Int. J. Appl. Comput. Math (2022) 8:87
https://doi.org/10.1007/s40819-022-01288-3

ORIGINAL PAPER

Solving Fredholm Integral Equations Using Deep Learning

Yu Guan[1] · Tingting Fang[1] · Diankun Zhang[1] · Congming Jin[1]

Accepted: 5 March 2022 / Published online: 29 March 2022

Abstract
The aim of this paper is to provide a deep learning based method that can solve high-dimensional Fredholm integral equations. A deep residual neural network is constructed at a fixed number of collocation points selected randomly in the integration domain. The loss function of the deep residual neural network is defined as a linear least-square problem using the integral equation at the collocation points in the training set. The training iteration is done for the same set of parameters for different training sets. The numerical experiments show that the deep learning method is efficient with a moderate generalization error at all points. And the computational cost does not suffer from "curse of dimensionality" problem.

Keywords Fredholm integral equation · High-dimensional problem · Residual neural network · Deep learning

Introduction

Integral equations have wide applications in electrical engineering [1], optics [2], mathematical biology [3] and other fields. The most popular integral equations are the Fredhom integral equations and the Volterra integral equations. The Fredholm integral equation can be considered as a reformulation of the elliptic partial differential equation and the Volterra integral equation is a reformulation of the fractional-order differential equation, which has wide applications in modeling the real problems, for instance, the chaotic system [4], the dynamics of COVID-19 [5], the motion of beam on nanowire [6], the capacitor microphone dynamical system [7], etc. Since these integral equations usually can not be solved explicitly, numerical methods are necessary to be considered.

We consider the linear Fredholm integral equation of the second kind

$$f(\mathbf{x}) - \int_{\Omega} k(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y} = g(\mathbf{x}), \tag{1}$$

where $\mathbf{x}, \mathbf{y} \in \Omega \subset \mathbb{R}^m$, the function $g(\mathbf{x})$ and the kernel $k(\mathbf{x}, \mathbf{y})$ are given, and $f(\mathbf{x})$ is the unknown that we want to find.

✉ Congming Jin
jincm@zstu.edu.cn

1 Department of Mathematics, Zhejiang Sci-Tech University, Hangzhou 310018, China

So far, many numerical methods have been proposed to solve the Fredholm integral equations, for example, the Nyström method [3, 8, 9], the Galerkin method [10], the wavelet analysis method [11], the neural network [12, 13], the collocation method [14], the maximum entropy method [15], etc. However, most of these traditional methods can only solve low-dimensional Fredholm integral equations and suffer from "curse of dimensionality".

The neural network has been successful in solving partial differential equations in mathematical modelling and the applied science, such as medical smoking model [16], nonlinear high order singular models [17], food chain system [18–20], Liénard differential model [21], etc. The neural network was also used to solve the Fredholm integral equations in [12, 13], where the authors only evaluated the approximation at some fixed points without generalization. And the integral was evaluated using numerical integral method whose cost depends on the dimension exponentially.

In recent years, deep learning method has been successfully used in artificial intelligence solving high-dimensional problems, such as image recognition [22, 23], speech recognition [24, 25], natural language processing [26], and also in mathematical problems [27–29] and physical problems [30].

E and his collaborators have done a series of works on solving high-dimensional differential equations based on deep learning method. In [28], a deep learning-based algorithm was proposed for solving high-dimensional semilinear parabolic partial differential equations and reverse stochastic differential equations from a relation between BSDE (backward stochastic differential equations) and reinforcement learning. In [29], the deep Ritz method for elliptic differential equations was given by numerically solving variational problems. In [27], a machine learning approximation algorithm was raised to solve high-dimensional fully nonlinear second-order partial differential equations. These works show that deep learning method provides a new idea to solve high-dimensional mathematical problems.

In this paper, a deep residual neural network method is proposed to approximate the solution of the high-dimensional linear Fredholm integral equations of the second kind. Few novel highlights of this deep learning method are briefly provided as follows:

- A deep residual neural network is constructed to solve numerically the linear Fredholm integral equations of the second kind.
- The proposed method can solve high-dimensional Fredholm integral equations and does not suffer from "curse of dimensionality" problem, that is the cost depends on the dimension linearly.
- The reasonable absolute error values validate the reliability of the deep learning method.
- The proposed method has a small generalization error in the domain.

This paper is organized as follows. In Sect. 2 we construct a deep residual neutral network for solving the Fredholm integral equations. In Sect. 3, some numerical experiments are given to show the efficiency of the numerical method. The conclusion is given in Sect. 4.

Proposed Deep Learning Method for Solving Fredhom Integral Equations

The output $F(\mathbf{x}, \boldsymbol{\theta})$ of the neural network is a composite function of the input $\mathbf{x}$, where $\boldsymbol{\theta}$ denotes the parameters of the neural network including the weighs and bias. Let $\mathbf{x}$ be any point in the domain Ω. Now we want to train a deep neural network whose output $F(\mathbf{x}, \boldsymbol{\theta})$ is

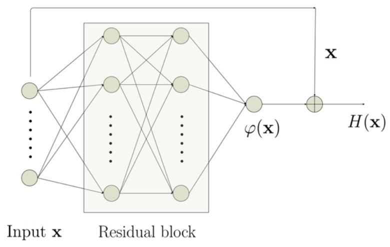

Fig. 1 Residual neural network block

the solution of the Fredholm integral Eq. (1), that is

$$F(\mathbf{x},\boldsymbol{\theta}) - \int_{\Omega} k(\mathbf{x},\mathbf{y})F(\mathbf{y},\boldsymbol{\theta})d\mathbf{y} = g(\mathbf{x}), \mathbf{x} \in \Omega.$$

To learn the parameters $\boldsymbol{\theta}$, and so the function $F(\mathbf{x},\boldsymbol{\theta})$, take randomly n points $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ with a uniform distribution as the training set. Initializing the parameter vector $\boldsymbol{\theta}$, the prediction values $F(\mathbf{x}_i,\boldsymbol{\theta})$ for $i = 1, 2, \cdots, n$, can be obtained by forward propagation neural network.

Define the loss function as

$$loss_{\mathbf{x}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(F(\mathbf{x}_i,\boldsymbol{\theta}) - \int_{\Omega} k(\mathbf{x}_i,\mathbf{y})F(\mathbf{y},\boldsymbol{\theta})d\mathbf{y} - g(\mathbf{x}_i)\right)^2. \tag{2}$$

The training of the neural network is to minimize the loss function (2) by the backward propagation neural network, which is a least-square problem

$$\min_{\boldsymbol{\theta}} loss_{\mathbf{x}}(\boldsymbol{\theta}). \tag{3}$$

In Eq. (2) the integral term $\int_{\Omega} k(\mathbf{x}_i,\mathbf{y})F(\mathbf{y},\boldsymbol{\theta})d\mathbf{y}$ can be evaluated using the Monte Carlo method, leading to

$$\int_{\Omega} k(\mathbf{x}_i,\mathbf{y})F(\mathbf{y},\boldsymbol{\theta})d\mathbf{y} \approx \frac{\beta}{n}\sum_{j=1}^{n} k(\mathbf{x}_i,\mathbf{x}_j)F(\mathbf{x}_j,\boldsymbol{\theta}), \tag{4}$$

where $\beta = \int_{\Omega} d\mathbf{x}$ is the volume of Ω.

The training can be done repeatedly for different training set until we get a stationary loss function.

As the network deepens, minimizing the loss function has great difficulties, such as vanishing gradient problem, gradient explosion, and degradation problem. The residual neural network can avoid the vanishing gradient problem and may greatly improve the solution. It also can reduce the risk of over-adapting the parameters to a specific dataset [22]. A residual block is shown in Fig. 1, where an identity shortcut connection is added to a shallow neural network, whose output is $H(\mathbf{x}) = \varphi(\mathbf{x}) + \mathbf{x}$, where $\varphi(\mathbf{x})$ is the output of the shallow neural network. Then the output of the residual block is taken as the input of the next residual block.

Our algorithm of deep residual neural network for solving Fredholm integral equations is shown in algorithm 1.

Algorithm 1 Framework of deep residual neural network.

```
Input: The number of training points $n$ and the number of training iterations $M$;
Output: The parameters $\boldsymbol{\theta}$ of the residual neural network;
1: Initialize $\boldsymbol{\theta}$ randomly;
2: for $k = 1; k \leq M; k++$ do
3:     Sample the region $\Omega$ with a uniform distribution to generate the training set $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$;
4:     Minimize the loss function in equation (2) by the following iteration.
5:     while not converge do
6:         Forward propagate the neural network to get $F(\mathbf{x}_i, \boldsymbol{\theta}), i = 1, 2, \cdots, n$;
7:         Back propagate the neural network to update $\boldsymbol{\theta}$;
8:         Sample $\Omega$ with a uniform distribution to generate the test set and evaluate the generalization error on
   the test set;
9:     end while
10: end for
11: Output $\boldsymbol{\theta}$ and obtain the approximate solution of the Fredholm integral equation;
```

Numerical Experiments

In this section, several Fredholm integral equations are numerically solved using algorithm 1. In the numerical experiments, $n = 1000$ points in Ω are randomly sampled uniformly as the training set to train the deep residual neural network, and the number of training iterations is $M = 2000$. The neural network consists of one input layer, two blocks of residual neural network shown in Fig. 1, and one output layer. There are 30 neurons in the second layer and the forth layer and 10 neurons in the other layers. The ReLU function is used as the active function in the neural network. Minimization is realized by "AdamOptimizer" [31] built in TensorFlow (version 1.13.1) with a learning rate 0.001.

To measure the efficiency of the deep learning method for solving the Fredholm integral equations, we consider several examples whose exact solutions are known. Denote $f^*(\mathbf{x}_1), f^*(\mathbf{x}_2), \cdots, f^*(\mathbf{x}_n)$ as the exact solutions at n points $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$ in the test set. Define the generalization error between the exact solution $f^*(\mathbf{x})$ of the integral equation and the approximate solution $F(\mathbf{x}, \boldsymbol{\theta})$ obtained by using the deep residual neural network as

$$error = \frac{1}{n} \sum_{i=1}^{n} \left(f^*(\mathbf{x}_i) - F(\mathbf{x}_i, \boldsymbol{\theta}) \right)^2 .$$

The generalization error is evaluated for each example in the following numerical experiments.

Example 1 Consider the three-dimensional Fredholm integral equation

$$f(x, y, z) + \int_1^2 \int_1^2 \int_1^2 k(x, y, z, s, t, v) f(s, t, v) ds dt dv = g(x, y, z), \tag{5}$$

where $\Omega = [1, 2]^3$,

$$k(x, y, z, s, t, v) = e^{-xs-yt-zv},$$

and

$$g(x, y, z) = 1 - \frac{1}{xyz}(e^{-2x} - e^{-x})(e^{-2y} - e^{-y})(e^{-2z} - e^{-z}).$$

The exact solution of Eq. (5) is $f^*(x, y, z) = 1$.

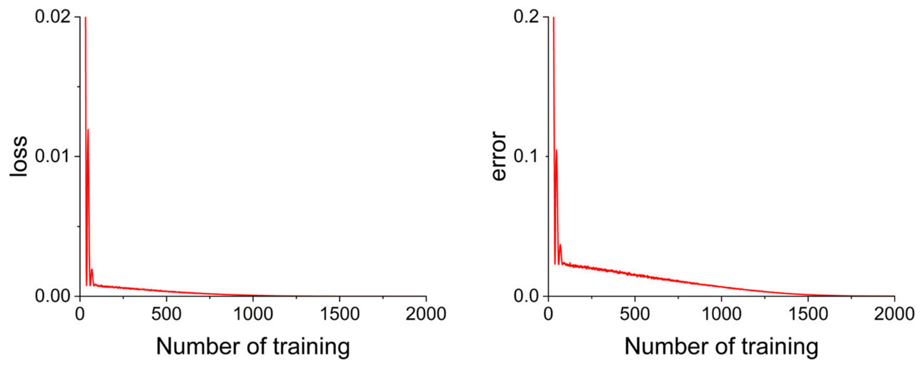

Fig. 2 Convergence of the loss function (left) and the generalization error (right) of Example 1

Table 1 Partial iterative results of the loss function and the generalization error for Example 1

Number of training	*loss*	*error*	Number of training	*loss*	*error*
1	0.8958	0.9446	800	3.498e-4	0.0161
60	0.0428	0.2040	1200	9.472e-5	0.0080
80	0.008	0.0804	1600	3.852e-5	0.0051
409	9.937e-4	0.0270	2000	1.917e-5	0.0035

For Example 1, the convergence of the loss function and the generalization error are shown in Fig. 2. Some typical iteration data of the loss function ($loss$) and the generalization error ($error$) are given in Table 1. The loss function converges to 10^{-5}, and the generalization error converges to 10^{-3}.

Example 2 Consider a high-dimensional version of the three-dimensional Fredholm integral Eq. (5), that is

$$f(\mathbf{x}) + \int\limits_{\Omega} k(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y} = g(\mathbf{x}),$$

where $\Omega = [1, 2]^m$, $\mathbf{x} = \{x_1, x_2, \cdots, x_m\}$, $\mathbf{y} = \{y_1, y_2, \cdots, y_m\}$,

$$k(\mathbf{x}, \mathbf{y}) = e^{-x_1 y_1 - \cdots - x_m y_m}$$

and

$$g(\mathbf{x}) = 1 + (-1)^m \frac{1}{x_1 x_2 \cdots x_m} (e^{-2x_1} - e^{-x_1})(e^{-2x_2} - e^{-x_2}) \cdots (e^{-2x_m} - e^{-x_m}).$$

The exact solution is $f^*(\mathbf{x}) = 1$.

For Example 2, when the dimension $m = 100$, the convergence of the loss function and the generalization error are shown in Fig. 3. Some typical iteration data of the loss function ($loss$) and the generalization error ($error$) are given in Table 2. The loss function converges to 10^{-4}, and the generalization error converges to 10^{-3}.

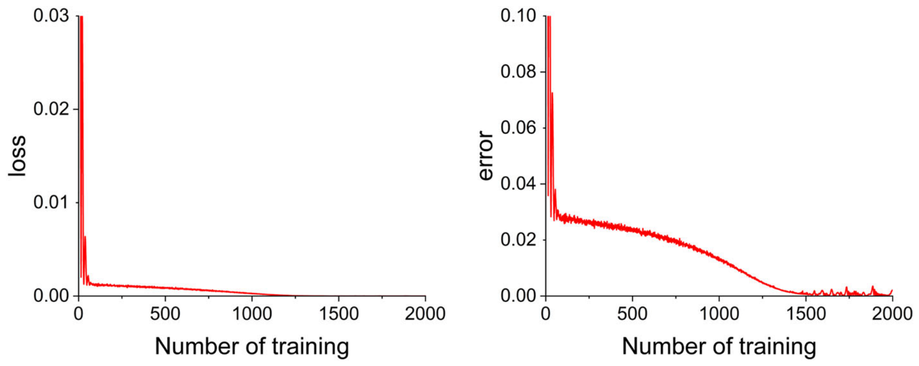

Fig. 3 Convergence of the loss function (left) and the generalization error (right) of Example 2

Table 2 Partial iterative results of the loss function and the generalization error for Example 2 when the dimension $m = 100$

Number of training	*loss*	*error*	Number of training	*loss*	*error*
1	1.264	1.123	800	5.814e-4	0.0195
32	0.0131	0.1055	1100	4.195e-4	0.0162
300	0.0012	0.0277	1600	2.239e-4	0.0119
400	9.805e-4	0.0254	2000	1.417e-4	0.0094

Example 3 Consider the four-dimensional Fredholm integral equation

$$f(x, y, z, w) - xyzw \int_0^1 \int_0^1 \int_0^1 \int_0^1 f(s, t, v, r) ds dt dv dr = \frac{15}{16} xyzw \tag{6}$$

where $\Omega = [0, 1]^4$,

$$k(x, y, z, w, s, t, v, r) = xyzw,$$

and

$$g(x, y, z, w) = \frac{15}{16} xyzw.$$

The exact solution is $f^*(x, y, z, w) = xyzw$.

For Example 3, the convergence of the loss function and the generalization error are shown in Fig. 4, and some typical iteration data of the loss function ($loss$) and the generalization error ($error$) are given in Table 3. The loss function and the generalized error function synchronously converge very fast to a stable state. The loss function converges to 10^{-3}, and the generalization error converges to 10^{-2}.

Example 4 Consider a high-dimensional version of the four-dimensional Fredholm integral Eq. (6), that is

$$f(\mathbf{x}) - x_1 x_2 \cdots x_m \int_{\Omega} f(\mathbf{y}) d\mathbf{y} = \left(1 - \frac{1}{2^m}\right) x_1 x_2 \cdots x_m$$

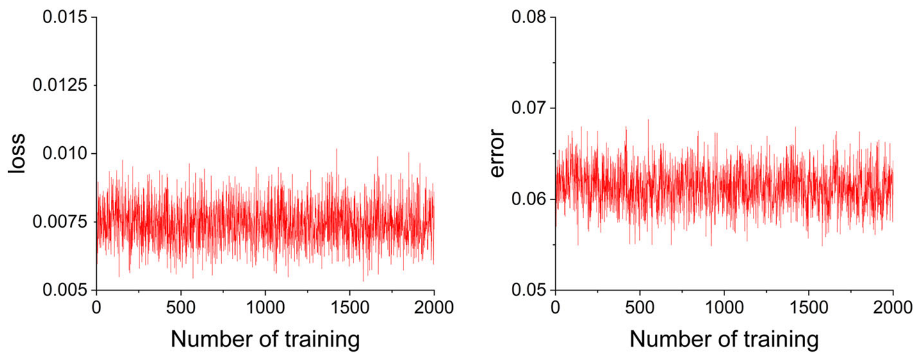

Fig. 4 Convergence of the loss function (left) and the generalization error (right) of Example 3

Table 3 Partial iterative results of the loss function and the generalization error for Example 3

Number of training	$loss$	$error$	Number of training	$loss$	$error$
1	0.0167	0.1190	1100	0.0074	0.0621
100	0.0070	0.0602	1400	0.0079	0.0603
500	0.0075	0.0632	1700	0.0064	0.0585
800	0.0064	0.0614	2000	0.0066	0.0606

where $\mathbf{x} \in [0, 1]^m$, $\mathbf{x} = \{x_1, x_2, \cdots, x_m\}$, $\mathbf{y} = \{y_1, y_2, \cdots, y_m\}$,

$$k(\mathbf{x}, \mathbf{y}) = x_1 x_2 \cdots x_m,$$

and

$$g(\mathbf{x}) = \left(1 - \frac{1}{2^m}\right) x_1 x_2 \cdots x_m.$$

The exact solution of the equation is $f^*(\mathbf{x}) = x_1 x_2 \cdots x_m$.

For Example 4, when the dimension $m = 100$, the convergence of the loss function and the generalization error are shown in Fig. 5, and some typical iteration data of the loss function ($loss$) and the generalization error ($error$) are given in Table 4. The loss function converges to 10^{-7}, and the generalization error converges to 10^{-4}.

Discussion and Conclusions

In this paper, we propose a deep learning method based on the residual neural network to solve numerically the linear Fredholm integral equations of the second kind. The output of the deep residual network is used as the numerical solution. The loss function is defined using the Fredholm integral equation. The loss function is optimized by Adam method built in TensorFlow. Then the numerical results, including high-dimensional problems, confirm the efficiency of the method. The main advantage of this method is that it can solve high-dimensional Fredholm integral equations with a cost less sensitive to the dimensionality of the problem.

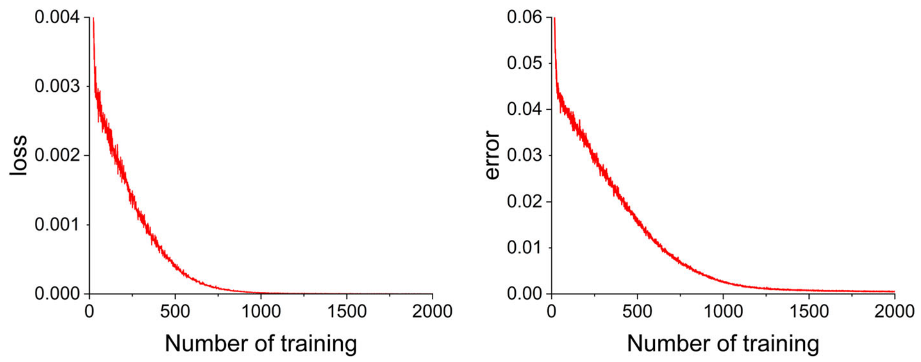

Fig. 5 Convergence of the loss function (left) and the generalization error (right) of Example 4

Table 4 Partial iterative results of the loss function and the generalization error for Example 4 when the dimension $m = 100$

Number of training	*loss*	*error*	Number of training	*loss*	*error*
1	0.6974	0.8341	1114	9.910e-6	0.0023
500	0.0010	0.0256	1243	3.728e-6	0.0010
800	2.731e-4	0.0129	1337	7.648e-7	4.484e-4
1000	4.119e-5	0.0050	2000	2.826e-7	1.718e-4

The accuracy of the residual neural network is not as good as that of the traditional method, such as the Galerkin method. Some error analysis of the neural network has been discussed in [32–34]. But so far rigorous error analysis for neural network can not be given yet. The error of the neural network consists of three parts, that is the error between the space of the output of the neural network and the exact solution of the Fredholm integral equation, the optimization error in Eq. (3), and the approximation error in Eq. (4). The error in our numerical experiments has a good accuracy compared to the error of the Monte Carlo method $1/\sqrt{n}$ in Eq. (4). In the future we will explore more techniques or theory to improve the convergent accuracy. Additionally, we will try to construct a deep residual neural network to solve the Volterra integral equations.

Acknowledgements The research of Congming Jin was supported by the National Natural Science Foundation of China (No. 11571314).

Funding The authors have not disclosed any funding.

Availability of data and materials All data generated or analyzed during this study are included in this manuscript.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Michalski, K.A., Mosig, J.R.: Multilayered media Green's functions in integral equation formulations. IEEE T. Antenn. Propag. **45**(3), 508–519 (2002)
2. Yang, P., Liou, K.N.: Geometric-optics-integral-equation method for light scattering by nonspherical ice crystals. Appl. Opt. **35**, 6568–6584 (1996)
3. De Bonis, M.C., Stanić, M.P., Tomović Mladenović, T.V.: Nystörm methods for approximating the solutions of an integral equation arising from a problem in mathematical biology. Appl. Numer. Math. **171**, 193–211 (2022)
4. Baleanu, D., Zibaei, S., Namjoo, M., Jajarmi, A.: A nonstandard finite difference scheme for the modelling and nonidentical synchronization of a novel fractional chaotic system. Adv. Differ. Equ. **2021**, 308 (2021)
5. Baleanu, D., Hassan Abadi, M., Jajarmi, A., Zarghami Vahid, K., Nieto, J.J.: A new comparative study on the general fractional model of COVID-19 with isolation and quarantine effects. Alex. Eng. J. **61**(6), 4779–4791 (2022)
6. Erturk, V.S., Godwe, E., Baleanu, D., Kumar, P., Asad, J., Jajarmi, A.: Novel fractional-order Lagrangian to describe motion of beam on nanowire. Acta Phys. Pol. A. **140**(3), 265–272 (2021)
7. Jajarmi, A., Baleanu, D., Zarghami Vahid, K., Mohammadi Pirouz, H., Asad, J.H.: A new and general fractional Lagrangian approach: a capacitor microphone case study. Results Phys. **31**, 104950 (2021)
8. Atkinson, K.: Iterative variants of the Nyström method for the numerical solution of integral equations. Numer. Math. **22**(1), 17–31 (1974)
9. Khorrami, N., Shamloo, A.S., Parsa Moghaddam, B.: Nyström method for solution of Fredholm integral equations of the second kind under interval data. J. Intell. Fuzzy Syst. **36**(3), 2807–2816 (2019)
10. Han, G., Wang, R.: Richardson extrapolation of iterated discrete Galerkin solution for two-dimensional Fredholm integral equations. J. Comput. Appl. Math. **139**(1), 49–63 (2002)
11. Babolian, E., Shahsavaran, A.: Numerical solution of nonlinear Fredholm integral equations of the second kind using Haar wavelets. J. Comput. Appl. Math. **225**(1), 87–95 (2009)
12. Asady, B., Hakimzadegan, F., Nazarlue, R.: Utilizing artificial neural network approach for solving two-dimensional integral equations. Math. Sci. **8**, 117 (2014)
13. Effati, S., Buzhabadi, R.: A neural network approach for solving Fredholm integral equations of the second kind. Neural Comput. Appl. **21**(5), 843–852 (2012)
14. Wang, K., Wang, Q.: Taylor collocation method and convergence analysis for the Volterra-Fredholm integral equations. J. Comput. Appl. Math. **260**, 294–300 (2014)
15. Jin, C., Ding, J.: Solving Fredholm integral equations via a piecewise linear maximum entropy method. J. Comput. Appl. Math. **304**, 130–137 (2016)
16. Saeed, T., Sabir, Z., Alhodaly, M.S., Alsulami, H.H., Guerrero Sánchez, Y.: An advanced heuristic approach for a nonlinear mathematical based medical smoking model. Results Phys. **32**, 105137 (2022)
17. Sabir, Z., Wahab, H.A., Javeed, S., Baskonus, H.M.: An efficient stochastic numerical computing framework for the nonlinear higher order singular models. Fractal Fract. **5**, 176 (2021)
18. Sabir, Z.: Stochastic numerical investigations for nonlinear three-species food chain system. Int. J. Biomath. **2250005**, (2021)
19. Sabir, Z., Wahab, H.A.: Evolutionary heuristic with Gudermannian neural networks for the nonlinear singular models of third kind. Phys. Scr. **96**, 125261 (2021)
20. Sabir, Z., Ali, M.R., Sadat, R.: Gudermannian neural networks using the optimization procedures of genetic algorithm and active set approach for the three-species food chain nonlinear model. J. Ambient Intell. Human. Comput. (2022). https://doi.org/10.1007/s12652-021-03638-3
21. Wang, B., Wang, Y., Gómez-Aguilar, J.F., Sabir, Z., Raja, M.A.Z., Jahanshahi, H., Alassafi, M.O., Alsaadi, F.E.: Gudermannian neural networks to investigate the Liénard differential model. Fractals (2021). https://doi.org/10.1142/S0218348X22500505
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition, (2016), Las Vegas, USA
23. Sun, Y., Chen, Y., Wang, X., Tang, X.: Deep learning face representation by joint identification-verification. NIPS **27**, (2014)
24. Graves, A., Mohamed, A. R., Hinton, G.: Speech recognition with deep recurrent neural networks. IEEE International Conference on Acoustics, Speech and Signal Processing, May 2013, Vancouver, Canada
25. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Proc. Mag. **29**(6), 82–97 (2012)
26. Tom, Y., Devamanyu, H., Soujanya, P., Erik, C.: Recent trends in deep learning based natural language processing. IEEE Comput. Intell. Mag. **13**(3), 55–75 (2018)

27. Beck, C.E.W., Jentzen, A.: Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. J. Nonlinear Sci. **29**, 1563–1619 (2019)
28. Han, J., Jentzen, A.: Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Commun. Math. Stat. **5**, 349–380 (2017)
29. Yu, B.: The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Commun. Math. Stat. **6**, 1–12 (2018)
30. Zhang, L., Han, J., Wang, H., Car, R.E.W.: Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. Phys. Rev. Lett. **120**(14), 143001 (2018)
31. Kingma, D. P., Ba, J. L.: Adam: A Method for stochastic Optimization. The International Conference on Learning Representations (ICLR), May 2015, San Diego, USA
32. Barron, A.R.: Universal approximation bounds for superpositions of a sigmoidal function. IEEE T. Inform. Theory. **39**(3), 930–945 (1993)
33. Barron, A. R., Klusowski, J. M.: Approximation and estimation for high-dimensional deep learning networks. arXiv:1809.03090v2, (2018)
34. Ma, C., Wu, L., Wojtowytsch, S., Wu, L.: Towards a mathematical understanding of neural network-based machine learning: what we know and what we don't. arXiv:2009.10713v2, (2020)