = 2/3 - 2\tanh^2(x + 4t)$ of the KdV equation (top left panel) and the predicted solution $\hat{u}$ (top right panel). Interestingly, the waveform of the single soliton does not change with time, as shown in the bottom panel of Figure 8 which gives the exact and predicted solutions for different times $t = 0.3, 0.5$, and 0.8.

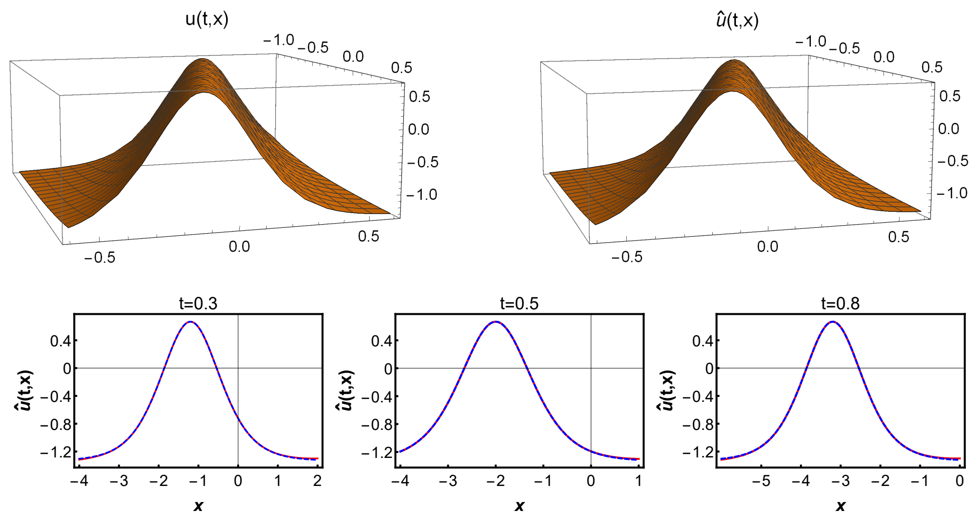

Figure 8. (**Top**): The true solution $u = 2/3 - 2\tanh^2(x + 4t)$ of the KdV equation is on the left, the predicted solution $\hat{u}(t, x)$ is on the right. (**Bottom**): Comparison of predicted and exact solutions at time $t = 0.3, 0.5$, and 0.8. (The dashed blue line indicates the exact solution $u(t, x)$, and the solid red line indicates the predicted solution $\hat{u}(t, x)$).

The complex nonlinear behavior of the KdV equation can be precisely captured by the Lie-groups-based neural network algorithm using just a minimal quantity of initial data (30 neurons in a single hidden layer with 250 training points).

4. Discussion and Conclusions

Our study focuses on the restoration of the dynamic behavior of the KdV equation using a Lie-group-based neural network algorithm. Compare with the existing PINN learning method, experimental findings demonstrate that our proposed method can accurately restore the dynamic behavior of the KdV equation with high accuracy and fast convergence under a small number of parameters and a simple network structure. In addition, a deep study is done for our proposed algorithm, and the accuracy is improved when the number of hidden layers increases with the number of neurons contained, but the time cost spent is also relatively high.

To confirm the accuracy and reliability of our proposed algorithm, we searched for other solitary solutions of the KdV equation. In the study presented in [31], an evaluation related to the design and efficacy of automatic tools for the derivation of solitary solutions of nonlinear differential equations is discussed. The study confirms by proof that the technique fails when considering the space of system parameters and initial conditions. To overcome these challenges, we can learn existing learning methods, such as the PINN method, to add both the errors generated by the initial and boundary conditions into the loss function. The change in our proposed algorithm in the way the loss function is constructed is made $L = L_I + L_F$, where L_I is the error generated by the network solution $\hat{u}$ in the initial or boundary term. This modification to the construction of the loss function ensures that errors arising from both the initial and boundary conditions are considered in the prediction process, leading to more accurate results overall. Furthermore, the choice of operator D_1 plays a crucial role in subsequent neural network computation, and selecting the appropriate operator is vital to ensure precise and reliable results.

Notably, the success of our approach depends on capturing the mathematical substance of the equation solutions, which is often overlooked in machine learning techniques used for numerical solutions of differential equations. Recent research has shown that more implicit information about the solutions could be ignored when using these approaches. However, our proposed algorithm overcomes this limitation with only a shallow neural network model and limited data. Through our validation process, we have shown that our proposed algorithm performs effectively, producing highly accurate predictions for solitary solutions of the KdV equation. Our approach offers a new avenue for accurately predicting complex nonlinear solutions of PDEs and lays the foundation for future studies into other similar problems,which motivates us to study models in other interdisciplinary fields, such as finance or medical biology. The future work requires more research on optimization techniques to improve performance in addition to addressing parameter constraints or initial value constraints encountered in appeal problems.

Author Contributions: Conceptualization, Y.W. and T.C.; methodology, Y.W.; software, Y.W.; validation, Y.W.; writing—original draft preparation, Y.W.; writing—review and editing, Y.W. and T.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China, grant number 11571008.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are included within the article. The link to the code is https://github.com/yingWWen/Learning_the_nonlinear_solitary_wave_solution_of_KdV_equation_with_novel_neural_network_algorithm, accessed on 16 March 2023.

Acknowledgments: The authors thank the support of the National Natural Science Foundation of China with grant number 11571008.

Conflicts of Interest: As far as we know, there are no conflicts of interest or financial or other conflicts. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Khater, A.; Callebaut, D.; Shamardan, A.; Ibrahim, R. Bäcklund transformations and Painlevé: Exact soliton solutions for strongly rarefied relativistic cold plasma. *Phys. Plasmas* **1997**, *4*, 3910–3922. [CrossRef]
2. Nimmo, J.; Freeman, N. The use of Backlund transformations in obtaining N-soliton solutions in Wronskian form. *J. Phys. A Math. Gen.* **1984**, *17*, 1415. [CrossRef]
3. Huang, D.; Li, D.; Zhang, H. Explicit N-fold Darboux transformation and multi-soliton solutions for the (1+1)-dimensional higher-order Broer–Kaup system. *Chaos Solitons Fractals* **2007**, *33*, 1677–1685. [CrossRef]
4. Arkadiev, V.; Pogrebkov, A.; Polivanov, M. Inverse scattering transform method and soliton solutions for Davey-Stewartson II equation. *Phys. D Nonlinear Phenom.* **1989**, *36*, 189–197. [CrossRef]
5. Kumar, S.; Kumar, D. Solitary wave solutions of (3+1)-dimensional extended Zakharov–Kuznetsov equation by Lie symmetry approach. *Comput. Math. Appl.* **2019**, *77*, 2096–2113. [CrossRef]
6. Ferapontov, E.; Galvao, C.; Mokhov, O.; Nutku, Y. Bi-Hamiltonian Structure in 2-d Field Theory. *Commun. Math. Phys.* **1997**, *186*, 649–669. [CrossRef]
7. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. Ser. B (Methodol.)* **1977**, *39*, 1–22.
8. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
9. Komar, M.; Yakobchuk, P.; Golovko, V.; Dorosh, V.; Sachenko, A. Deep neural network for image recognition based on the Caffe framework. In Proceedings of the 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 21–25 August 2018; pp. 102–106.
10. Li, H. Deep learning for natural language processing: Advantages and challenges. *Natl. Sci. Rev.* **2018**, *5*, 24–26. [CrossRef]
11. Louati, H.; Bechikh, S.; Louati, A.; Hung, C.C.; Said, L.B. Deep convolutional neural network architecture design as a bi-level optimization problem. *Neurocomputing* **2021**, *439*, 44–62. [CrossRef]
12. Lee, H.; Kang, I.S. Neural algorithm for solving differential equations. *J. Comput. Phys.* **1990**, *91*, 110–131. [CrossRef]
13. Lagaris, I.E.; Likas, A.; Fotiadis, D.I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **1998**, *9*, 987–1000. [CrossRef] [PubMed]
14. Chen, R.T.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D.K. Neural ordinary differential equations. In Proceedings of the Advances in Neural Information Processing Systems 31 (NeurIPS 2018), Montreal, QC, Canada, 3–8 December 2018; Volume 31.
15. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv* **2017**, arXiv:1711.10561.
16. Raissi, M. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *J. Mach. Learn. Res.* **2018**, *19*, 932–955.

17. Rackauckas, C.; Ma, Y.; Martensen, J.; Warner, C.; Zubov, K.; Supekar, R.; Skinner, D.; Ramadhan, A.; Edelman, A. Universal differential equations for scientific machine learning. *arXiv* **2020**, arXiv:2001.04385.
18. Habiba, M.; Pearlmutter, B.A. Continuous Convolutional Neural Networks: Coupled Neural PDE and ODE. In Proceedings of the 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET), Cape Town, South Africa, 9–10 December 2021; pp. 1–4.
19. Bilotta, E.; Pantano, P. Cellular nonlinear networks meet KdV equation: A new paradigm. *Int. J. Bifurc. Chaos* **2013**, *23*, 1330003. [CrossRef]
20. Fang, Y.; Wu, G.Z.; Kudryashov, N.A.; Wang, Y.Y.; Dai, C.Q. Data-driven soliton solutions and model parameters of nonlinear wave models via the conservation-law constrained neural network method. *Chaos Solitons Fractals* **2022**, *158*, 112118. [CrossRef]
21. Cui, S.; Wang, Z.; Han, J.; Cui, X.; Meng, Q. A deep learning method for solving high-order nonlinear soliton equations. *Commun. Theor. Phys.* **2022**, *74*, 075007. [CrossRef]
22. Lin, S.; Chen, Y. A two-stage physics-informed neural network method based on conserved quantities and applications in localized wave solutions. *J. Comput. Phys.* **2022**, *457*, 111053. [CrossRef]
23. Lin, S.; Chen, Y. Physics-informed neural network methods based on Miura transformations and discovery of new localized wave solutions. *Phys. D Nonlinear Phenom.* **2023**, *445*, 133629. [CrossRef]
24. 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]
25. Blechschmidt, J.; Ernst, O.G. Three ways to solve partial differential equations with neural networks– review. *GAMM-Mitteilungen* **2021**, *44*, e202100006. [CrossRef]
26. Vitanov, N.K. Modified method of simplest equation: Powerful tool for obtaining exact and approximate traveling-wave solutions of nonlinear PDEs. *Commun. Nonlinear Sci. Numer. Simul.* **2011**, *16*, 1176–1185. [CrossRef]
27. Wen, Y.; Chaolu, T.; Wang, X. Solving the initial value problem of ordinary differential equations by Lie group based neural network method. *PLoS ONE* **2022**, *17*, e0265992. [CrossRef]
28. Bluman, G.W.; Cheviakov, A.F.; Anco, S.C.; Bluman, G.W.; Cheviakov, A.F.; Anco, S.C. Construction of Mappings Relating Differential Equations. In *Applications of Symmetry Methods to Partial Differential Equations*; Springer: New York, NY, USA, 2010; pp. 121–186.
29. Dennis, J.E., Jr.; Schnabel, R.B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*; SIAM: Philadelphia, PA, USA, 1996.
30. Griffiths, G.; Schiesser, W.E. *Traveling Wave Analysis of Partial Differential Equations: Numerical and Analytical Methods with MATLAB and Maple*; Academic Press: Cambridge, MA, USA, 2010.
31. Navickas, Z.; Ragulskis, M. Comments on "A new algorithm for automatic computation of solitary wave solutions to nonlinear partial differential equations based on the Exp-function method". *Appl. Math. Comput.* **2014**, *243*, 419–425. [CrossRef]