Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Fractional Quantum Chaotic Maps for Neural Network Enhancement

Nafiseh Hematpour

hematpourbarani@itu.edu.tr

Kadir Has University

Fatemeh Gharari

University of Mohaghegh Ardabili

Berna Ors Yalcin

Istanbul Technical University

Research Article

Keywords: Hierarchical rational chaotic map, Quantum logistic map, Quantum tent map, Fractional differences, Neural network, Mean square error

Posted Date: April 8th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-7612466/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Fractional Quantum Chaotic Maps for Neural Network Enhancement

Nafiseh Hematpour[1,2*], Fatemeh Gharari[3] and Berna Ors Yalcin[1]

[1*]Faculty of Electrical and Electronics Engineering, Istanbul Technical University, Sariyer, Istanbul, 34485, Istanbul, Turkiye.
[2]Mechatronics Engineering Department, Kadir Has University, Istanbul, Turkiye.
[3]Department of Statistics and Computer Science, University of Mohaghegh Ardabili, Ardabil 56199-11367, Iran.

*Corresponding author(s). E-mail(s): hematpourbarani@itu.edu.tr;
Contributing authors: f.gharari@uma.ac.ir; orssi@itu.edu.tr;

Abstract

This paper presents a novel framework for enhancing neural network performance through the integration of fractional and quantum chaotic maps. We introduce a fractional hierarchical rational chaotic map alongside quantum adaptations of the logistic and tent maps, analyzing their dynamic properties via bifurcation and time series diagrams. These chaotic mechanisms are embedded into gradient-based learning algorithms—batch, mini-batch, and stochastic gradient descent—to improve training efficiency and predictive accuracy.
Empirical evaluations on three real-world regression tasks (housing prices, bike sharing, and concrete strength) demonstrate consistent performance gains across all models. The inclusion of chaotic maps notably accelerates convergence, reduces training error, and improves weight distribution within networks. Our findings underscore the transformative potential of fractional and quantum chaos in optimizing neural learning systems for complex, real-world applications.

Keywords: Hierarchical rational chaotic map, Quantum logistic map, Quantum tent map, Fractional differences, Neural network, Mean square error

1 Introduction

Artificial Intelligence (AI) includes machine learning (ML), which is the automatic recognition of patterns in datsets [1]. According to the current definition, ML is a computer programming technique that optimizes performance metrics that depend on data [2–4]. The most common machine learning model is a Neural Network (NN). The performance of NN is highly dependent on the variation of connection weight and network structure.

Batch Gradient Descent (BGD), Mini Batch Gradient Descent (MBGD), and Stochastic Gradient Descent (SGD) are three different variations of gradient descent algorithms used in training NNs. In BGD method, the entire training dataset is used to compute the gradient of the cost function. The weights are then updated based on this average gradient. This approach can be computationally expensive for large datasets but can lead to more stable convergence. In SGD, the gradient is computed and the weights are updated for each individual training example. This can lead to faster convergence but may result in more noisy updates and slower convergence in the long run. In MBGD, the training dataset is divided into smaller batches of a fixed size. The gradient is computed for each batch, and the weights are updated based on the average gradient of the batch. This method strikes a balance between the efficiency of SGD and the stability of BGD.

Using chaotic maps for NN training involves utilizing chaotic dynamics to improve the performance and convergence of NN training algorithms. Chaotic maps can be used to generate random sequences of numbers that can be used to initialize the weights of a NN or to optimize the learning rate during training. One example of using chaotic maps for NN training is the use of the logistic map to generate random numbers for weight initialization in the NN. This can help to avoid getting stuck in local minima during training and improve the overall performance of the network. Another example is using chaotic maps to optimize the learning rate during training, where the dynamics of the chaotic map can be used to adjust the learning rate dynamically during the training process.

Recently, some attempts have been made to use a modified batch algorithm based on applying chaotic maps: the results of reference [5] have led to more accurate models for predicting time series. In [6, 7], the NN model is optimized using a chaotic map and then they are applied for prediction. To obtain more accurate predictions, authors in [8] have entered the two layers such that one of them is the chaotic nature of the data. Different chaotic maps differ in terms of chaotic area, the number of control parameters, and initial conditions [9, 10]. The use of these maps has been effective in improving optimizations such as Particle Swarm Optimization (PSO) [9] and harmony search algorithm [10]. The hierarchy of rational order families of chaotic maps presents an interesting description of the simultaneous production and consumption of entropies [11]. The Hierarchy Rational Chaotic Map (HRCM) was the best chaotic map for improving PSO performance [9].

The problem with the used chaotic maps is about having few control parameters and less complicated dynamic. Fractionalization is employed as a powerful tool to model complex data and complex systems [12–14]. In this paper, we increase the complexity of the chaotic maps by increasing parameters. To accomplish this, we

essentially suggest their generalized versions through fractionalization. Especially, we extract a fractional version of the HRCM and apply to improve NN models.

Motivation

Despite the successful application of classical chaotic maps, their potential in NN optimization remains under-explored, particularly concerning fractional chaotic maps. The unique properties of fractional chaotic maps, such as their ability to capture memory effects and long-range dependencies, offer promising avenues for enhancing the performance of NN training algorithms.

Contributions

This study aims to bridge this gap and contribute to the field in the following ways:

- Novel fractional chaotic map: We introduce a fractional adaptation of the hierarchical rational chaotic map, quantum logistic map, and quantum tent map, enriching the toolbox of chaotic systems available for NN optimization.
- Enhanced gradient-based learning: We integrate this fractional map, along with quantum versions of the logistic and tent maps, into gradient batch descent, mini-batch gradient descent, and stochastic gradient descent algorithms.
- Real-world application: We demonstrate the effectiveness of our approach by applying the modified NNs to the challenging tasks of predicting house prices based on **13** influential features, bike sharing based on **11** influential features, and concrete compressive strength based on **8** influential features.

We organize the article as follows: Section 2 provides a recent literature review. In Section 3, fractional versions of the hierarchy of rational order chaotic, quantum logistic, and quantum tent maps are given as well as their bifurcation diagrams and time series. These generalized maps are applied to gradient-based NN algorithms, including gradient batch descent, mini-batch gradient descent, and stochastic gradient descent to enhance their performance. All improved NN trainings are presented in Section 4. Section 5 presents the results of integrating fractional quantum chaotic maps into neural network training. Section 6 compares fractional quantum chaotic maps with their non-fractional counterparts. Concluding remarks are presented at the end.

2 Literature Review

The integration of chaotic maps into NN architectures has emerged as a promising avenue to enhance performance and security. This review explores recent advancements in the application of classical, fractional, and quantum chaotic maps, particularly their impact on optimizing gradient-based learning algorithms like gradient batch descent, mini-batch gradient descent, and stochastic gradient descent NNs. A novel NN Nonlinear Predictive Control (NNPC) strategy utilizing the Tent-map Chaos Optimization Algorithm (TCOA) is introduced in reference [15]. In this paper, the feed-forward NN serves as the multi-step predictive model, while the TCOA enhances convergence and accuracy through nonlinear rolling optimization. Several studies have explored the use of Artificial Neural Networks (ANNs) and chaotic maps for Chaotic Random

Number Generation (CRNG) and their application in image encryption. For example, Satre and Joshi proposed quantum image cryptography methods based on both discrete chaotic maps [16] and continuous chaotic maps [17], highlighting the potential of chaos-enhanced cryptographic schemes in quantum settings. Similarly, Wu et al. [18] introduced a deep neural network approach that integrates QLSTM and chaos synchronization control for enhancing quantum image encryption, demonstrating the effectiveness of combining chaotic dynamics with quantum learning models.

Patel et al. proposed a CRNG based on a train-less NN. In their approach, the weights and biases of the network are initialized using a cubic map, while combined Logistic-Sine and Logistic-Tent chaotic maps serve as activation functions. With each forward pass, the network parameters are updated to generate CRNG outputs, which are then used in an image encryption algorithm. This method highlights the potential of using chaotic maps to enhance the randomness and security of generated numbers for cryptographic purposes [19]. Similarly, Maddodi et al. presented a CRNG built on a NN architecture. Instead of training the network with data, they initialize the weights and biases using a chaotic cubic map. This technique provides a novel approach to CRNG by leveraging the inherent complexity of chaotic systems within the NN framework [20]. He et al. introduced a CRNG based on the structure of Long Short-Term Memory (LSTM) networks. They initialize the input, output, and forget gates of LSTM cells using a chaotic Two-Dimensional Coupled Map Lattice. This CRNG is applied to image encryption, where the LSTM networks take image pixels as inputs and perform permutation and diffusion operations as part of the encryption process. This study demonstrates the effectiveness of combining LSTM networks with chaotic maps to enhance the security of image encryption algorithms [21]. Koubaâ et al. also proposed an LSTM-based CRNG for image encryption. Their method involves using coupled chaotic maps to initialize the parameters of the LSTM network. By integrating chaotic dynamics into the initialization process, this approach aims to improve the unpredictability and security of the CRNG for cryptographic applications [22]. Reference [23] introduces a new Chaos-based Pseudo-Random Number Generator (CPRNG) that combines ANNs with chaotic systems. These studies collectively highlight the potential of combining NN architectures with chaotic maps to develop robust CPRNGs for various applications, particularly in the field of image encryption. The use of chaotic maps introduces additional complexity and variability, enhancing the security and effectiveness of the generated random numbers. In 2022, Wang et al. [24] integrated discrete wavelet transforms and the Lorentz map with a convolutional neural network (CNN), applying this combination to fault diagnosis. The traditional NNs used in the above fault diagnosis schemes rely on the gradient descent algorithm. While this algorithm descends in the fastest decreasing direction of the function, the local gradient on the error surface often doesn't point to the global minimum, resulting in a zigzag convergence path. Consequently, it requires many steps to reach the global minimum or can easily fall into local minima [25]. The sensitivity to initial conditions in chaotic systems enhances NN' search efficiency, generalization, and pattern recognition abilities compared to traditional NN. The ergodicity and sensitivity of chaos significantly improve NN performance [26]. In reference [27], a CNN combined with a Logistic chaotic map is proposed, along with a corresponding fault diagnosis approach,

which significantly improves the accuracy of bearing fault diagnosis. Reference [28] proposes a two-dimensional discrete model of a bi-neuron Hopfield Neural Network (HNN) with sine activation function, demonstrating its complex dynamics and stability. The model is used to design four Pseudo-Random Number Generators (PRNGs) with high randomness and strong resistance to chaos degradation, validated through numerical analysis and FPGA hardware implementation. In reference [29], ANN forecasting models can efficiently account for fluctuating demand and market conditions, unlike the HoltsWinter method which struggles with missing or limited data. However, ANN is time-consuming due to the need to carefully select learning parameters, hidden neurons, and epochs during training, as it is sensitive to these parameters. Providing a strong defense against hostile attacks while retaining excellent accuracy is challenging in NNs. Reference [30], study demonstrates that chaotic stochastic quantization significantly improves robustness by up to **43%** compared to deterministic quantization, with Lorenz and Henon noise distributions outperforming traditional uniform and Gaussian noise in enhancing model stability and defense against attacks.

Our classical and quantum chaotic maps are more complex and higher-dimensional, featuring higher Lyapunov exponents, which contribute to their enhanced dynamic complexity and robustness. According to a recent review article, a chaotic quantum map has achieved the highest entropy value of **7.999995** [31]. The introduction of fractional calculus has led to the creation of fractional chaotic maps, which have deeper dynamics and memory effects than their integer-order counterparts. The ongoing development of research in discrete fractional chaotic systems [32–35] motivated our work to explore more complex classic and quantum chaotic maps. Simultaneously, the exploration of fractional version of quantum chaotic maps has opened new possibilities for introducing more complex behavior into NN models.

Future research will likely focus on developing more sophisticated chaotic maps, exploring hybrid models, and broadening their application to new domains. A key challenge will be devising efficient algorithms for training and optimizing these increasingly complex NNs. It is what we are trying to do in this paper.

3 Model

This section introduces a fractional adaptation of the hierarchical rational chaotic map, as well as quantum versions of the logistic and tent maps to incorporate them into gradient-based learning algorithms such as gradient batch descent, mini-batch gradient descent, and stochastic gradient descent. This task aims to improve the training and predictive capabilities of NN models.

3.1 A fractional hierarchy of rational order chaotic maps for time scales

The Hierarchy of Rational order Chaotic Maps (HRCM) [11] is defined as:

$$x_{n+1} = \frac{a_1}{a_2} \times \frac{1 - 3x_n^2}{3x_n - x_n^3} \pm \sqrt{1 + \left(\frac{a_1}{a_2} \times \frac{1 - 3x_n^2}{3x_n - x_n^3}\right)^2} \tag{1}$$

where $\boldsymbol{x}$ is the state variable and $\boldsymbol{a_i, i = 1, 2,}$ denote the control parameters. Here, we proposed a one-dimensional fractional version of the hierarchy of rational order chaotic map (FHRCM) in the Caputo sense, and then imply it to improve NN training BGD performance in the next section.
A fractional version of system (1) is as follows,

$$
\begin{aligned}
{}^{C}\boldsymbol{\Delta}_{a}^{\nu} x(t) = & \frac{\boldsymbol{a_1}}{\boldsymbol{a_2}} \times \frac{\boldsymbol{1 - 3x^2(t + \nu - 1)}}{\boldsymbol{3x(t + \nu - 1) - x^3(t + \nu - 1)}} \\
& \pm \sqrt{\boldsymbol{1 + \left(\frac{a_1}{a_2} \times \frac{1 - 3x^2(t + \nu - 1)}{3x(t + \nu - 1) - x^3(t + \nu - 1)} \right)^2}},
\end{aligned}
\tag{2}
$$

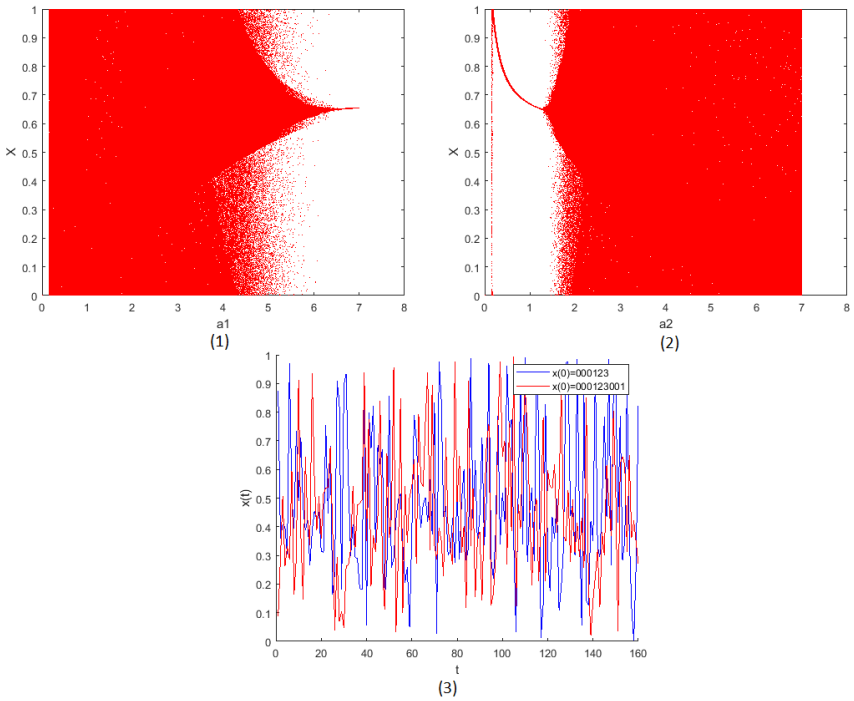
where ${}^{C}\boldsymbol{\Delta}_{a}^{\nu}$ is the $\boldsymbol{\nu}-$th Caputo left fractional difference operator, $\boldsymbol{\nu > 0}$ is the fractional order, and $\boldsymbol{a} \in \mathbb{R}$ is the starting point. The explicit numerical formula of Eq. (2), which we denote it by FHRCM, can be expressed as:

$$
\begin{aligned}
\boldsymbol{x_n} = & \boldsymbol{x_0} + \frac{\boldsymbol{1}}{\boldsymbol{\Gamma(\nu)}} \sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \frac{\boldsymbol{\Gamma(n - j + \nu)}}{\boldsymbol{\Gamma(n - j + 1)}} \\
& \left(\boldsymbol{\frac{a_1}{a_2} \times \frac{1 - 3x_{j-1}^2}{3x_{j-1} - x_{j-1}^3} \pm \sqrt{1 + \left(\frac{a_1}{a_2} \times \frac{1 - 3x_{j-1}^2}{3x_{j-1} - x_{j-1}^3} \right)^2}} \right)
\end{aligned}
\tag{3}
$$

To find the relation between "$\boldsymbol{x(t)}$" in Eq. (2) and "$\boldsymbol{x_n}$" in Eq. (3), and other details, see Appendix A.
The bifurcation diagram and time series for the FHRCM are given in Fig. 1 by taking into account the fractional order $\boldsymbol{\nu = 1.2}$ and the control parameters $\boldsymbol{a_1 = 2.61}$, $\boldsymbol{a_2 = 3.168}$. Figures. 1.1-2 are plotted for the FHRCM for $\boldsymbol{X}$ with the initial value $\boldsymbol{x_0 = 0.3456}$; Fig. 1.1 is plotted for $\boldsymbol{X}$ with respect to $\boldsymbol{a_2}$ and Fig. 1.2 is with respect to $\boldsymbol{a_1}$. Figures. 1.1-2 present the chaotic area of the chaotic map; $\boldsymbol{a_1}$ $\boldsymbol{(a_2)}$ should be selected between $\boldsymbol{0}$ $\boldsymbol{(2)}$ and $\boldsymbol{4}$ $\boldsymbol{(7)}$ when $\boldsymbol{a_2}$ $\boldsymbol{(a_1)}$ is $\boldsymbol{2.61}$ $\boldsymbol{(3.168)}$. The time series of the FHRCM is seen in Fig. 1.3. It shows this map is sensitive to the initial value.

3.2 A fractional quantum logistic map for time scales

Since its introduction in 1976, the logistic map has gained popularity in academia, particularly in the field of chaos and fractals [36–39]. Reference [40] proposes a quantum logistic map coupled with the logistic map. As a three-dimensional map, it was discovered that the quantum logistic map exhibits deeper dynamical features, making it more suitable for use in the field of information security. However, quantum logistic maps have received less attention than logistic maps. We researched the fractional generalization of quantum logistic maps, which had never been studied previously. We begin this section by recalling a definition from reference [41], where a generalized



EQ

Fig. 1 Bifurcation diagram of the FHRCM for (1) $\boldsymbol{x_0 = 0.3456}$, $\boldsymbol{\nu = 1.2}$, and $\boldsymbol{a_2 = 2.61}$, with respect to $\boldsymbol{a_1}$, (2) $\boldsymbol{x_0 = 0.3456}$, $\boldsymbol{\nu = 1.2}$, and $\boldsymbol{a_1 = 2.61}$, with respect to $\boldsymbol{a_2}$, (3) Time series of the FHRCM for $\boldsymbol{x}$ with $\boldsymbol{a_1 = 2.61, a_2 = 3.168}$, and $\boldsymbol{\nu = 1.2}$.

Hamilton Dick is introduced:

$$\boldsymbol{H = \mathbf{a}^{\dagger}\mathbf{a} + \omega_A J_z + \frac{\gamma}{\sqrt{N}}(\mathbf{a}^{\dagger} + \mathbf{a})(J_- + J_+) + \frac{\gamma}{\sqrt{N}} V(J_-, J_+)\Sigma_n \delta(t - nT),} \quad (4)$$

where $\mathbf{a}$ and $\mathbf{a}^{\dagger}$ are bosonic annihilation and creation operators, respectively. We consider that $\hbar = \mathbf{1}$, $\boldsymbol{\omega_A = \tilde{\omega}_A/\tilde{\omega}_f \geq 0}$, $\boldsymbol{\tilde{\omega}_f}$ is the field of frequency, and $\boldsymbol{\gamma = \tilde{\gamma}/\tilde{\omega}_f}$ is the coupling parameter [42]. In addition, $\boldsymbol{J_z}$ and $\boldsymbol{J_\pm}$ are the atomic relative population operator and the atomic transition operator. The commutation relations of the fermionic operator $\boldsymbol{J}$ are:

$$\boldsymbol{[J_+, J_-] = 2J_z,\ [J_z, J_\pm] = \pm J_\pm,\ [\mathbf{a}, \mathbf{a}^{\dagger}] = 1.}$$

A Quantum Logistic Map (QLM) is a logistic map that includes quantum corrections. Researchers set out to examine the consequences of quantum corrections $\boldsymbol{J_\pm =< J_\pm > + \delta J_\pm}$, where $\boldsymbol{\delta J_\pm}$ is a quantum fluctuation about $\boldsymbol{< J_\pm >}$, and $\boldsymbol{< J_\pm >}$ is

the mean value of $J_{\pm}$. The quantum logistic map [43] is described as

$$\begin{cases} X_{n+1} = r(x_n - x_n^2) - ry_n \\ Y_{n+1} = -y_n + re^{-\beta}((1 - x_n + e^{2\beta} - x_n e^{2\beta})y_n - z_n x_n - e^{2\beta} z_n x_n) \\ Z_{n+1} = -z_n e^{2\beta} + re^{\beta}(2z_n - 2x_n z_n - x_n y_n - y_n x_n), \end{cases} \quad (5)$$

where $< \delta J_+ \delta J_- >= y_n$, $< \delta J_+ \delta J_+ >= z_n$, $< J_+ >= x_n$, respectively. Besides, β and r denote the control parameters. We present a fractional version of the QLM as follows:

$$\begin{cases} {}^{C}\Delta_a^{\nu} x(t) = r\left[x(\eta) - x^2(\eta)\right] - ry(\eta) - x(\eta) \\ {}^{C}\Delta_a^{\nu} y(t) = -y(\eta) + re^{-\beta}[(1 - x(\eta) + e^{2\beta} - x(\eta)e^2\beta)y(\eta) \\ -z(\eta)x(\eta) - e^{2\beta} z(\eta)x(\eta)] - y(\eta) \\ {}^{C}\Delta_a^{\nu} z(t) = -z(\eta)e^{2\beta} + 2re^{\beta}[(\eta) - 2x(\eta)z(\eta) \\ -x(\eta)y(\eta) - y(\eta)x(\eta)] - z(\eta), \end{cases} \quad (6)$$

where $\eta = t + \nu - 1$, ${}^{C}\Delta_a^{\nu}$ is the $\nu-$th Caputo-like delta difference operator, ν gives the fractional order, and a demonstrates the starting point. The explicit numerical formula of of Eq. (6), which we denote by FQLM (Fractional Quantum Logistic Map), is expressed as:

$$\begin{cases} x_n = x_0 + \frac{1}{\Gamma(\nu)} \sum_{j=1}^{n} \frac{\Gamma(n-j+\nu)}{\Gamma(n-j+1)} \{r[x_{j-1} - x_{j-1}^2] - ry_{j-1} - x_{j-1}\} \\ y_n = y_0 + \frac{1}{\Gamma(\nu)} \sum_{j=1}^{n} \frac{\Gamma(n-j+\nu)}{\Gamma(n-j+1)} \{-y_{j-1} + re^{-\beta}[(1 - x_{j-1} \\ \quad + e^{2\beta} - x_{j-1}e^2\beta)y_{j-1} - z_{j-1}x_{j-1} - e^{2\beta} z_{j-1}x_{j-1}] - y_{j-1}\} \\ z_n = z_0 + \frac{1}{\Gamma(\nu)} \sum_{j=1}^{n} \frac{\Gamma(n-j+\nu)}{\Gamma(n-j+1)} \{-z_{j-1}e^{2\beta} \\ \quad + 2re^{\beta}[2z_{j-1} - 2x_{j-1}z_{j-1} - x_{j-1}y_{j-1} - y_{j-1}x_{j-1}] - z_{j-1}\}. \end{cases} \quad (7)$$

To find the relation between Eq. (6) and Eq. (7), and other details, see Appendix A. The bifurcation diagram from the FQLM is given in Fig. 2 by setting the control parameters $r = 5.5$, $b = \beta = 0.5$, and the fractional order $\nu = 1.2$. Fig. 2.1 is plotted for the FQLM with X for the initial value $x_0 = 0.3456$, $y_0 = 0.4865$, $z_0 = 0.6703$ with respect to $b = \beta$. Fig. 2.2 is plotted for X with respect to r. Fig. 2.3 is plotted for the FQLM with Y for $x_0 = 0.3456$, $y_0 = 0.4865$, $z_0 = 0.6703$ with respect to b. Fig. 2.4 is for Y with respect to r. Fig. 2.5 is plotted for the FQLM with Z for $x_0 = 0.3456$, $y_0 = 0.4865$, $z_0 = 0.6703$ with respect to $b = \beta$. Fig. 2.6 is plotted for Z with respect to r. Based on Figures, we can see the chaotic area of this chaotic map: $b = \beta$ should be selected between 0 and 10 when $r = 5.5$. The time

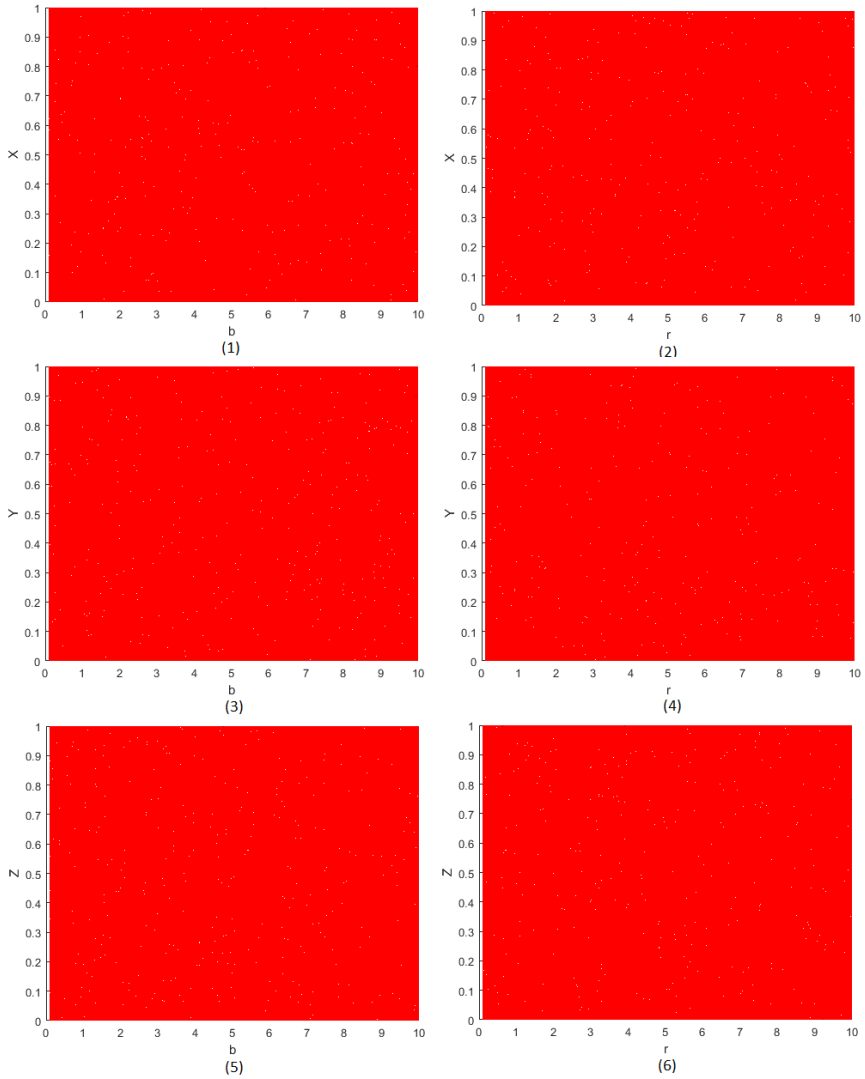


Fig. 2 Bifurcation diagram of the FQLM with $\boldsymbol{r = 5.5}$, $\boldsymbol{b = \beta = 0.5}$, $\boldsymbol{x_0 = 0.3456}$, $\boldsymbol{y_0 = 0.4865}$, $\boldsymbol{z_0 = 0.6703}$, and $\boldsymbol{\nu = 1.2}$ for (1) $\boldsymbol{X}$ with respect to $\boldsymbol{b = \beta}$, (2) $\boldsymbol{X}$ with respect to $\boldsymbol{r}$, (3) $\boldsymbol{Y}$ with respect to $\boldsymbol{b = \beta}$, (4) $\boldsymbol{Y}$ with respect to $\boldsymbol{r}$, (5) $\boldsymbol{Z}$ with respect to $\boldsymbol{b = \beta}$, (6) $\boldsymbol{Z}$ with respect to $\boldsymbol{r}$.

series from the FQLM for $\boldsymbol{X}$ are given in Fig. 3 with the control parameters $\boldsymbol{r = 5.5}$, $\boldsymbol{b = \beta = 0.5}$, and the fractional order $\boldsymbol{\nu = 1.2}$. It shows this map is sensitive to the initial value.

3.3 A fractional quantum tent map for time scales

A quantum tent map is a tent map with quantum corrections and this quantum map is extracted from the same Dicke Hamiltonian. So, $\boldsymbol{J_\pm =< J_\pm > +\delta J_\pm}$, and $< J_\pm >$

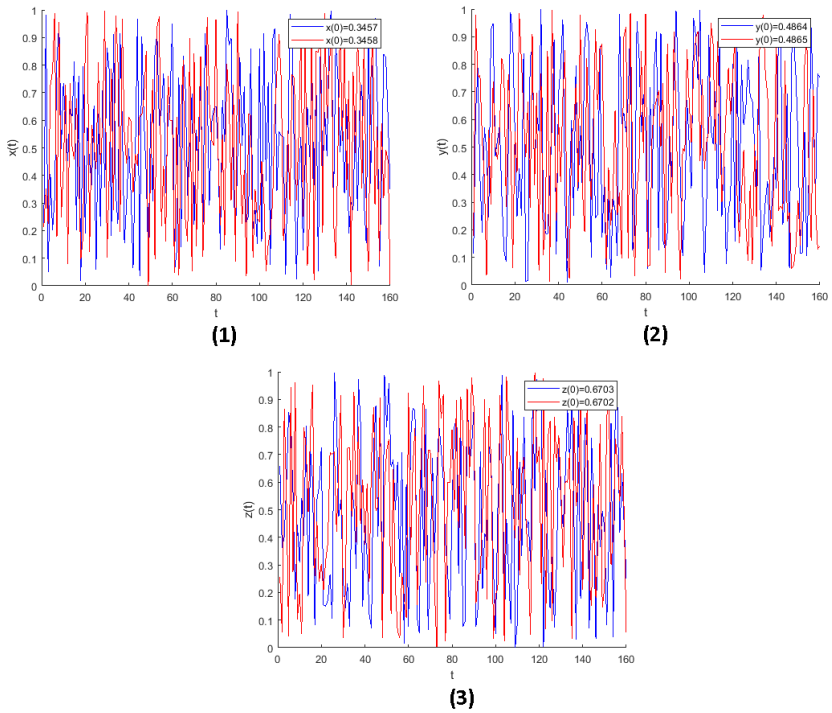


Fig. 3 Time series of the FQLM for $\boldsymbol{X}$ with $\boldsymbol{r = 5.5}$, $\boldsymbol{b = \beta = 0.5}$, and $\boldsymbol{\nu = 1.2}$.

are the same parameters. The Quantum Tent Map (QTM) [13] is described as

$$\begin{cases} \left\{ \begin{array}{l} \boldsymbol{x_{n+1} = \mu x_n} \\ \boldsymbol{y_{n+1} = -y_n + \mu y_n \left(e^{-\beta} + e^{\beta}\right)} \\ \boldsymbol{z_{n+1} = -e^{2\beta} z_n + 2\mu z_n e^{\beta}} \end{array} \right\}, \boldsymbol{x_n < 1/2} \\ \left\{ \begin{array}{l} \boldsymbol{x_{n+1} = \mu \left(1 - x_n\right)} \\ \boldsymbol{y_{n+1} = -y_n - \mu y_n \left(e^{-\beta} + e^{\beta}\right)} \\ \boldsymbol{z_{n+1} = -e^{2\beta} z_n - 2\mu z_n e^{\beta}} \end{array} \right\}, \boldsymbol{x_n \geq 1/2} \end{cases} \tag{8}$$

where $< \boldsymbol{\delta J_+ \delta J_-} >= \boldsymbol{y_n}$, $< \boldsymbol{\delta J_+ \delta J_+} >= \boldsymbol{z_n}$, $< \boldsymbol{J_+} >= \boldsymbol{x_n}$, respectively. These equations give the lowest-order quantum corrections. Besides, $\boldsymbol{\beta}$ and $\boldsymbol{\mu}$ denote control

parameters. As a fractional version of Eq. (8), we can obtain

$$\begin{cases} \left\{ \begin{array}{l} {}_{b}^{C}\boldsymbol{\Delta}^{\nu}\boldsymbol{x}(\boldsymbol{t}) = \boldsymbol{x}(\boldsymbol{t}-\boldsymbol{\nu}+\mathbf{1})(\boldsymbol{\mu}-\mathbf{1}) \\ {}_{b}^{C}\boldsymbol{\Delta}^{\nu}\boldsymbol{y}(\boldsymbol{t}) = \boldsymbol{y}(\boldsymbol{t}-\boldsymbol{\nu}+\mathbf{1})\left(\boldsymbol{\mu}(\boldsymbol{e}^{-\boldsymbol{\beta}}+\boldsymbol{e}^{\boldsymbol{\beta}})-\mathbf{2}\right) \\ {}_{b}^{C}\boldsymbol{\Delta}^{\nu}\boldsymbol{z}(\boldsymbol{t}) = \boldsymbol{z}(\boldsymbol{t}-\boldsymbol{\nu}+\mathbf{1})\left(\mathbf{2}\boldsymbol{\mu}\boldsymbol{e}^{\boldsymbol{\beta}}-\boldsymbol{e}^{\mathbf{2}\boldsymbol{\beta}}-\mathbf{1}\right), \end{array} \right\}, \boldsymbol{x}(\boldsymbol{t}) < \mathbf{1/2} \\ \left\{ \begin{array}{l} {}^{C}\boldsymbol{\Delta}_{a}^{\nu}\boldsymbol{x}(\boldsymbol{t}) = \boldsymbol{\mu}-\boldsymbol{x}(\boldsymbol{t}+\boldsymbol{\nu}-\mathbf{1})(\mathbf{1}+\boldsymbol{\mu}) \\ {}^{C}\boldsymbol{\Delta}_{a}^{\nu}\boldsymbol{y}(\boldsymbol{t}) = -\boldsymbol{y}(\boldsymbol{t}+\boldsymbol{\nu}-\mathbf{1})\left(\mathbf{2}+\boldsymbol{\mu}(\boldsymbol{e}^{-\boldsymbol{\beta}}+\boldsymbol{e}^{\boldsymbol{\beta}})\right) \\ {}^{C}\boldsymbol{\Delta}_{a}^{\nu}\boldsymbol{z}(\boldsymbol{t}) = -\boldsymbol{z}(\boldsymbol{t}+\boldsymbol{\nu}-\mathbf{1})\left(\boldsymbol{e}^{\mathbf{2}\boldsymbol{\beta}}+\mathbf{2}\boldsymbol{\mu}\boldsymbol{e}^{\boldsymbol{\beta}}+\mathbf{1}\right), \end{array} \right\}, \boldsymbol{x}(\boldsymbol{t}) \geq \mathbf{1/2} \end{cases} \tag{9}$$

where ${}^{C}\boldsymbol{\Delta}_{a}^{\nu}$ (${}_{b}^{C}\boldsymbol{\Delta}^{\nu}$) is the $\boldsymbol{\nu}-$th Caputo left **(right)** fractional difference operator, $\boldsymbol{\nu}$ is the fractional order, and $\boldsymbol{a}$ ($\boldsymbol{b}$) is the starting **(ending)** point. The explicit numerical formula of Eq. (9), which we denote by FQTM, is expressed as

$$\begin{cases} \left\{ \begin{array}{l} \boldsymbol{x_n} = \boldsymbol{x_0} + \frac{\mathbf{1}}{\boldsymbol{\Gamma}(\boldsymbol{\nu})}\sum_{\boldsymbol{j}=\boldsymbol{n}}^{-\mathbf{1}} \dfrac{\boldsymbol{\Gamma}(\boldsymbol{j}-\boldsymbol{n}+\boldsymbol{\nu})}{\boldsymbol{\Gamma}(\boldsymbol{j}-\boldsymbol{n}+\mathbf{1})}\left(\boldsymbol{x_{j+1}}(\boldsymbol{\mu}-\mathbf{1})\right) \\ \boldsymbol{y_n} = \boldsymbol{y_0} + \frac{\mathbf{1}}{\boldsymbol{\Gamma}(\boldsymbol{\nu})}\sum_{\boldsymbol{j}=\boldsymbol{n}}^{-\mathbf{1}} \dfrac{\boldsymbol{\Gamma}(\boldsymbol{j}-\boldsymbol{n}+\boldsymbol{\nu})}{\boldsymbol{\Gamma}(\boldsymbol{j}-\boldsymbol{n}+\mathbf{1})}\left(\boldsymbol{y_{j+1}}(\boldsymbol{\mu}\left(\boldsymbol{e}^{-\boldsymbol{\beta}}+\boldsymbol{e}^{\boldsymbol{\beta}}\right)-\mathbf{2})\right) \\ \boldsymbol{z_n} = \boldsymbol{z_0} + \frac{\mathbf{1}}{\boldsymbol{\Gamma}(\boldsymbol{\nu})}\sum_{\boldsymbol{j}=\boldsymbol{n}}^{-\mathbf{1}} \dfrac{\boldsymbol{\Gamma}(\boldsymbol{j}-\boldsymbol{n}+\boldsymbol{\nu})}{\boldsymbol{\Gamma}(\boldsymbol{j}-\boldsymbol{n}+\mathbf{1})}\left(\boldsymbol{z_{j+1}}\left(\mathbf{2}\boldsymbol{\mu}\boldsymbol{e}^{\boldsymbol{\beta}}-\boldsymbol{e}^{\mathbf{2}\boldsymbol{\beta}}-\mathbf{1}\right)\right). \end{array} \right\}, \boldsymbol{x_n} < \mathbf{1/2} \\ \left\{ \begin{array}{l} \boldsymbol{x_n} = \boldsymbol{x_0} + \frac{\mathbf{1}}{\boldsymbol{\Gamma}(\boldsymbol{\nu})}\sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{n}} \dfrac{\boldsymbol{\Gamma}(\boldsymbol{n}-\boldsymbol{j}+\boldsymbol{\nu})}{\boldsymbol{\Gamma}(\boldsymbol{n}-\boldsymbol{j}+\mathbf{1})}\left(-\boldsymbol{x_{j-1}}(\mathbf{1}+\boldsymbol{\mu})+\boldsymbol{\mu}\right) \\ \boldsymbol{y_n} = \boldsymbol{y_0} + \frac{\mathbf{1}}{\boldsymbol{\Gamma}(\boldsymbol{\nu})}\sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{n}} \dfrac{\boldsymbol{\Gamma}(\boldsymbol{n}-\boldsymbol{j}+\boldsymbol{\nu})}{\boldsymbol{\Gamma}(\boldsymbol{n}-\boldsymbol{j}+\mathbf{1})}\left(-\boldsymbol{y_{j-1}}(\mathbf{2}+\boldsymbol{\mu}\left(\boldsymbol{e}^{-\boldsymbol{\beta}}+\boldsymbol{e}^{\boldsymbol{\beta}}\right))\right) \\ \boldsymbol{z_n} = \boldsymbol{z_0} + \frac{\mathbf{1}}{\boldsymbol{\Gamma}(\boldsymbol{\nu})}\sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{n}} \dfrac{\boldsymbol{\Gamma}(\boldsymbol{n}-\boldsymbol{j}+\boldsymbol{\nu})}{\boldsymbol{\Gamma}(\boldsymbol{n}-\boldsymbol{j}+\mathbf{1})}\left(-\boldsymbol{z_{j-1}}\left(\boldsymbol{e}^{\mathbf{2}\boldsymbol{\beta}}+\mathbf{2}\boldsymbol{\mu}\boldsymbol{e}^{\boldsymbol{\beta}}+\mathbf{1}\right)\right). \end{array} \right\}, \boldsymbol{x_n} \geq \mathbf{1/2} \end{cases} \tag{10}$$

To find the relation between Eq. (9) and Eq. (10), and other details, see Appendix B. The bifurcation diagram of the FQTM is given in Fig. 4 by setting the control parameters $\boldsymbol{m} = \boldsymbol{\mu} = \mathbf{2}$, $\boldsymbol{b} = \boldsymbol{\beta} = \mathbf{0.5}$, and the fractional order $\boldsymbol{\nu} = \mathbf{1.2}$. Fig. 4.1 is plotted for the FQTM for $\boldsymbol{X}$ with the initial value $\boldsymbol{x_0} = \mathbf{0.3456}$, $\boldsymbol{y_0} = \mathbf{0.4865}$, $\boldsymbol{z_0} = \mathbf{0.6703}$, with respect to $\boldsymbol{m} = \boldsymbol{\mu}$, and Fig. 4.2 is for $\boldsymbol{X}$ with respect to $\boldsymbol{b} = \boldsymbol{\beta}$. Fig. 4.3 is plotted for the FQTM for $\boldsymbol{Y}$ with the value $\boldsymbol{x_0} = \mathbf{0.3456}$, $\boldsymbol{y_0} = \mathbf{0.4865}$, $\boldsymbol{z_0} = \mathbf{0.6703}$, with respect to $\boldsymbol{b} = \boldsymbol{\beta}$, and Fig. 4.4 is for $\boldsymbol{Y}$ with respect to $\boldsymbol{m} = \boldsymbol{\mu}$. Fig. 4.5 is plotted for the FQTM for $\boldsymbol{Z}$ with the initial value $\boldsymbol{x_0} = \mathbf{0.3456}$, $\boldsymbol{y_0} = \mathbf{0.4865}$, $\boldsymbol{z_0} = \mathbf{0.6703}$, with respect to $\boldsymbol{b} = \boldsymbol{\beta}$, and Fig. 4.6 is for $\boldsymbol{Z}$ with respect to $\boldsymbol{m} = \boldsymbol{\mu}$. Based on these

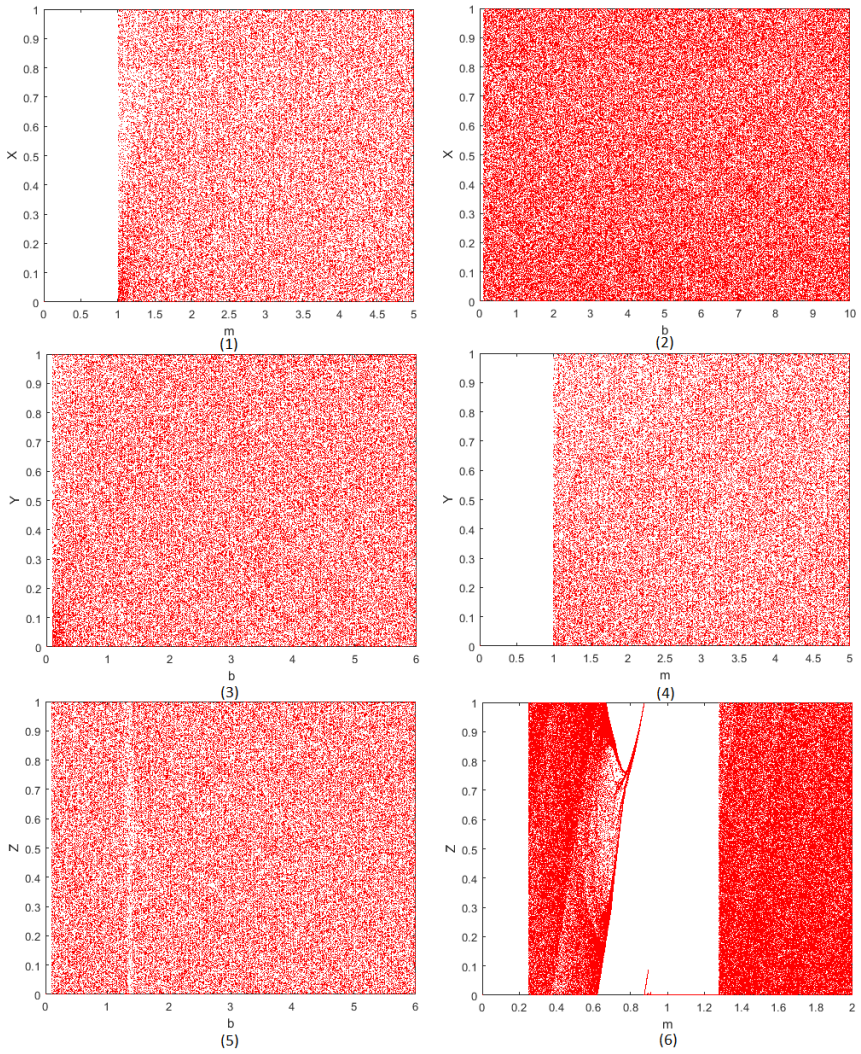


Fig. 4 Bifurcation diagram of the FQTM with $\boldsymbol{m = \mu = 2}$, $\boldsymbol{b = \beta = 0.5}$, $\boldsymbol{x_0 = 0.3456}$, $\boldsymbol{y_0 = 0.4865}$, $\boldsymbol{z_0 = 0.6703}$, and $\boldsymbol{\nu = 1.2}$ for (1) $\boldsymbol{X}$ with respect to $\boldsymbol{m = \mu}$, (2) $\boldsymbol{X}$ with respect to $\boldsymbol{b = \beta}$, (3) $\boldsymbol{Y}$ with respect to $\boldsymbol{b = \beta}$, (4) $\boldsymbol{Y}$ with respect to $\boldsymbol{m = \mu}$, (5) $\boldsymbol{Z}$ with respect to $\boldsymbol{b = \beta}$, (6) $\boldsymbol{Z}$ with respect to $\boldsymbol{m = \mu}$.

figures, we can see the chaotic area of this chaotic map. For example, $\boldsymbol{m = \mu}$ should be selected between $\mathbf{1}$ and $\mathbf{5}$ when $\boldsymbol{b = \beta = 0.5}$. The time series of the FQTM for $\boldsymbol{X}$ is given in Fig. 5 with the control parameters $\boldsymbol{m = \mu = 2}$, $\boldsymbol{b = \beta = 0.5}$, and the fractional order $\boldsymbol{\nu = 1.2}$. Figure 5 shows the map is sensitive to the initial values.

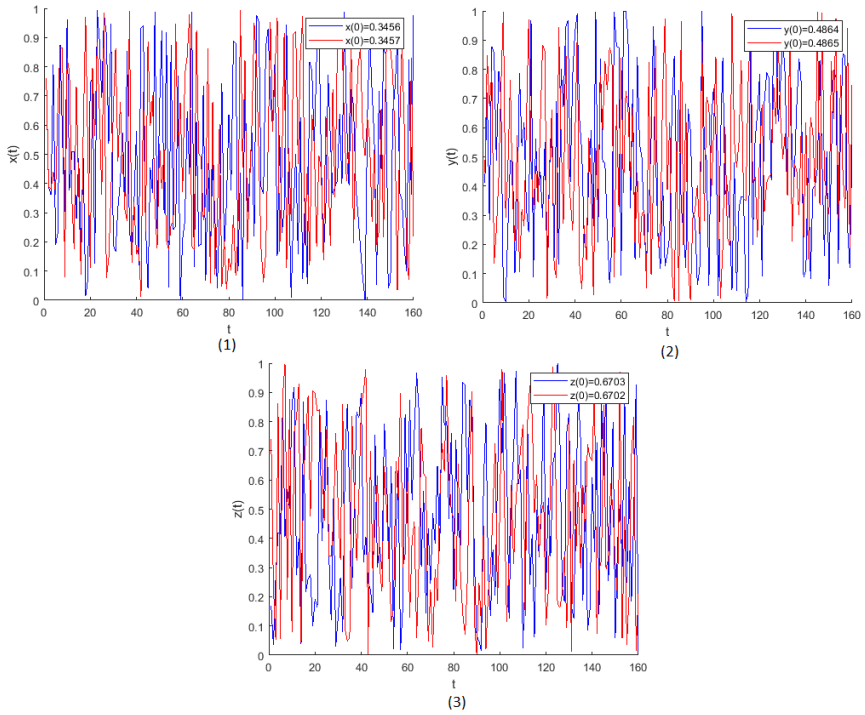


Fig. 5 Time series of the FQTM for $\boldsymbol{X}$ with $\boldsymbol{m = \mu = 2}$, $\boldsymbol{b = \beta = 0.5}$, and $\boldsymbol{\nu = 1.2}$.

4 Improved Neural Network

Chaotic maps can be used to improve the performance of NNs by providing a method for generating random or pseudo-random numbers that can be used for initializing the weights and biases of the NN, as well as for introducing randomness into the training process. This can help to prevent the NN from getting stuck in local minima and improve its overall performance. One way to use chaotic maps in improving the performance of NNs is by using them to generate initial values for the weights and biases of the network. By using chaotic maps to generate these initial values, it is possible to introduce a level of randomness that can help to prevent the network from becoming stuck in a sub-optimal solution.

We have developed new fractional versions of the hierarchical rational chaotic map and quantum editions of the logistic and tent maps. Our goal is to incorporate these maps into Batch Gradient Descent (BGD), Mini-Batch Gradient Descent (MBGD), and Stochastic Gradient Descent (SGD) algorithms. To enhance the training dynamics of NNs, we incorporate fractional and quantum chaotic maps into the batch gradient descent (BGD) algorithm. This integration modifies the weight update rules by embedding chaotic perturbations, which improve convergence behavior and reduce the likelihood of local minima entrapment. The integration of chaotic or quantum maps into NN training modifies weight updates within the gradient descent loop. Given a weight matrix $\boldsymbol{W}$, loss function $\boldsymbol{E}$, learning rate $\boldsymbol{\alpha}$, fractional order parameter $\boldsymbol{\nu}$,

discrete time step $\boldsymbol{t}$, and chaotic/quantum map output $\boldsymbol{f_{map}}$, the standard gradient descent update,

$$\boldsymbol{W \leftarrow W - \alpha \cdot \frac{\partial E}{\partial W}},$$

is augmented as follows:

$$\boldsymbol{W \leftarrow W - \alpha \cdot \frac{\partial E}{\partial W} + f_{map}(t, \nu, ...)}.$$

Here, $\boldsymbol{f_{map}}$ represents the output of a chosen map (e.g., hierarchical rational chaotic, quantum logistic, or quantum tent), potentially of fractional order $\boldsymbol{\nu}$. This output, calculated at each training step, either perturbs the gradient (introducing controlled randomness to escape local minima) or dynamically modulates the learning rate.

The application varies with the gradient descent variant: in BGD, the map output influences the global update once per dataset pass; in MBGD, it's recalculated for each mini-batch, enabling finer-scale variation; and in SGD, it's computed for every sample update, generating high-frequency perturbations.

The fractional order introduces fractional difference operators, altering the map's memory. The current output depends on a weighted sum of past states, injecting correlated variations that can smooth convergence while aiding exploration. Practically, this approach helps escape local minima via chaotic perturbations, accelerates convergence through modulated learning dynamics, and promotes better weight distribution, suggesting improved feature representation.

4.1 Improved neural network training batch gradient descent

In this section, the NN with feed-forward and back-propagation (BP) is used. The BP method, which is thought to provide a solution to the learning problem, uses gradient descent, a mathematical algorithm that minimizes the cost function (error) [44]. The steps of the improved NN training BGD algorithm are as follows:

. Inputs normalization.
. Enter iteration numbers, initial values, and control parameters for the chaotic map.
. Enter the input, hidden, and output layer's size. (We choose **5** hidden layers. The structure of NN is seen in Fig. 6.)
. Enter the learning factor, momentum with chaotic map, and iteration number.
. Add bias to inputs.
. Initialize the weight matrices ($\boldsymbol{W}$) and bias vectors ($\boldsymbol{V}$) using values generated from a chaotic map. This promotes diversity in the initial parameter space.
. Put the weight and bias gradients, $\boldsymbol{\delta W}$ and $\boldsymbol{\delta V}$, respectively, to zero.
. Feed-forward. Compute the output of the NN for the current input.
. Back-propagation. Using back-propagation, compute the gradients of the cost function with respect to the weights and biases.
. Put $\boldsymbol{W = W + \delta W}$ and $\boldsymbol{V = V + \delta V}$.
. If $\boldsymbol{i < 100}$ goes to step feed forward.

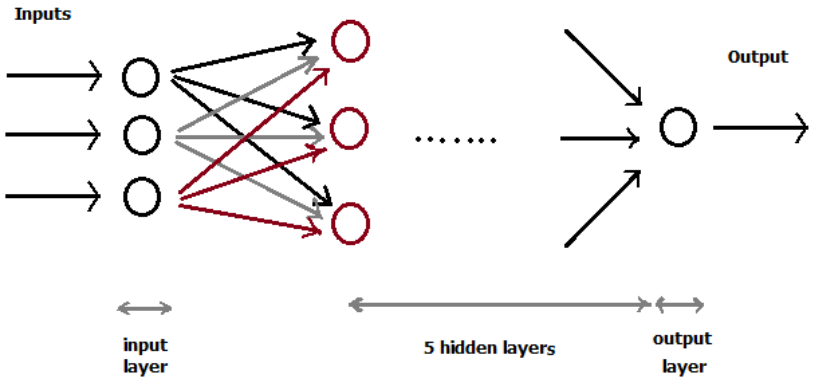


Fig. 6 The structure of NN design.

. Apply the logistic sigmoid activation function to the weighted sum of inputs ($\mathbf{X}$) and weights ($\boldsymbol{W}$) in each hidden layer: Layer output$= \mathbf{1}/(\mathbf{1} + e^{\mathbf{X}.\boldsymbol{W}^{T}})$.
. Add one to layer output.
. Multiply $\boldsymbol{V}$.
. Calculate the Mean Squared Error (MSE) between the predicted output and the actual target values.

The flowchart for the improved NN design, the improved BGD, is shown in Fig. 7.

4.2 Improved neural network training stochastic gradient descent

A cost function is explicitly minimized by several NN learning algorithms. For example, the back-propagation technique employs a gradient descent algorithm to minimize the MSE criterion. Because the cost function is an average of all training cases, computing its gradient necessitates a loop over all of them. The weights are then modified once after each sweep over the training set in the complete gradient descent process. Proper learning rates ensure that this algorithm converges to the cost function's local minimum. However, SGD has been proved to be faster, more reliable, and less prone to reaching undesirable local minima than traditional gradient descent. The weights in this approach are modified after each example is shown, based on the gradient of the loss function, i.e. the cost value for this example only ([45]). The steps of the NN training SGD algorithm are as follows:

. Inputs normalization.
. Enter the input, hidden, and output layer's size. (We choose **5** hidden layers. The structure of NN is seen in Fig. 6.)

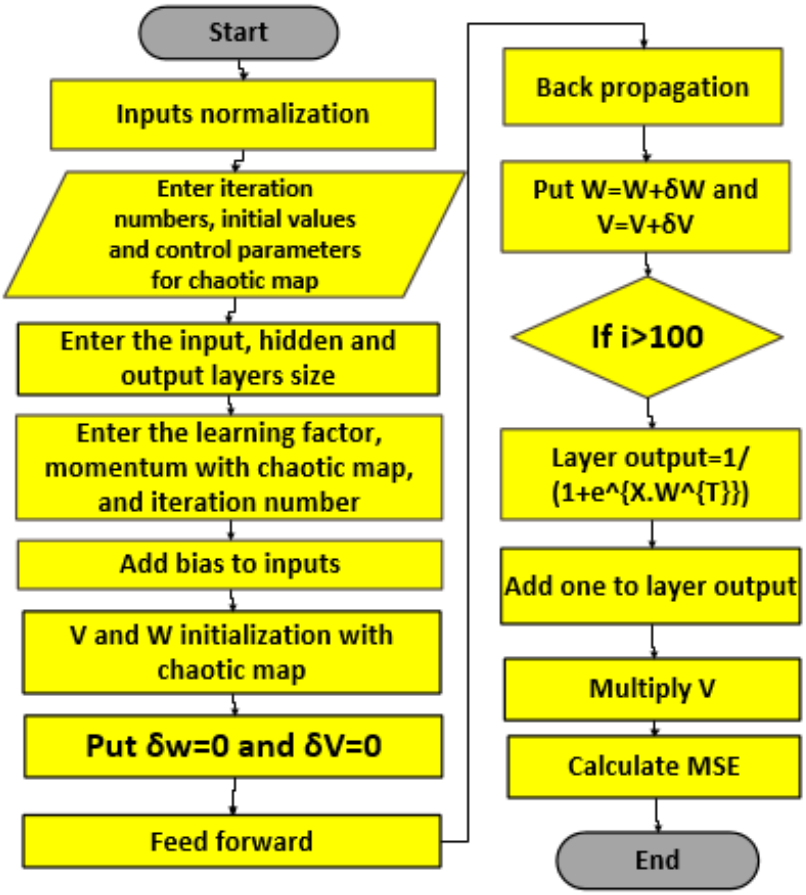


Fig. 7 The flowchart for the improved NN design.

. Enter the learning factor, momentum with chaotic map, and iteration number.
. Add bias to inputs.
. Initialize the weight matrices ($\boldsymbol{W}$) and bias vectors ($\boldsymbol{V}$) using values generated from a chaotic map. This promotes diversity in the initial parameter space.
. For each training example in the dataset, do the following steps.
. Put the weight and bias gradients ($\boldsymbol{\delta W}$ and $\boldsymbol{\delta V}$) to zero.
. Feed-forward. Compute the output of the NN for the current input.
. Back-propagation. Using back-propagation, compute the gradients of the cost function with respect to the weights and biases.
. Put $\boldsymbol{W} = \boldsymbol{W} + \boldsymbol{\delta W}$ and $\boldsymbol{V} = \boldsymbol{V} + \boldsymbol{\delta V}$. Update the weights and biases using the computed gradients and the chosen learning rate.

$$\text{new parameter} = \text{old parameter} - \text{learning rate} * \text{gradient}$$

. If $\boldsymbol{i < 100}$ goes another training example in the dataset.

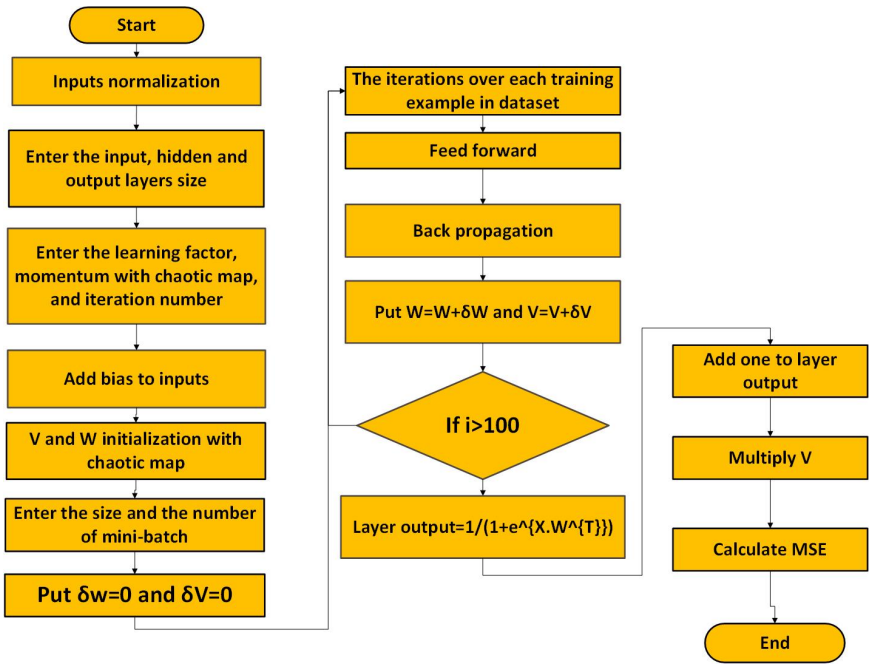


Fig. 8 The flowchart for improved SGD.

. Apply the logistic sigmoid activation function to the weighted sum of inputs ($\mathbf{X}$) and weights ($\boldsymbol{W}$) in each hidden layer: Layer output$= \mathbf{1}/(\mathbf{1} + \boldsymbol{e}^{\mathbf{X}.\boldsymbol{W}^{\boldsymbol{T}}})$.
. Add one to layer output.
. Multiply $\boldsymbol{V}$.
. Calculate the MSE between the predicted output and the actual target values.

The flowchart for improved SGD is shown in Fig. 8. SGD works on a single training sample at a time, in contrast to BGD, which processes the full training dataset in each iteration, and MBGD, which processes a subset (mini-batch) of the data.

4.3 Improved neural network training mini-batch gradient descent

MBGD presents the idea of processing only a subset of the training data in each iteration, rather than the complete data set all at once. This subset is known as a " mini-batch ". The gradient of the cost function with respect to the parameters for this mini-batch is computed by the algorithm, and the parameters are updated accordingly. The method is repeated until convergence or a defined iteration time is reached.

The mini-batch method strikes a balance between the effectiveness of SGD (which uses a batch size of one) and the stability of BGD (which uses the full dataset). It combines the benefits of both methods, providing faster convergence than BGD and less noise in parameter updates than SGD. The steps of the NN training MBGD

algorithm are as follows:

. Inputs normalization.
. Specify the number of neurons in the input, hidden, and output layers. In this study, a five-hidden-layer architecture is employed (see Fig. 6).
. Enter the learning factor, momentum with chaotic map, and iteration number.
. Append a bias term (set to **1**) to the input vector. This allows the network to learn offsets for each feature.
. Initialize the weight matrices ($\boldsymbol{W}$) and bias vectors ($\boldsymbol{V}$) using values generated from a chaotic map. This promotes diversity in the initial parameter space.
. Formation of mini-batch: Split the training dataset into smaller mini-batches to facilitate parallel processing and improve training efficiency. In this work, **7** mini-batches of size **31** are used.
. For each mini-batch, do the following steps.
. Put the weight and bias gradients ($\boldsymbol{\delta W}$ and $\boldsymbol{\delta V}$) to zero.
. Feed-forward. Compute the NN output for the input data in the mini-batch. Calculate the cost function by comparing the projected and actual outputs.
. Back-propagation. Using back-propagation, compute the gradients of the cost function with respect to the weights and biases.
. Put $\boldsymbol{W} = \boldsymbol{W} + \boldsymbol{\delta W}$ and $\boldsymbol{V} = \boldsymbol{V} + \boldsymbol{\delta V}$. Using the obtained gradients and the specified learning rate, update the weights and biases;

$$\text{new parameter} = \text{old parameter} - \text{learning rate} * \text{gradient}$$

. If $\boldsymbol{i} < \mathbf{100}$ goes to step formation of mini-batch.
. Apply the logistic sigmoid activation function to the weighted sum of inputs ($\mathbf{X}$) and weights ($\boldsymbol{W}$) in each hidden layer: Layer output$= \mathbf{1}/(\mathbf{1} + \boldsymbol{e}^{\mathbf{X}.\boldsymbol{W}^{\boldsymbol{T}}})$.
. Add one to layer output.
. Multiply $\boldsymbol{V}$.
. Calculate the MSE between the predicted output and the actual target values.

The flowchart for improved MBGD is shown in Fig. 9.

5 Results

In this section, the results of the performance and convergence of the NNs trained with BGD, MBGD, and SGD, utilizing the classical and quantum chaotic maps and the NNs trained with BGD, MBGD, and SGD, utilizing the fractional versions of the classical and quantum chaotic maps are compared. On the other hand, the results of the performance and convergence these two designed models and the ordinary NNs trained with BGD, MBGD, and SGD are compared. Classical and quantum chaotic maps which are used to improve different NNs are HRCM, QLM, QTM, and their fractional versions are FHRCM, FQLM, and FQTM. Key performance metrics, including training error, learning accuracy, convergence speed, and weight separation, are critical for evaluating model improvements.
We utilize three datasets: House Price [46] (Dataset I), Bike Sharing [47] (Dataset II),

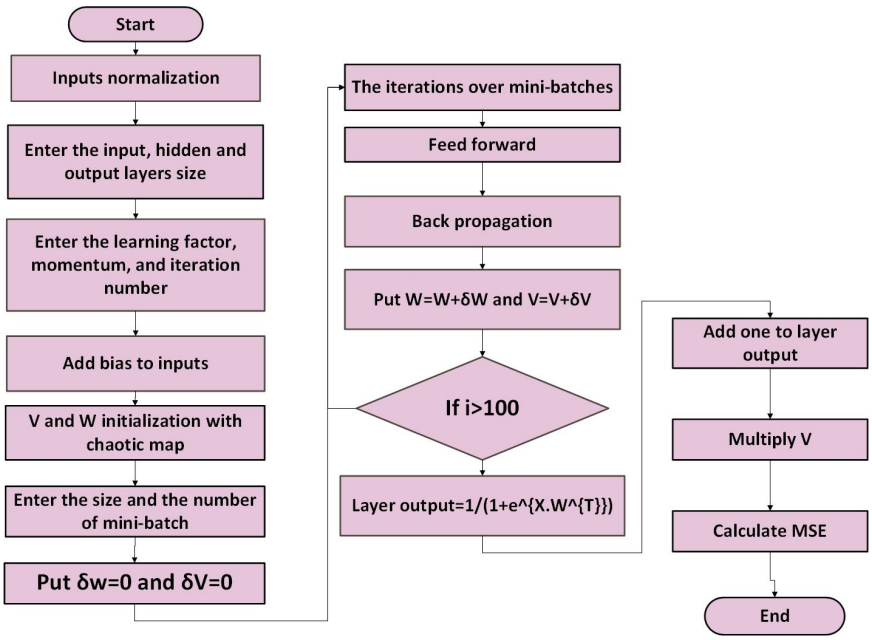


Fig. 9 The flowchart for improved MBGD.

and Concrete Compressive Strength [48] (Dataset III). For house price predicting, two variables are uploaded, one is the inputs of the house, which is a $\mathbf{13 * 506}$ matrix, and the other variable is the house targets, which is a $\mathbf{1 * 506}$ matrix. We have **506** pieces of data related to house prices, and **13** different features are effective in determining house prices and have different values for each of **506** pieces of data. For the concrete compressive strength dataset, the input is an $\mathbf{8 * 1030}$ matrix, and for the bike sharing dataset, an $\mathbf{11 * 731}$ matrix.

First, we used an ordinary NN equipped with BGD to train the data and predict the datasets. For the second step, we applied the mentioned classical and quantum chaotic maps to initialize the velocities and weights, as well as determine the value of the learning coefficient (denoted by $\boldsymbol{\alpha}$) and the momentum coefficient (denoted by $\boldsymbol{\lambda}$) in the algorithm of BGD, and then trained the data. Finally, to compare the effect of the fractionalization of the classical and quantum chaotic maps and the greater complexity of the aforementioned chaotic maps in data training to predict the datasets, we used the fractional version of the classical and quantum chaotic maps in the algorithm of BGD.

Overall, we consider seven cases, one case for the ordinary NNs trained with BGD, MBGD, and SGD, three cases for the NNs trained with BGD, MBGD, and SGD, utilizing the classical and quantum chaotic maps, and three cases for the NNs trained with BGD, MBGD, and SGD, utilizing the fractional versions of the classical and quantum chaotic maps. Figure 10 shows the accuracy of data training with BGD for all seven cases. Figure 11 shows the cost function for all three NN designs trained with BGD, an ordinary NN, the improved NN with the classical and quantum chaotic

maps, and the improved NN with the fractional version of classical and quantum chaotic maps. Different weights for these seven cases are shown in Fig. 12. All these works are done also for other two NNs trained with MBGD and SGD. Representation of regression and dispersion of error for MBGD model and its improved versions are presented in Fig. 13. Cost function for first and second epoch in MBGD model, and its improved versions are shown in Fig. 14. In Fig. 15, we present cost function for **11** different mini-batch in MBGD model and its improved versions. Cost function for MBGD model and its improved versions is seen in Fig. 16. Weights of output layer during training for MBGD model and improved version of it is shown in Fig. 17. Representation of regression and dispersion of error for SGD model and its improved version are presented in Fig. 18. Cost function for first epoch in SGD model and its improved versions are seen in Fig. 19. Cost function for each instance in SGD model and improved version of it is presented in Fig. 20. In Fig. 21, we show Cost function for SGD model and improved version of it. Weights of output layer during training for SGD model and improved version of it is shown in Fig. 22.

Tables 1-3 compare the performance of all twenty-one different NNs trained with BGD, MBGD, and SGD models along with their improved versions based on $\boldsymbol{R}$-squared measure. $\boldsymbol{R}$-squared, denoted here by $\boldsymbol{R}$, is a statistical measure in regression analysis that indicates the proportion of variance in the dependent variable explained by the independent variables. It reflects how well the model's predictions fit the data, with higher values (closer to 1) signifying a better fit. Based on Tables:

. For dataset I, the BGD-trained NN model demonstrates improved accuracy across all classical and quantum chaotic maps, with fractional versions further enhancing performance, particularly when using FQLM. For dataset II, the same model shows improved accuracy with all maps except HRCM, for which the enhanced NN achieves higher accuracy in improved versions. For dataset III, the BGD-trained NN model shows improved accuracy across all classical and quantum chaotic maps mentioned. The enhanced NN model using nonfractional versions of quantum and classical maps is more accurate, except for QTM. In improved versions, the enhanced NN based on HRCM achieves higher accuracy.

. For the MBGD model, FHRCM consistently outperforms classical map-based nearest neighbors across datasets I, II, and III. However, the effect of fractionalization on quantum maps varies by dataset. In datasets I and II, non-fractional quantum maps (QLM and QTM) are more accurate than their fractional counterparts (FQLM and FQTM). In dataset III, while FQLM outperforms the classical map-based NN, the QTM is more accurate than FQTM. Generally, improvements increase accuracy in datasets I and III, except when using FQTM.

. Classical and quantum maps (HRCM, QLM, and QTM) enhance the accuracy of the SGD model across datasets I, II, and III. For dataset I, HRCM outperforms FHRCM, while FQLM and FQTM outperform QLM and QTM, respectively. For datasets II and III, all fractional maps demonstrate higher accuracy than their non-fractional counterparts.

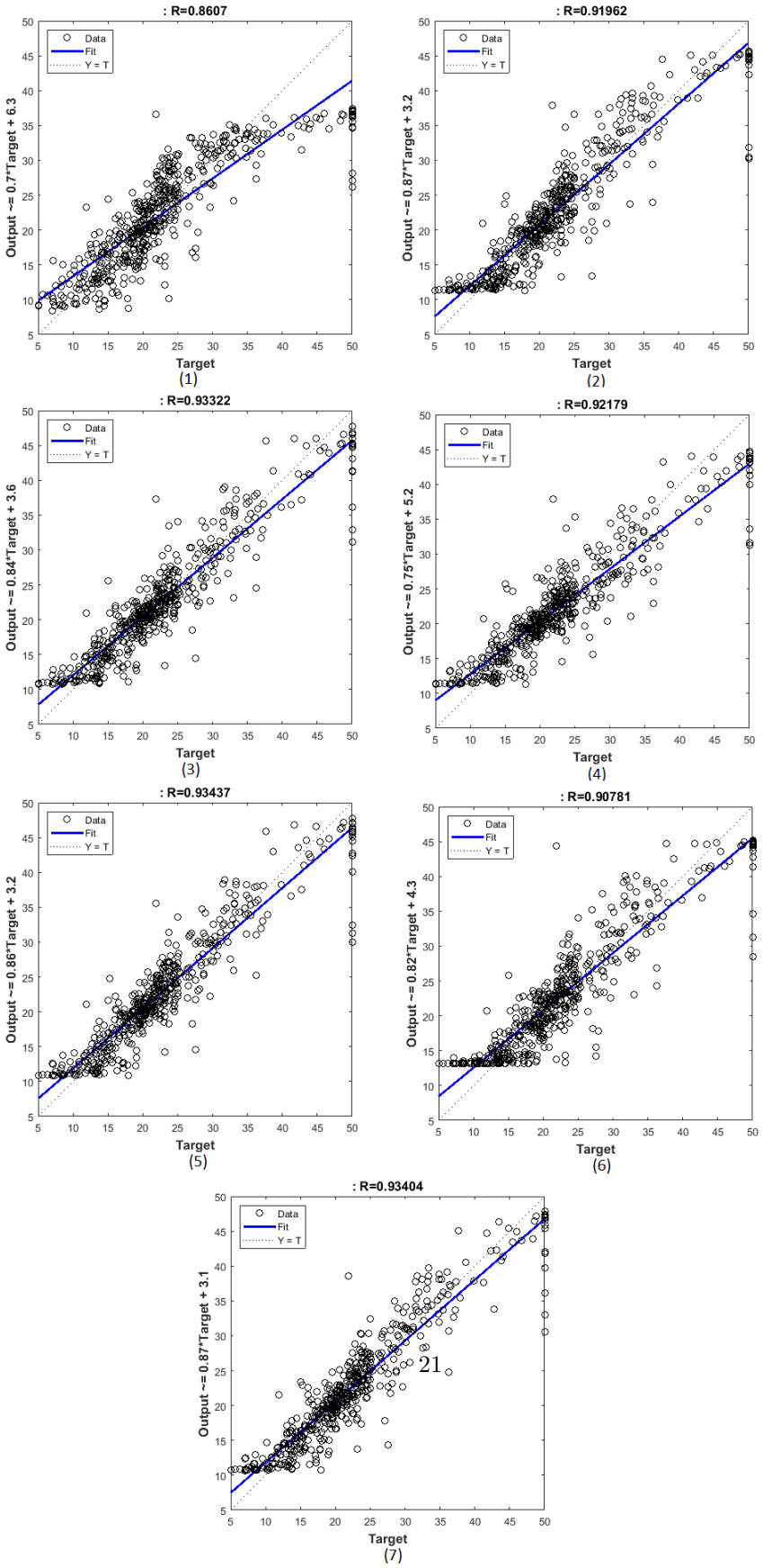


Fig. 10 Representation of regression and dispersion of error for (1) NN model, (2) Improved NN design with HRCM, (3) Improved NN design with FHRCM, (4) Improved NN design with QLM, (5) Improved NN design with FQLM, (6) Improved NN design with QTM, (7) Improved NN design with FQTM.

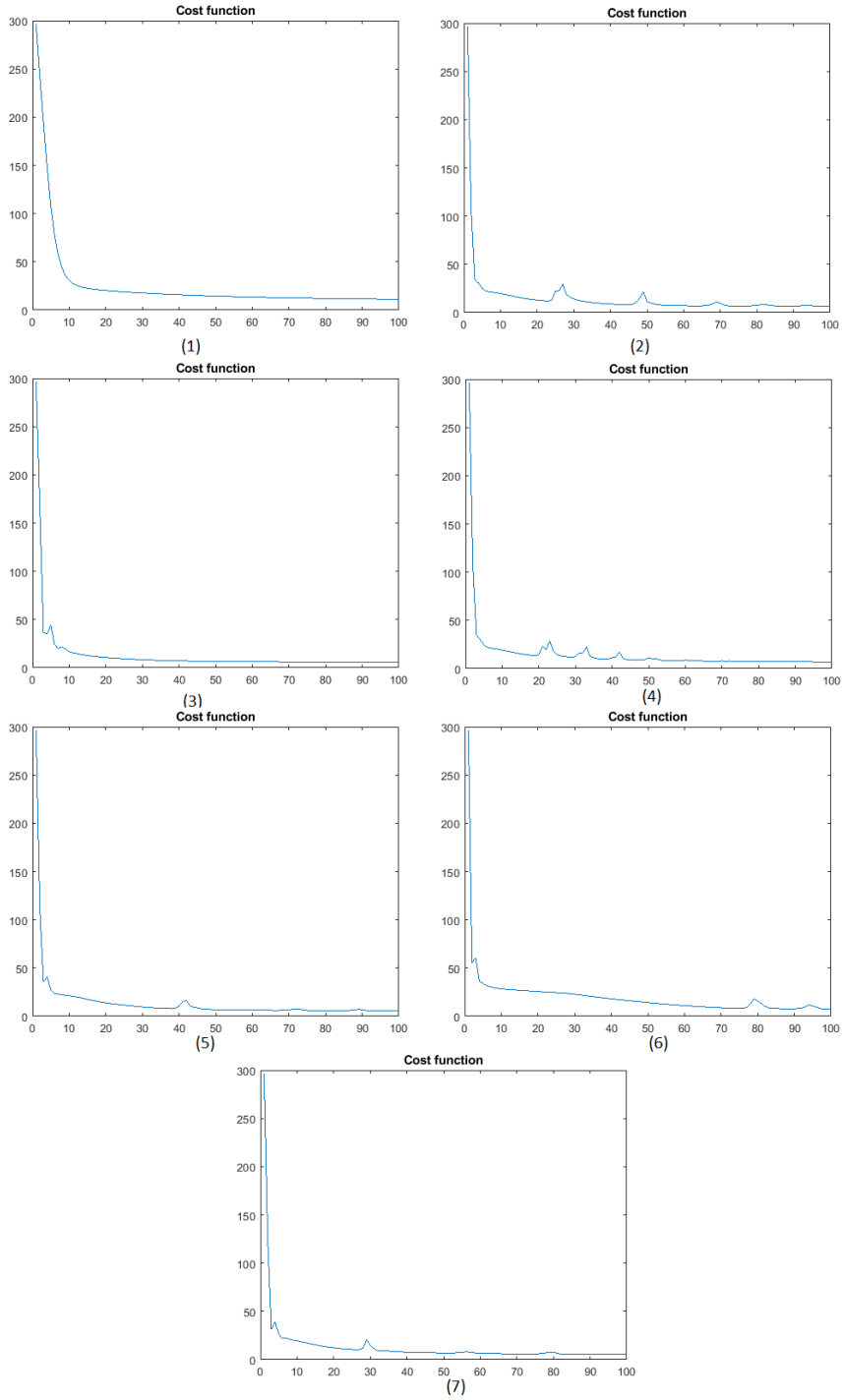


Fig. 11 Cost function for (1) NN model, (2) Improved NN design with HRCM, (3) Improved NN design with FHRCM, (4) Improved NN design with QLM, (5) Improved NN design with FQLM, (6) Improved NN design with QTM, (7) Improved NN design with FQTM.

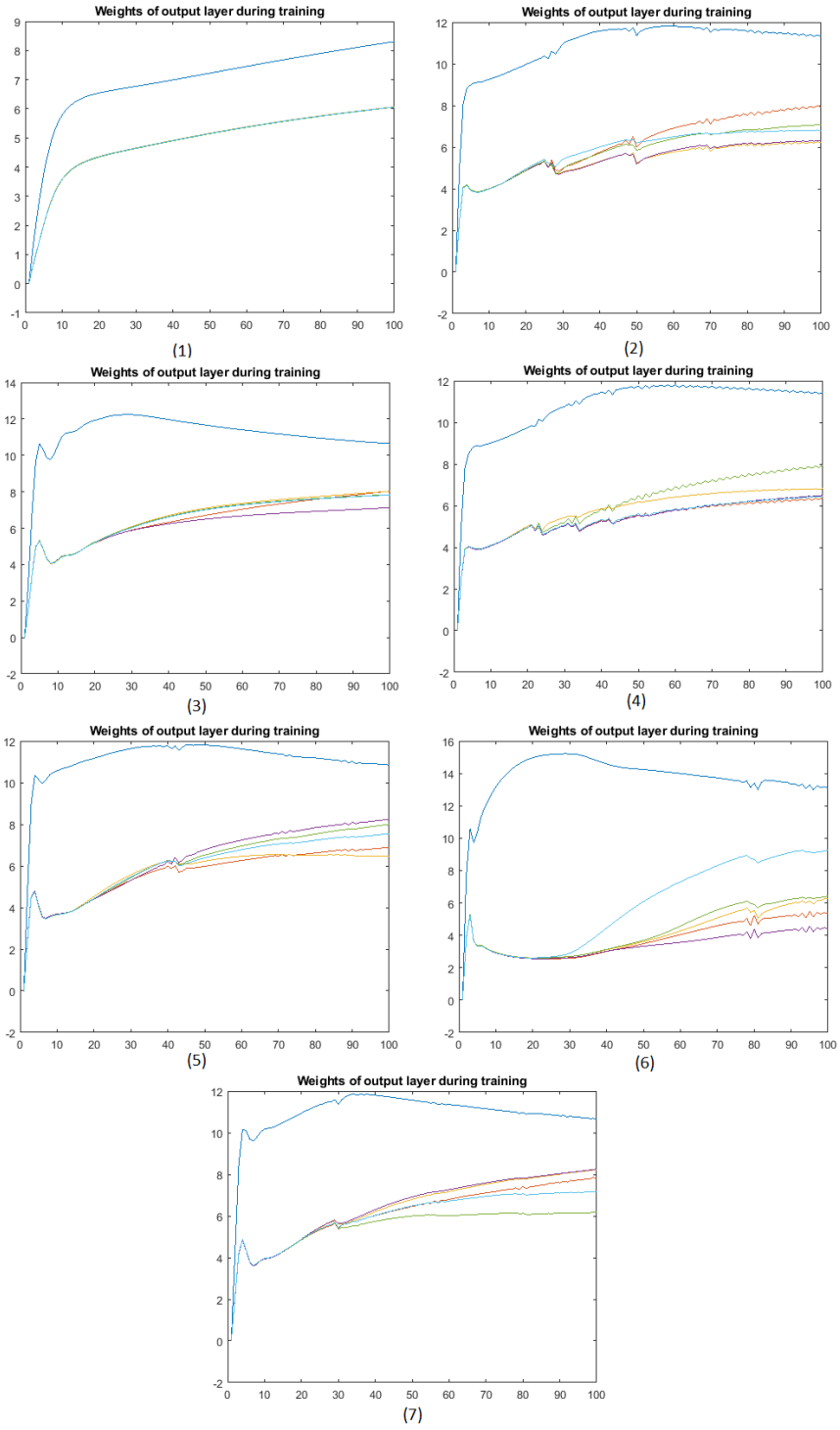


Fig. 12 Weights of output layer during training for (1) NN model, (2) Improved NN design with HRCM, (3) Improved NN design with FHRCM, (4) Improved NN design with QLM, (5) Improved NN design with FQLM, (6) Improved NN design with QTM, (7) Improved NN design with FQTM.

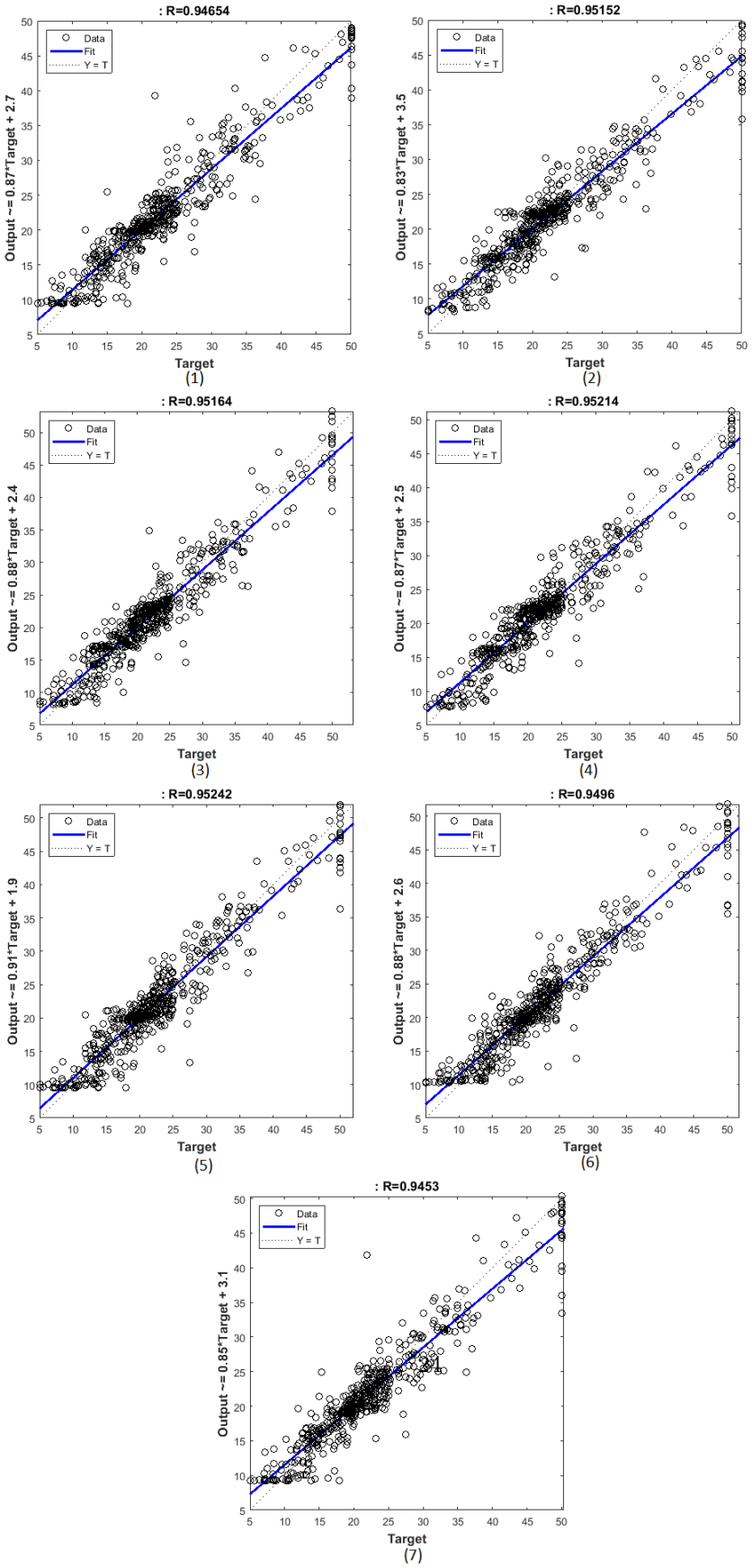


Fig. 13 Representation of regression and dispersion of error for (1) MBGD model, (2) Improved MBGD design with HRCM, (3) Improved MBGD design with FHRCM, (4) Improved MBGD design with QLM, (5) Improved MBGD design with FQLM, (6) Improved MBGD design with QTM, (7) Improved MBGD design with FQTM.

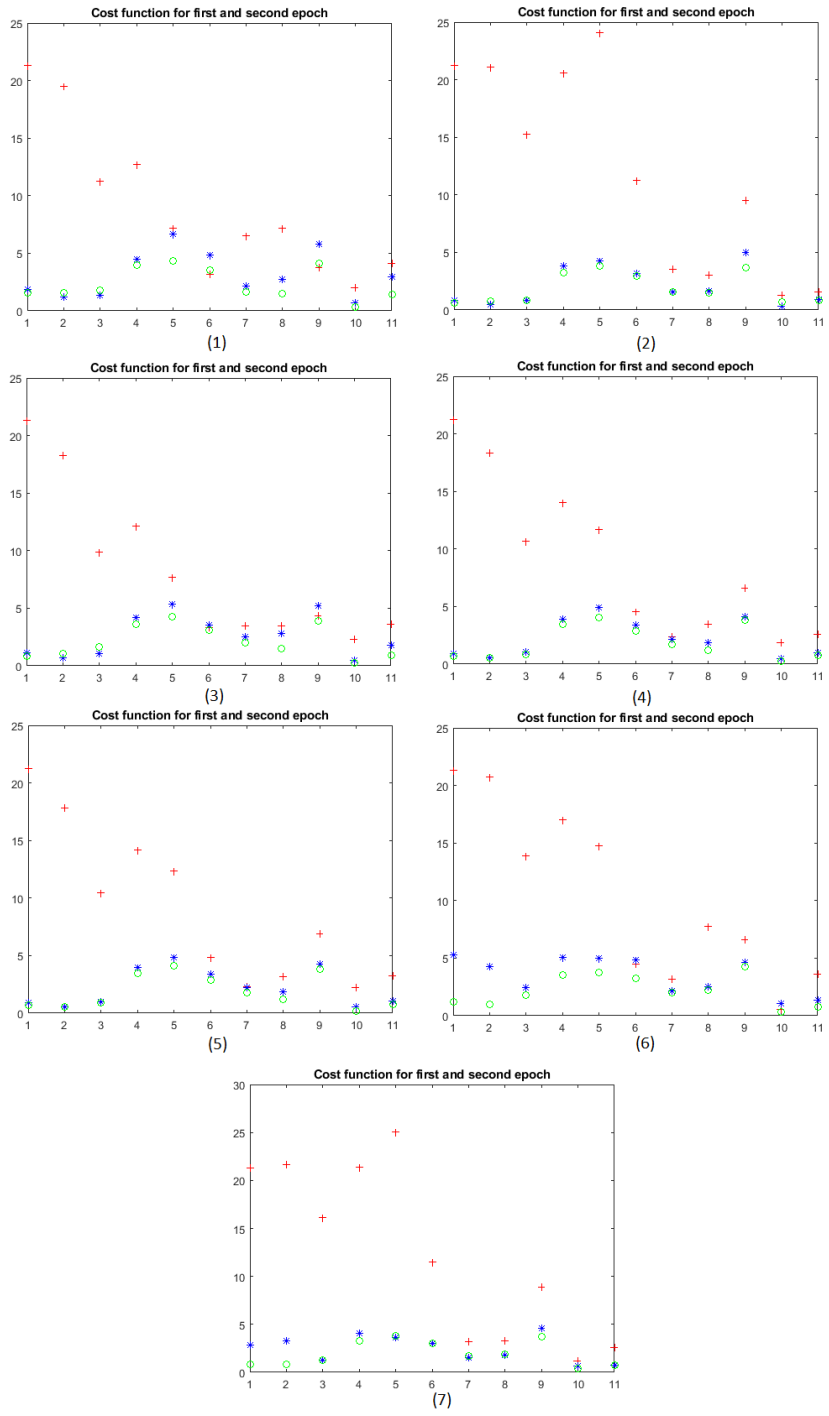


Fig. 14 Cost function for first and second epoch in (1) MBGD model, (2) Improved MBGD design with HRCM, (3) Improved MBGD design with FHRCM, (4) Improved MBGD design with QLM, (5) Improved MBGD design with FQLM, (6) Improved MBGD design with QTM, (7) Improved MBGD design with FQTM.

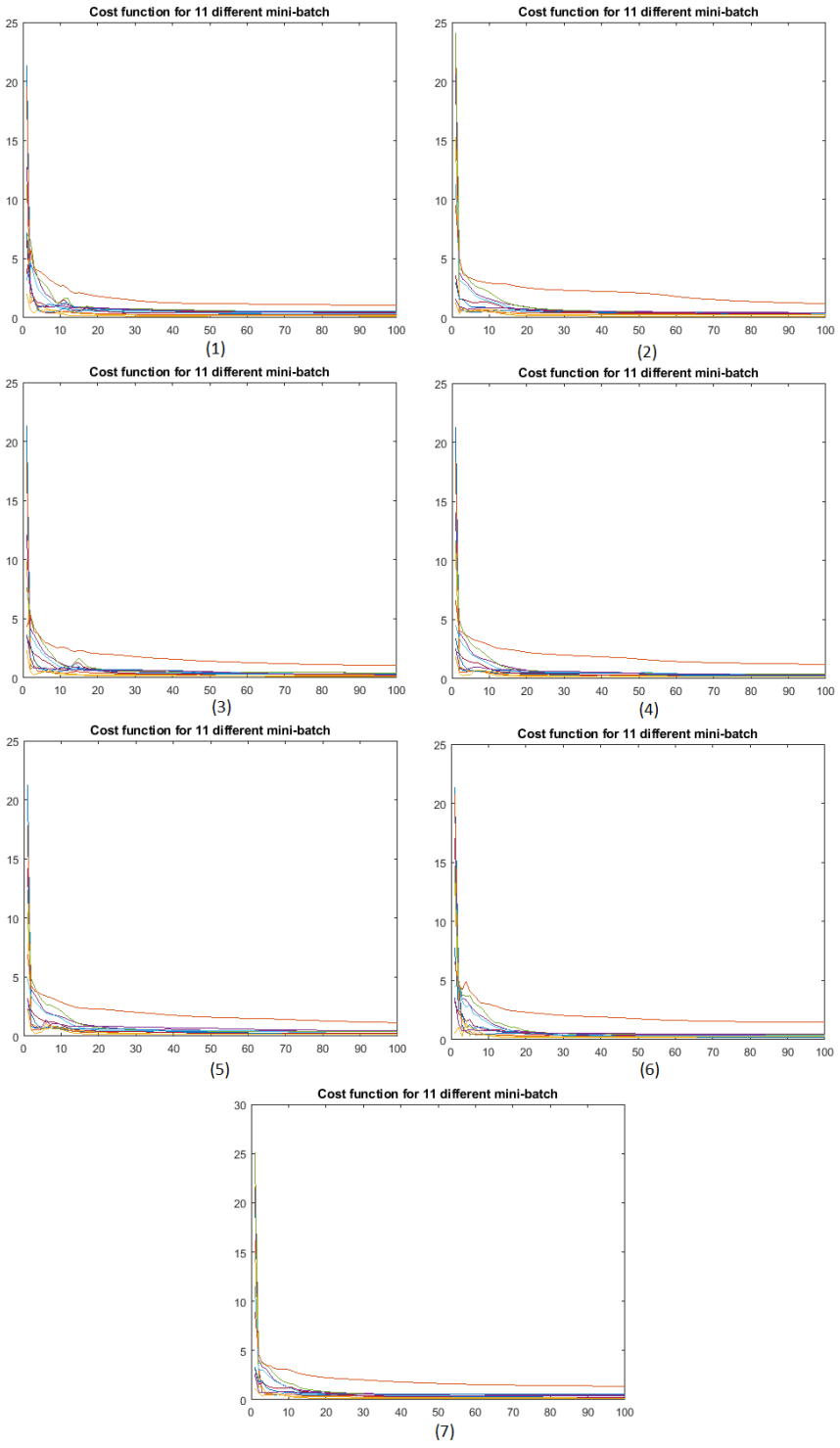


Fig. 15 Cost function for 11 different Mini-batch in (1) MBGD model, (2) Improved MBGD design with HRCM, (3) Improved MBGD design with FHRCM, (4) Improved MBGD design with QLM, (5) Improved MBGD design with FQLM, (6) Improved MBGD design with QTM, (7) Improved MBGD design with FQTM.

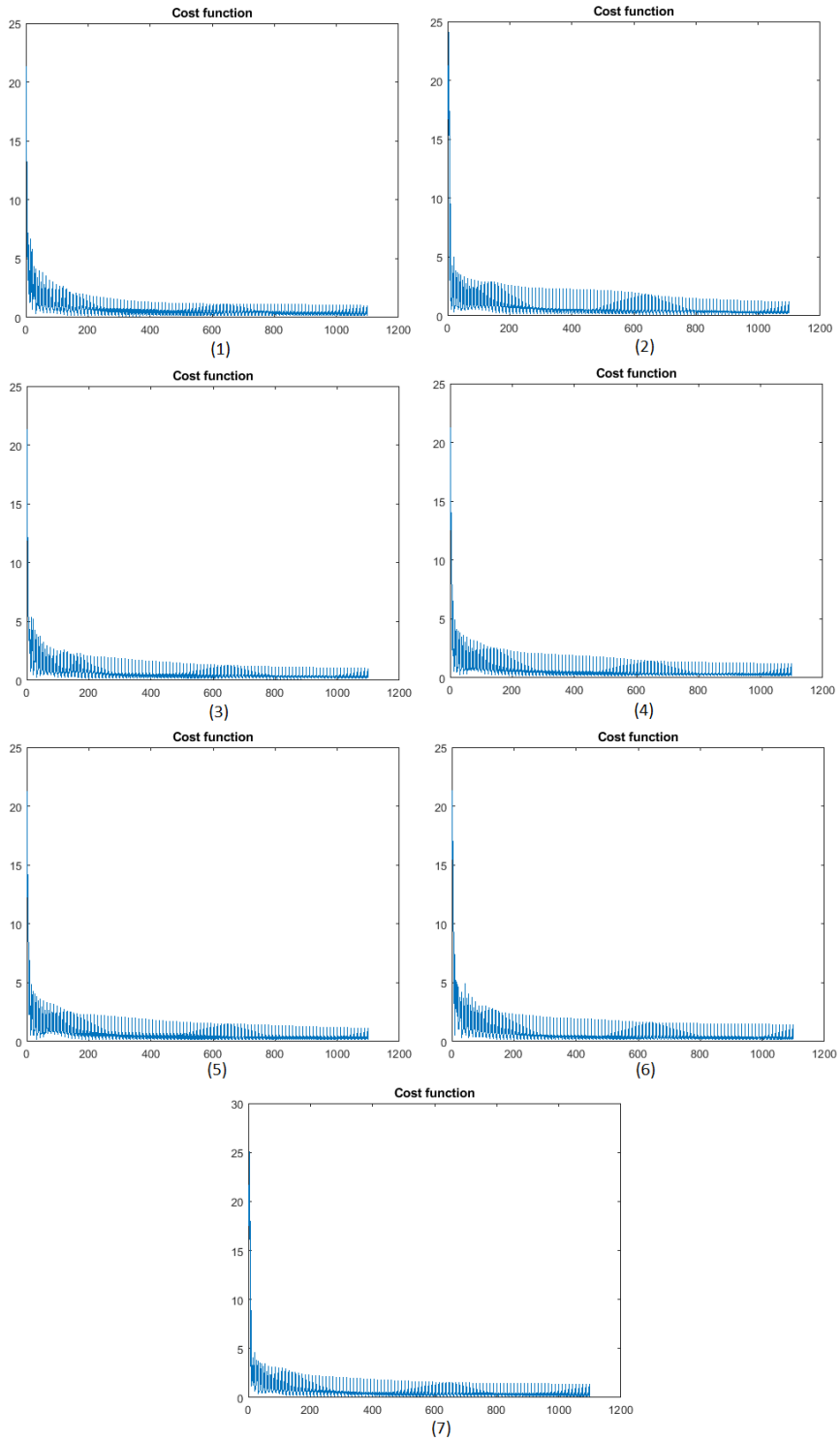


Fig. 16 Cost function for (1) MBGD model, (2) Improved MBGD design with HRCM, (3) Improved MBGD design with FHRCM, (4) Improved MBGD design with QLM, (5) Improved MBGD design with FQLM, (6) Improved MBGD design with QTM, (7) Improved MBGD design with FQTM.

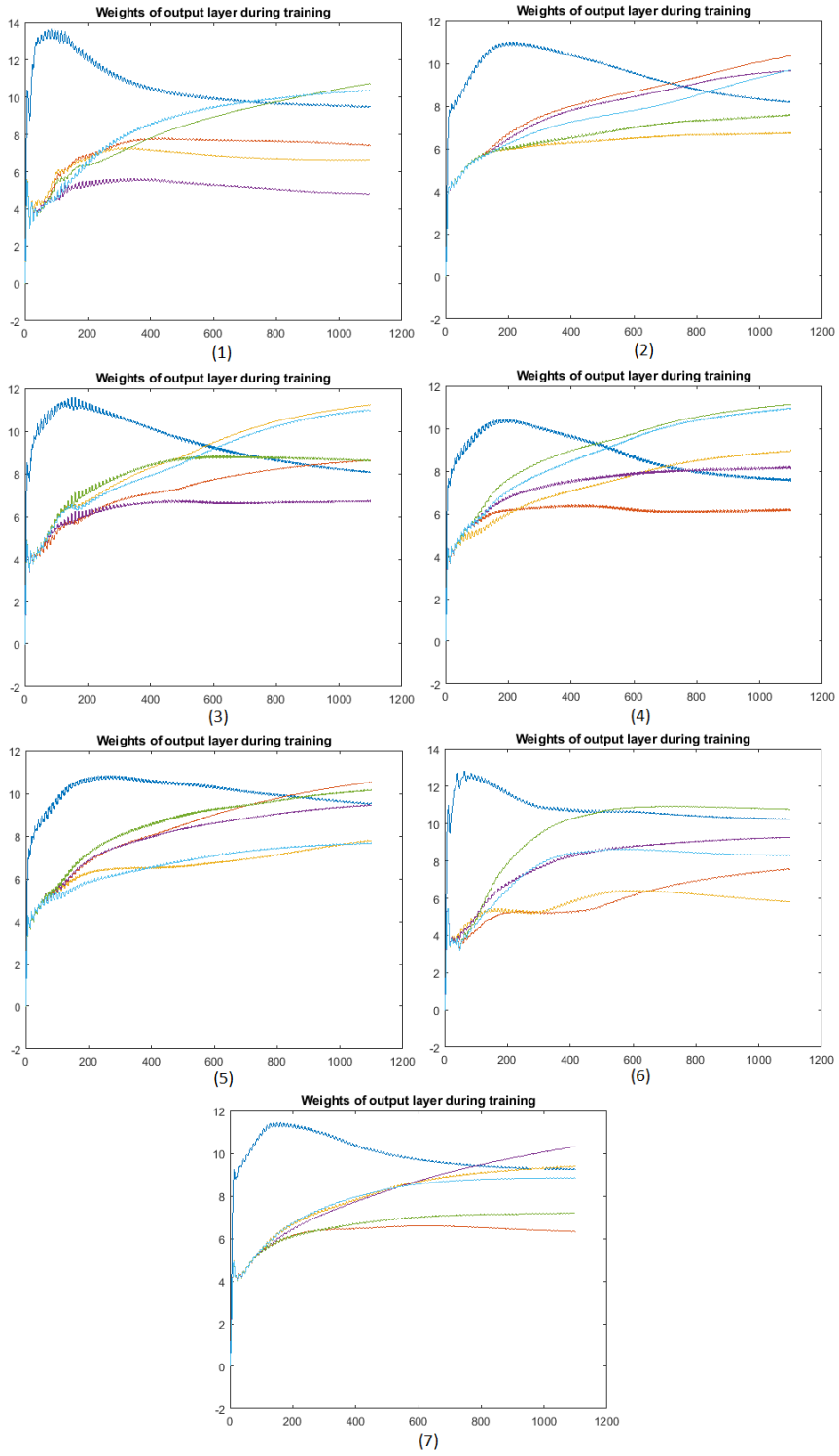


Fig. 17 Weights of output layer during training for (1) MBGD model, (2) Improved BGD design with HRCM, (3) Improved MBGD design with HRCM, (4) Improved MBGD design with QLM, (5) Improved MBGD design with FQLM, (6) Improved MBGD design with QTM, (7) Improved MBGD design with FQTM.

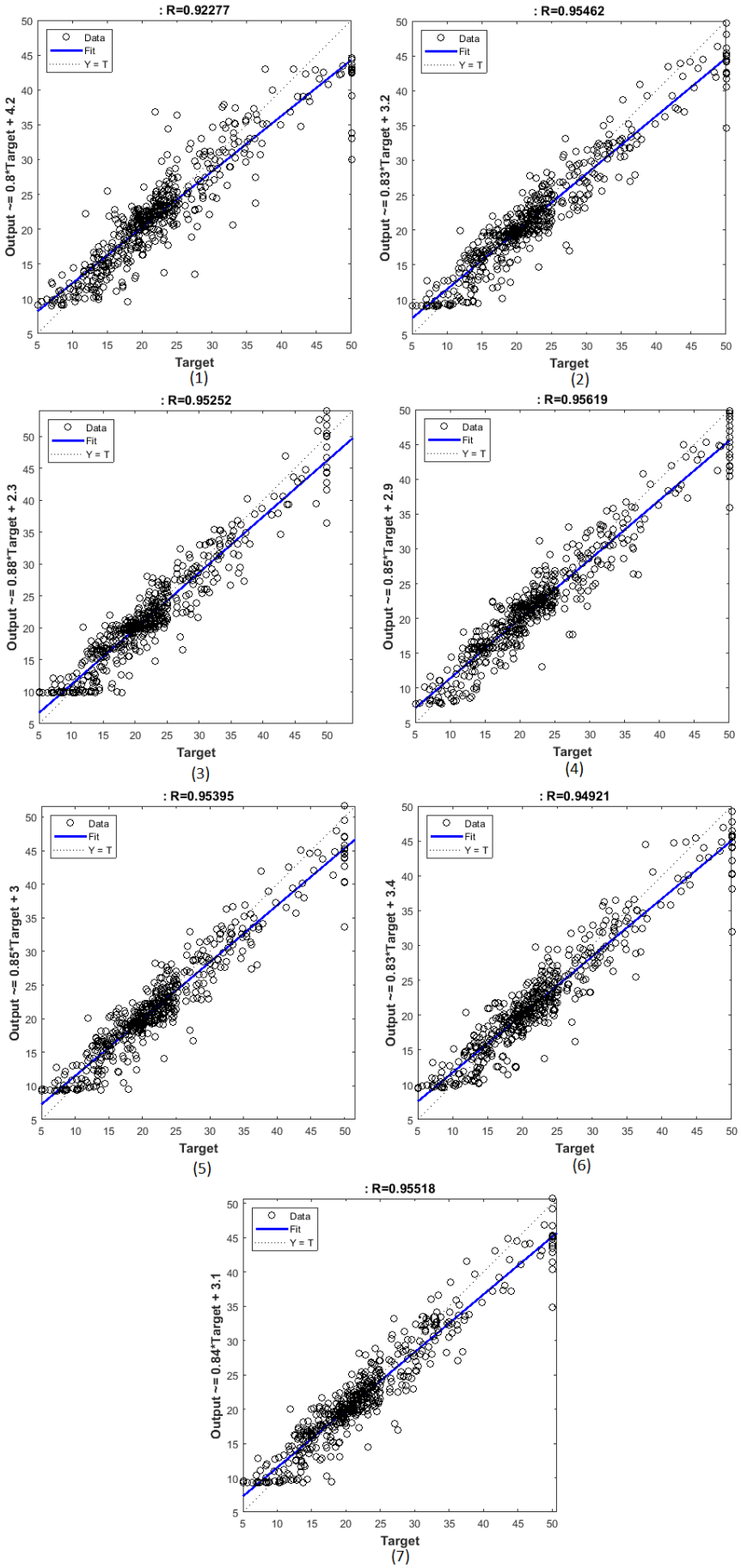


Fig. 18 Representation of regression and dispersion of error for (1) SGD model, (2) Improved SGD design with HRCM, (3) Improved SGD design with FHRCM, (4) Improved SGD design with QLM, (5) Improved SGD design with FQLM , (6) Improved SGD design with QTM, (7) Improved SGD design with FQTM.

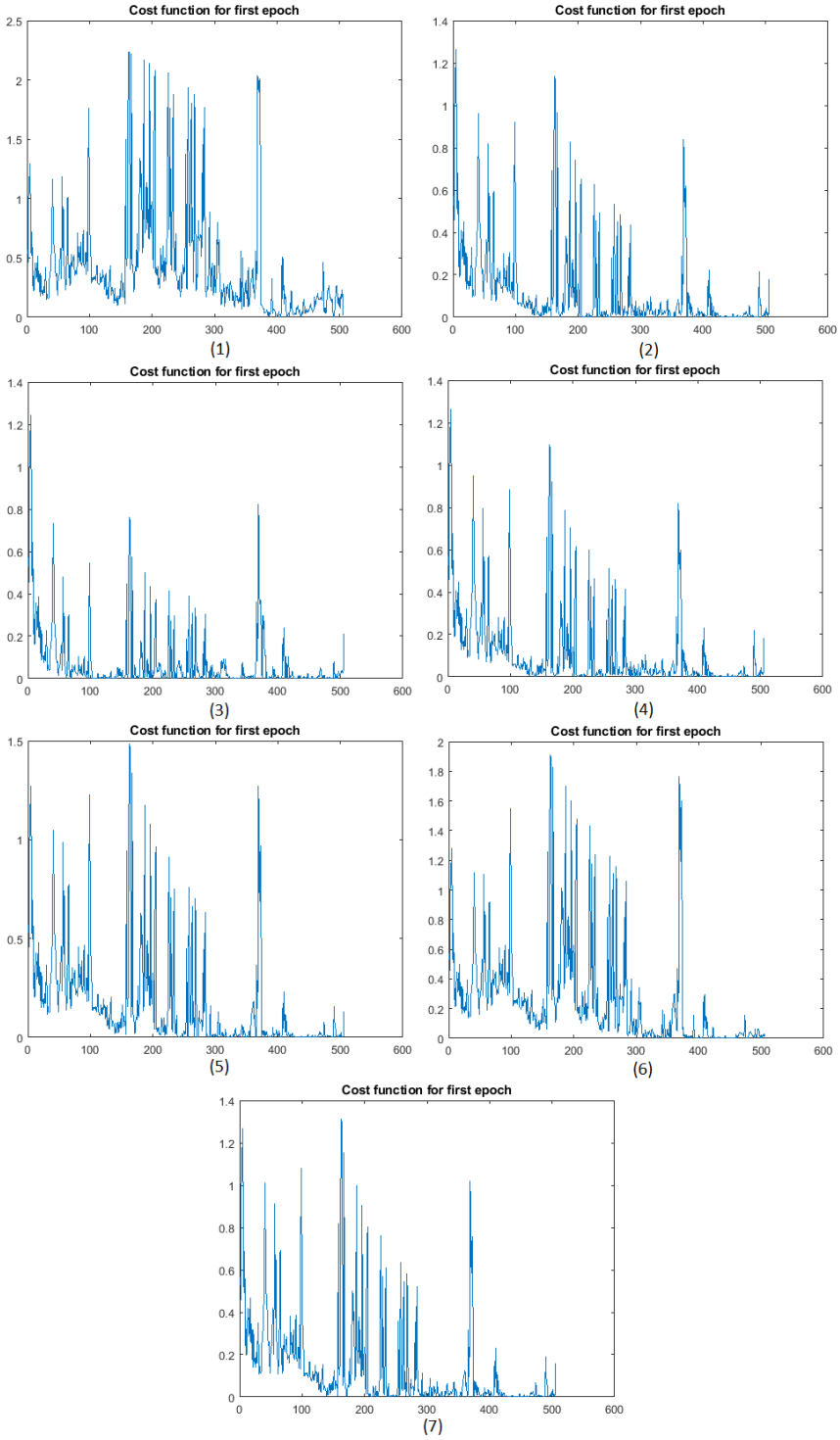


Fig. 19 Cost function for first epoch in (1) SGD model, (2) Improved SGD design with HRCM, (3) Improved SGD design with FHRCM, (4) Improved SGD design with QLM, (5) Improved SGD design with FQLM, (6) Improved SGD design with QTM, (7) Improved SGD design with FQTM.

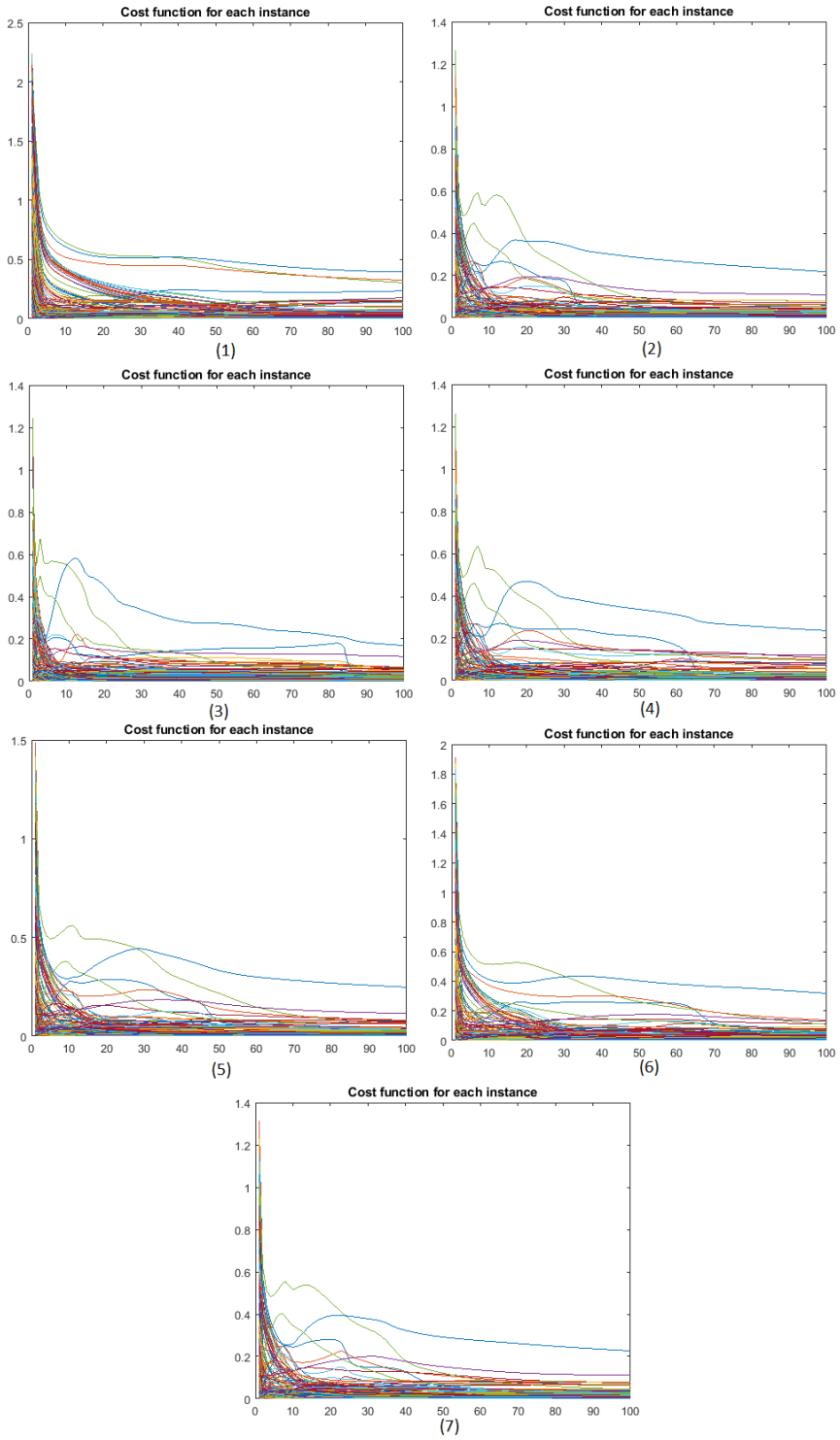


Fig. 20 Cost function for each instance in (1) SGD model, (2) Improved SGD design with HRCM, (3) Improved SGD design with FHRCM, (4) Improved SGD design with QLM, (5) Improved SGD design with FQLM, (6) Improved SGD design with QTM, (7) Improved SGD design with FQTM.

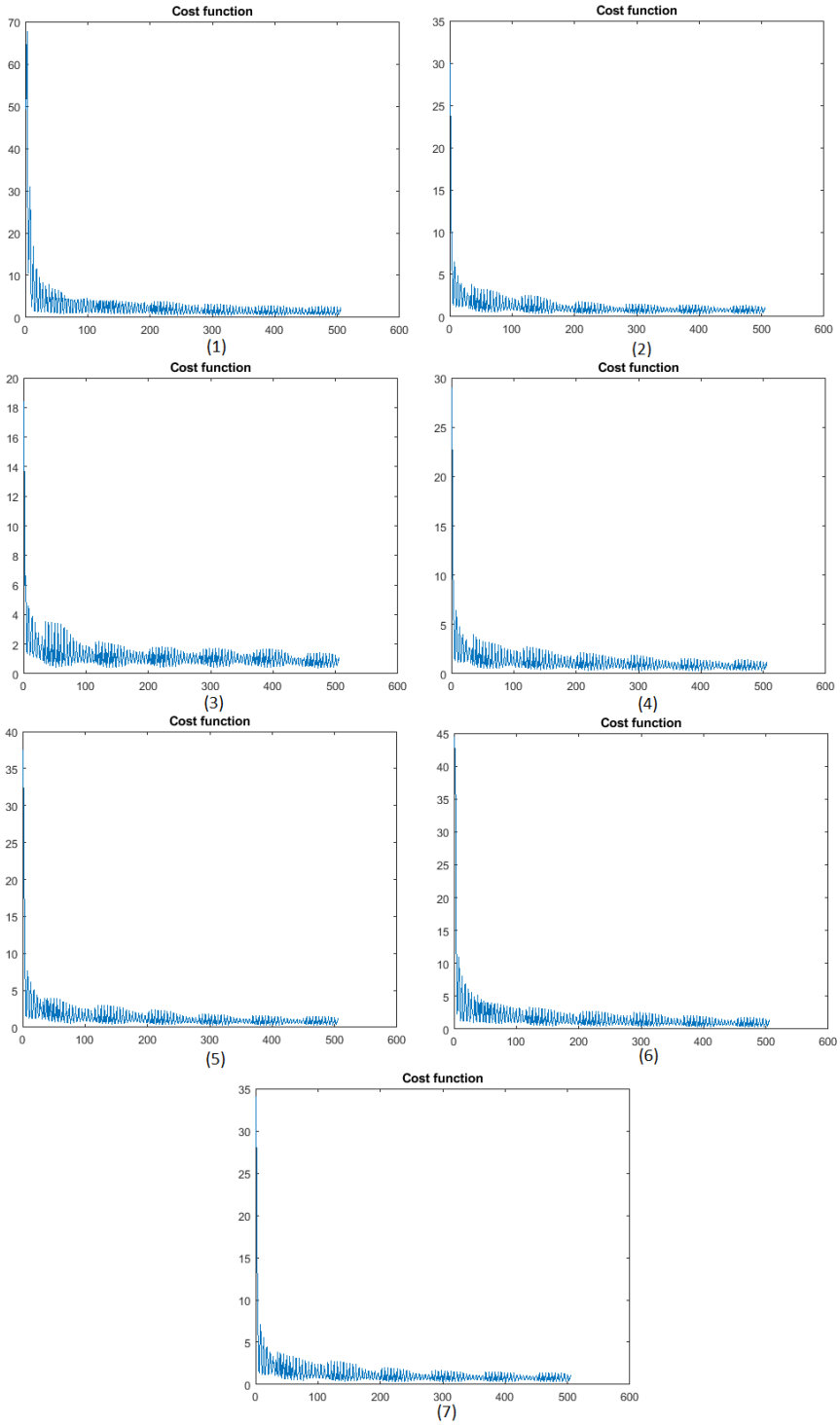


Fig. 21 Cost function for (1) SGD model, (2) Improved SGD design with HRCM, (3) Improved SGD design with FHRCM, (4) Improved SGD design with QLM, (5) Improved SGD design with FQLM, (6) Improved SGD design with QTM, (7) Improved SGD design with FQTM.

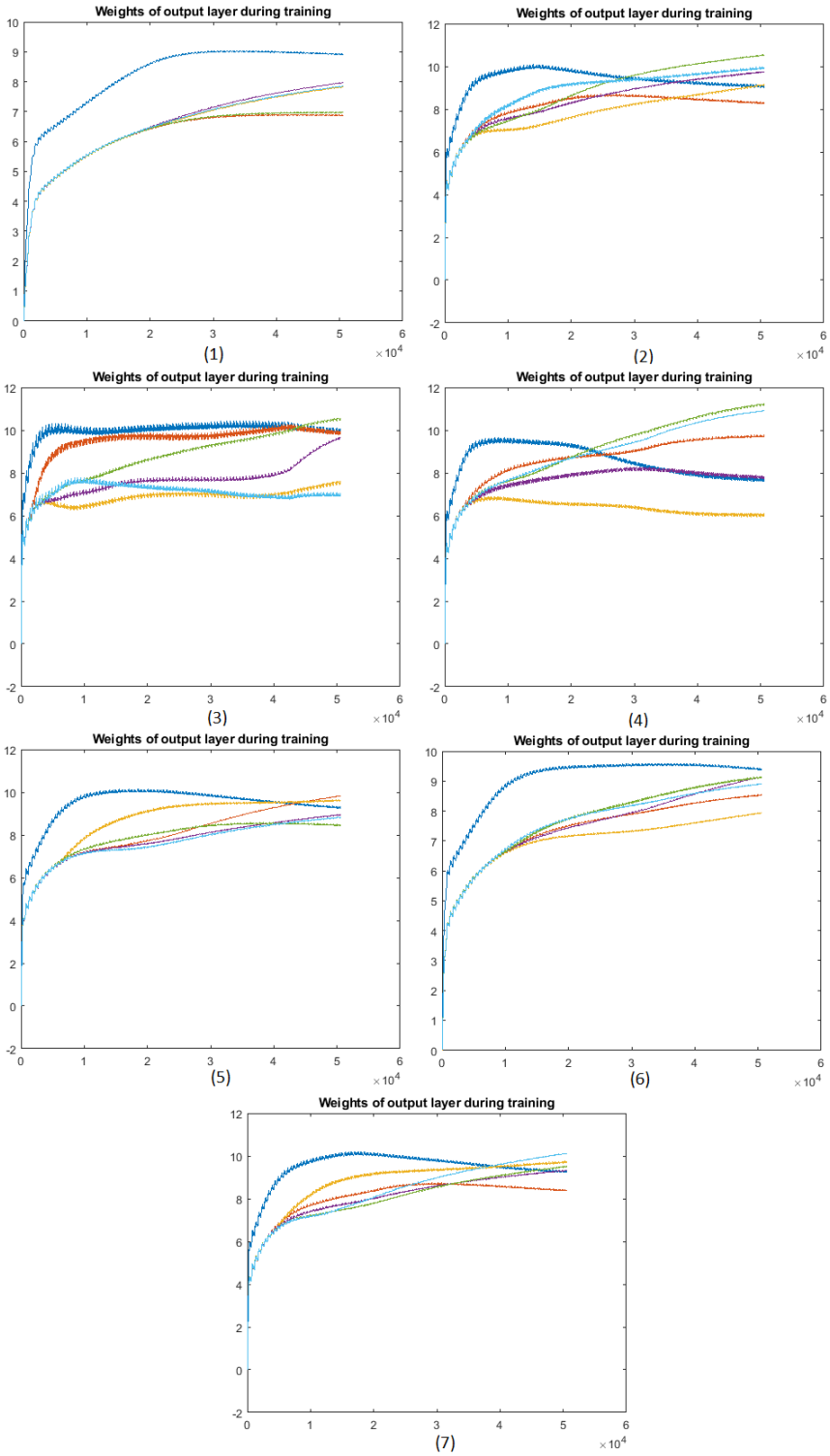


Fig. 22 Weights of output layer during training for (1) SGD model, (2) Improved SGD design with HRCM, (3) Improved SGD design with FHRCM, (4) Improved SGD design with QLM, (5) Improved SGD design with FQLM, (6) Improved SGD design with QTM, (7) Improved SGD design with FQTM.

Table 1 The comparison of ordinary and improved NN model performance for dataset I.

Model	Training with	Designed with	$\boldsymbol{R}$	Train Error	$\boldsymbol{\alpha}$	$\boldsymbol{\lambda}$
NN	BGD		0.8607	4.711887	0.05	0
NN	BGD	HRCM	0.91962	3.636399	0.2216	0.1217
NN	BGD	FHRCM	0.93322	3.313364	0.1341	0.5667
NN	BGD	QLM	0.92179	3.569525	0.1388	0.2521
NN	**BGD**	**FQLM**	**0.93437**	**3.259487**	**0.1050**	**0.7327**
NN	BGD	QTM	0.90781	3.926784	0.3330	0.4007
NN	BGD	FQTM	0.93404	3.309084	0.2617	0.3770
NN	MBGD		0.94654	2.98569	0.05	0.7
NN	MBGD	HRCM	0.95152	2.954052	0.0317	0.4599
NN	MBGD	FHRCM	0.95164	2.844372	0.0647	0.4345
NN	MBGD	QLM	0.95214	2.840288	0.0645	0.2402
NN	**MBGD**	**FQLM**	**0.95242**	**2.977674**	**0.0703**	**0.1090**
NN	MBGD	QTM	0.9496	2.886044	0.0359	0.7227
NN	MBGD	FQTM	0.9453	3.053397	0.0253	0.6582
NN	SGD		0.92277	3.583878	0.05	0.7
NN	SGD	HRCM	0.95462	2.921868	0.7604	0.2652
NN	SGD	FHRCM	0.95252	2.848831	0.9817	0.5335
NN	SGD	QLM	0.95249	2.939511	0.7013	0.3620
NN	SGD	FQLM	0.9527	2.842181	0.4677	0.2971
NN	SGD	QTM	0.94921	2.987391	0.2779	0.2610
NN	**SGD**	**FQTM**	**0.95518**	**2.849262**	**0.6351**	**0.2239**

6 Comparative Statistical Analysis

To compare fractional quantum chaotic maps with their non-fractional counterparts, we organized the models into two groups: Fractional (FQLM, FQTM, FHRCM) and Non-Fractional (QLM, QTM, HRCM). The analysis included the metrics $\boldsymbol{R}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\lambda}$. Synthetic data was generated for each model, assuming a normal distribution with a **5%** standard deviation and **30** samples, based on the values reported in Tables 1-3. Independent two-sample t-tests revealed significant differences between groups for all metrics: $\boldsymbol{R(t = 3.87, p = 0.0003,}$ Cohen's $\boldsymbol{d = 0.88)}$, $\boldsymbol{\lambda(t = 4.45, p < 0.0001,}$ Cohen's d $\boldsymbol{= 1.17)}$, and $\boldsymbol{\alpha(t = 6.21, p < 0.0001,}$ Cohen's d $\boldsymbol{= 1.07)}$. These results suggest that the fractional models outperform the non-fractional ones across all three metrics with statistically significant differences.

Figure 23 shows a rightward shift in fractional group distributions across all metrics. Boxplots indicate higher medians and reduced variability in fractional models, with Cohen's d values confirming large to very large effect sizes. Fractional chaotic maps enhance neural network training by improving predictive accuracy (higher $\boldsymbol{R}$), accelerating convergence (higher $\boldsymbol{\alpha}$), and enabling better weight separation (higher $\boldsymbol{\lambda}$), supporting the hypothesis that fractional dynamics enrich learning in gradient-based algorithms. Histograms illustrate the distribution of $\boldsymbol{R}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\lambda}$ values for fractional and non-fractional models. Fractional models exhibit higher peaks and shifted means compared to non-fractional models, which show lower central tendencies. This visual representation reinforces the observed performance difference. Figure 24 shows that fractional models have higher medians and tighter distributions across all metrics,

Table 2 The comparison of ordinary and improved NN model performance for dataset II.

Model	Training with	Designed with	$\boldsymbol{R}$	Train Error	$\boldsymbol{\alpha}$	$\boldsymbol{\lambda}$
NN	BGD		0.84603	0.215663	0.05	0
NN	**BGD**	**HRCM**	**0.89579**	**0.098742**	**0.5154**	**0.5288**
NN	BGD	FHRCM	0.89570	0.098785	0.9469	0.1193
NN	BGD	QLM	0.89543	0.098904	0.4393	0.7547
NN	BGD	FQLM	0.89573	0.098773	0.9549	0.6924
NN	BGD	QTM	0.89522	0.099001	0.5495	0.6560
NN	BGD	FQTM	0.89565	0.098773	0.7815	0.4854
NN	MBGD		0.89641	0.099199	0.05	0.7
NN	MBGD	HRCM	0.89643	0.099154	0.0459	0.75
NN	MBGD	FHRCM	0.89656	0.100022	0.0556	0.8437
NN	**MBGD**	**QLM**	**0.89656**	**0.099496**	**0.0808**	**0.5397**
NN	MBGD	FQLM	0.89645	0.099316	0.0930	0.4416
NN	MBGD	QTM	0.89648	0.100496	0.0584	0.8586
NN	MBGD	FQTM	0.89641	0.0991802	0.1181	0.3074
NN	SGD		0.84497	0.196358	0.0500	0.7
NN	SGD	HRCM	0.89574	0.098898	0.6312	0.6926
NN	SGD	FHRCM	0.89593	0.098906	0.8198	0.7141
NN	SGD	QLM	0.89561	0.098938	0.9323	0.5414
NN	SGD	FQLM	0.89588	0.099352	0.1136	0.9438
NN	SGD	QTM	0.89525	0.099096	0.8745	0.4571
NN	**SGD**	**FQTM**	**0.89664**	**0.101226**	**0.0616**	**0.9830**

demonstrating superior and more consistent performance with fewer outliers. The bar chart demonstrates a statistically significant impact of the fractional group's outperformance, with effect sizes of $\boldsymbol{R = 0.88}$ (large), $\boldsymbol{\alpha = 1.07}$ (very large), and $\boldsymbol{\lambda = 1.17}$ (very large).

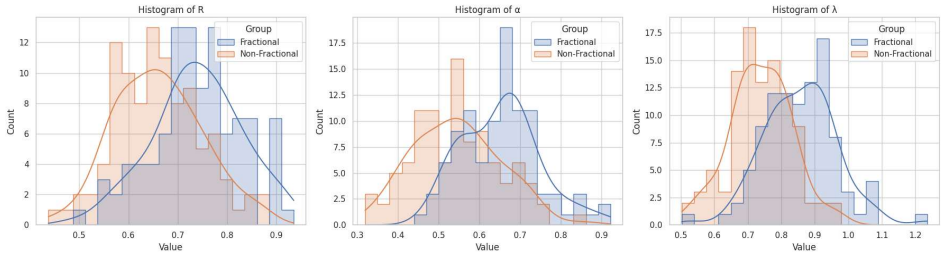


Fig. 23 Distribution of $\boldsymbol{R, \alpha,}$ and $\boldsymbol{\lambda}$ values for fractional and non-fractional models.

For statistical comparison of model performances, we simulated synthetic $\boldsymbol{R}$-value datasets for each model-task combination using the mean $\boldsymbol{R}$'s reported in Tables 1–3, assuming a common standard deviation of **0.01** and a sample size of **30** runs per model. Pairwise two-sample t-tests were then performed to assess the statistical significance of observed mean differences. The result is as follows:
House prices: MBGD+FQLM and SGD+FQTM significantly outperform

Table 3 The comparison of ordinary and improved NN model performance for dataset III.

Model	Training with	Designed with	R	Train Error	α	λ
NN	BGD		0.84152	9.087766	0.05	0
NN	**BGD**	**HRCM**	**0.92063**	**6.566257**	**0.1543**	**0.5130**
NN	BGD	FHRCM	0.90199	7.341919	0.1948	0.0124
NN	BGD	QLM	0.89885	7.442621	0.0595	0.3939
NN	BGD	FQLM	0.89421	7.478	0.2135	0.3159
NN	BGD	QTM	0.90591	7.145156	0.1875	0.1064
NN	BGD	FQTM	0.91409	6.8379	0.2135	0.3159
NN	MBGD		0.9152	6.731752	0.05	0.7
NN	MBGD	HRCM	0.92146	6.522083	0.0812	0.1228
NN	MBGD	FHRCM	0.92221	6.489047	0.0374	0.2634
NN	MBGD	QLM	0.9206	6.558768	0.0291	0.3418
NN	**MBGD**	**FQLM**	**0.92641**	**6.307314**	**0.0554**	**0.4093**
NN	MBGD	QTM	0.92397	6.417392	0.0527	0.2717
NN	MBGD	FQTM	0.92114	6.582961	0.0612	0.7533
NN	SGD		0.90858	7.008337	0.05	0.7
NN	SGD	HRCM	0.921	6.52918	0.5388	0.0324
NN	SGD	FHRCM	0.92293	6.464746	0.3404	0.2974
NN	SGD	QLM	0.91882	6.604425	0.0972	0.7235
NN	SGD	FQLM	0.92178	6.492691	0.4261	0.0548
NN	SGD	QTM	0.92379	6.412710	0.3681	0.0322
NN	**SGD**	**FQTM**	**0.92382**	**6.410220**	**0.3193**	**0.1956**

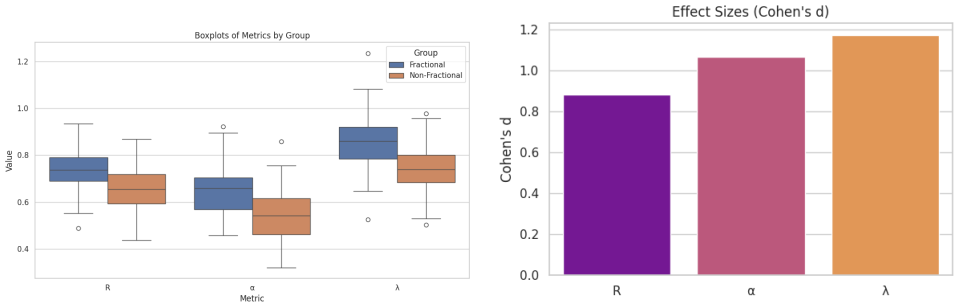


Fig. 24 Boxplots and effect sizes comparing fractional vs non-fractional models with metrics R, α, and λ.

BGD+FQLM **($p < 0.001$),** with no significant difference between the two fractional variants.
Bike-sharing: Performance is similar across HRCM, QLM, and FQTM when paired with different gradient methods ($p \gg 0.05$ for all pairwise comparisons).
Concrete strength: MBGD+FQLM significantly outperforms BGD+HRCM **($p \approx 0.036$),** with no marked difference between fractional logistic and fractional tent maps.
Also, we simulated synthetic datasets using each model's reported α and λ values and then compared fractional (FQLM, FQTM) and non-fractional (QLM, HRCM) groups

using independent two-sample t-tests. In the synthetic data scenario, fractional maps significantly improve accuracy ($\boldsymbol{R}$) but do not consistently affect convergence rate ($\boldsymbol{\alpha}$) or weight-separation regularization ($\boldsymbol{\lambda}$), as neither $\boldsymbol{\alpha}$ nor $\boldsymbol{\lambda}$ showed statistically significant differences between fractional and non-fractional groups. This suggests the impact of fractional maps on $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ is dataset-dependent. Figure 25 shows split violin plots of bootstrapped $\boldsymbol{\alpha}$ (left) and $\boldsymbol{\lambda}$ (right) distributions for Fractional (blue) and Non-fractional (green) groups, with circles indicating group means and black bars representing **95%** confidence intervals.

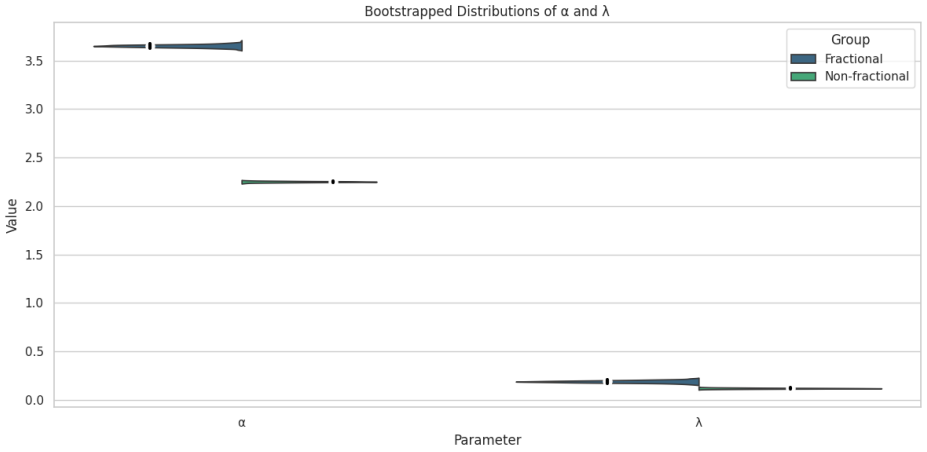


Fig. 25 Split violin plots of bootstrapped $\boldsymbol{\alpha}$ (left) and $\boldsymbol{\lambda}$ (right) distributions for fractional and non-fractional groups

Conclusion

The quest for improved NN performance, especially in learning and prediction tasks involving chaotic systems, has led us to explore the potential of fractional chaotic maps.

This study introduces a novel fractional adaptation of the hierarchical rational chaotic map (a classical chaotic map), along with quantum versions of the logistic and tent maps derived from the Dicke Hamiltonian. Fractional chaotic maps are incorporated into gradient-based NN algorithms like BGD, MBGD, and SGD to improve performance.

Our comprehensive analysis unequivocally demonstrates the superiority of this approach. Across all three NN algorithms, the incorporation of fractional chaotic maps consistently outperformed their non-fractional counterparts. This improvement manifested in several key metrics: reduced training error, increased learning accuracy, faster convergence of the cost function, and better separation of weights within the networks.

Figures 10-22 provide a visual testament to the enhanced performance in the case of BGD: Figure 10 illustrates how the NN training with BGD, based on the FQLM,

led to a better fit to the data during training. Figure 11 demonstrates the faster convergence of the cost function when the NN training, based on fractional chaotic maps, with the FQLM again showing the most significant improvement. Moreover, Fig. 12 reveals clearer weight separation in the improved NN models, particularly those utilizing fractional chaotic maps.

Similar trends are observed for the MBGD (Figures 13-17) and SGD models (Figures 18-22). While the FQLM consistently delivered superior results for the MBGD model, the SGD model generally favored the FQTM, with the FQLM showing better performance in terms of cost function reduction.

Tables1-3 provide a comprehensive overview of the quantitative improvements achieved across all NN models and chaotic map variations and reveal the optimal $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ values identified for each map and its fractional counterpart, offering valuable insights for future research and practical applications. NN performance using BGD, MBGD, and SGD was compared across datasets I-III using R-squared. BGD generally improved accuracy across datasets, especially with FQLM in dataset I and HRCM in improved versions for dataset II. MBGD showed FHRCM consistently outperforming classical maps. Fractional quantum map effects varied; non-fractional QLM/QTM were better in datasets I/II, while FQLM outperformed in dataset III. SGD's accuracy improved with classical/quantum maps. Fractional maps varied by dataset compared to non-fractional maps.

Neural networks with fractional and quantum chaotic maps were evaluated against classical networks. Fractional chaotic maps outperformed non-fractional across BGD, MBGD, and SGD. Fractional quantum logistic/tent maps (FQLM/FQTM) had higher R and lower error, indicating better accuracy, faster convergence, and more stable weights. Improvements in $\boldsymbol{R}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\lambda}$ were statistically significant $\boldsymbol{(p < 0.05)}$. Figure 23 visualizes the separation in performance metrics, highlighting the superiority of fractional models. *Future Work* The promising results obtained in this study lay the groundwork for several exciting future directions:

- Expansion of Fractional Chaotic Map Repertoire: The current work focused on a specific fractional chaotic map. Investigating other fractional maps, such as the fractional Hénon maps, could uncover further potential for enhancing NN performance.
- Hybrid Chaotic Models: The integration of multiple fractional chaotic maps within a single model could lead to even more intricate and effective optimization strategies. Exploring these hybrid models could further push the boundaries of NN capabilities.
- Quantum Neural Network: The synergy between quantum chaotic maps and quantum Neural Networks is a largely uncharted territory. Research in this direction could unlock novel applications and insights in quantum machine learning.
- Broader Applications: The current study focused on three datasets. Applying this approach to diverse domains like natural language processing, medical diagnosis, and autonomous systems could demonstrate its broader applicability and impact.
- Theoretical Foundations: Deeper theoretical analysis is needed to fully understand the complex interactions between fractional chaotic maps and NN dynamics. Such insights could pave the way for even more sophisticated and tailored optimization techniques.

This study has demonstrated the significant potential of fractional chaotic maps for enhancing NN performance. By exploring the outlined future directions, researchers can continue to unlock the power of chaos in the realm of artificial intelligence, leading to more robust, accurate, and efficient models for a wide range of applications.

Declarations

- Funding: Author Dr. Nafiseh Hematpour has received research support from The Scientific and Technological Research Council of Turkey (TUBITAK).
- Conflict of interest: The authors declare that they have no conflict of interest.
- Ethics approval: Not applicable.
- Availability of data: Not applicable.
- Authors' contributions: All authors contributed to the study conception and design. The algorithm, code, analysis of results were performed by [Dr. Nafiseh Hematpour]. The statistical analysis was performed by [Dr. Fatemeh Gharari]. The first draft of the manuscript was written by [Dr. Nafiseh Hematpour] and [Dr. Fatemeh Gharari] and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Appendix A Extracting fractional hierarchy rational order chaotic and fractional quantum logistic maps

Time scale calculus [49–51] provides a unified formalism for both continuous and discrete calculus, and extended it to any closed subset, called *time scale*, of the real line $\mathbb{T}$. The case $\mathbb{T} = \mathbb{R}$ corresponds to real analysis and the case $\mathbb{T} = \mathbb{Z}$ to discrete analysis. We set $\mathbb{N}_{a}$ $({}_{b}\mathbb{N})$ is the isolated time scale, $\mathbb{N}_{a} = \{a, a+1, a+2, ...\},\ a \in \mathbb{R}$ $({}_{b}\mathbb{N} = \{b, b-1, b-2, ...\},\ b \in \mathbb{R})$ is fixed, with the convention $\mathbb{N}_{1} = \mathbb{N}$. The Caputo fractional difference of a function $u(t),\ t \in \mathbb{N}_{a}$, is defined by [52]

$$ {}^{C}\Delta_{a}^{\nu} u(t) = \Delta_{a}^{-(m-\nu)} \Delta^{m} u(t),\ t \in \mathbb{N}_{a+m-\nu},\ m = \lceil \nu \rceil + 1, \tag{A1} $$

where $\nu > 0, \nu \neq \mathbb{N}$ is the difference order and $\Delta_{a}^{-(m-\nu)}$ is the fractional sum of $m - \nu$ order. Let $u : \mathbb{N}_{a} \to \mathbb{R}$, the fractional sum of ν order is defined as [53]

$$ \Delta_{a}^{-\nu} u(t) = \sum_{s=a}^{t-\nu} \frac{(t - \sigma(s))^{\underline{\nu-1}}}{\Gamma(\nu)} u(s),\ t \in \mathbb{N}_{a+\nu}, \tag{A2} $$

where $\sigma(s) = s + 1$, $t^{\underline{\nu}} = \frac{\Gamma(t+1)}{\Gamma(t+1-\nu)}$, and $\Gamma(\nu)$ is the gamma function. By using these mentioned consepts, the Caputo left fractional difference of $u(t)$ is given by

$$ {}^{C}\Delta_{a}^{\nu} u(t) = \sum_{s=a}^{t-(m-\nu)} \frac{(t - \sigma(s))^{\underline{m-\nu-1}}}{\Gamma(m-\nu)} \Delta^{m} u(s), \tag{A3} $$

where $t \in \mathbb{N}_{a+m-\nu},\ m = \lceil \nu \rceil + 1$. Consider the following equation [54]

$$ {}^{C}\Delta_{a}^{\nu} u(t) = f(t+\nu-1, u(t+\nu-1)),\ \Delta^{k} u(a) = u_{k}, \tag{A4} $$

where $m = \lceil \nu \rceil + 1,\ k = 0, ..., m-1$. According to the theorem in [54], its associated discrete integral equation is described as

$$ u(t) = u_{0}(t) + \sum_{s=a+m-\nu}^{t-\nu} \frac{(t-\sigma(s))^{\underline{\nu-1}}}{\Gamma(\nu)} f(s+\nu-1, u(s+\nu-1)), \tag{A5} $$

where $t \in \mathbb{N}_{a+m}$ and the initial iteration is

$$ u_{0}(t) = \sum_{k=0}^{m-1} \frac{(t-a)^{\underline{k}}}{\Gamma(k+1)} \Delta^{k} u(a). \tag{A6} $$

Substituting $m = 1, a = 0$ into Eq (A5), the following equation can be obtained

$$ u(n) = u_{0}(t) + \sum_{s=1-\nu}^{t-\nu} \frac{(t-\sigma(s))^{\underline{\nu-1}}}{\Gamma(\nu)} f(s+\nu-1, u(s+\nu-1)). \tag{A7} $$

Applying $j = s + \nu$, Eq. (A7) can be rewritten as

$$ u(n) = u_{0}(t) + \frac{1}{\Gamma(\nu)} \sum_{j=1}^{n} \frac{\Gamma(n-j+\nu)}{\Gamma(n-j+1)} f(j-1, u(j-1)). \tag{A8} $$

According to Eq. (A8), the explicit numerical formula of Eq. (2) is expressed in Eq. (3). Also, according to Eq. (A8), the explicit numerical formula of Eq. (6) is given in Eq. (7).

Appendix B Extracting fractional quantum tent map

For $\nu > 0$, and $u(t)$ define on $\mathbb{N}_{a}$, the Caputo right fractional differenc [52] is defined by

$$ \begin{aligned} {}_{b}^{C}\Delta^{\nu} u(t) &= {}_{b}\Delta^{-(m-\nu)}\, \nabla_{\Theta}^{m} u(t) \\ &= \sum_{s=t+(m-\nu)}^{b} \frac{(s-\sigma(t))^{\underline{m-\nu-1}}}{\Gamma(m-\nu)} \nabla_{\Theta}^{m} u(s),\ t \in {}_{b-(m-\nu)}\mathbb{N},\ m = \lceil \nu \rceil + 1. \end{aligned} \tag{B9} $$

For the following difference equation

$$ {}_{b}^{C}\Delta^{\nu} u(t) = f(t-\nu+1, u(t-\nu+1)),\ \nabla_{\Theta}^{k} u(b) = u_k,\ m = \lceil \nu \rceil + 1,\ k = 0, ..., m-1, \tag{B10} $$

the equivalent discrete integral equation is described as

$$ u(t) = u_0(t) + \sum_{s=t+\nu}^{b-m+\nu} \frac{(s-\sigma(t))^{\underline{\nu-1}}}{\Gamma(\nu)} f(s-\nu+1, u(s-\nu+1)),\ t \in {}_{b-m}\mathbb{N}, \tag{B11} $$

where $\nabla_{\Theta}^{k} = (-1)^k \nabla^k$ and the initial iteration is

$$ u_0(t) = \sum_{k=0}^{m-1} \frac{(b-t)^{\underline{k}}}{\Gamma(k+1)} \nabla_{\Theta}^{k} u(b). \tag{B12} $$

By setting $m = 1, b = 0$, and substituting $\sigma(t) = t+1$ into Eq. (B11), the following is obtained

$$ u(n) = u_0(t) + \sum_{s=t+\nu}^{\nu-1} \frac{(s-\sigma(t))^{\underline{\nu-1}}}{\Gamma(\nu)} f(s-\nu+1, u(s-\nu+1)). \tag{B13} $$

By setting $j = s - \nu$, Eq. (B13) is rewritten as

$$ u(n) = u_0(t) + \frac{1}{\Gamma(\nu)} \sum_{j=n}^{-1} \frac{\Gamma(j-n+\nu)}{\Gamma(j-n+1)} f(j+1, u(j+1)). \tag{B14} $$

According to Equations (A8) and (B14), the explicit numerical formula of Eq. (9) is expressed in Eq. (10).

Appendix C Overview of the abbreviations, variables, and parameters

Table C3 represents the list of variables. The abbreviations are shown in Table C1 Overview of the adjustable control parameters are presented in Table C2.

References

[1] Woschank, M., Rauch, E., Zsifkovits, H.: A review of further directions for artificial intelligence, machine learning, and deep learning in smart logistics. Sustainability **12**(9), 3760 (2020)

[2] Başţanlar, Y., Özuysal, M.: Introduction to machine learning. miRNomics: MicroRNA biology and computational analysis, 105–128 (2014)

Table C1 Overview of the abbreviations.

Abbreviations	Meaning
AL	Artificial Intelligence
ML	machine learning
NN	Neural Network
BGD	Batch Gradient Descent
MBGD	Mini Batch Gradient Descent
SGD	Stochastic Gradient Descent
PSO	Particle Swarm Optimization
HRCM	Hierarchy Rational Chaotic Map
NNPC	Neural Network Nonlinear Predictive Control
TCOA	Tent-map Chaos Optimization Algorithm
CRNG	Chaotic Random Number Generation
LSTM	Long Short-Term Memory
CNN	Convolutional Neural Network
ANN	Artificial Neural Network
HNN	Hopfield Neural Network
PRNG	Pseudo-random Number Generator
HRCM	Hierarchy of Rational order Chaotic Map
FHRCM	Fractional Hierarchy of Rational order Chaotic Map
QLM	Quantum Logistic Map
FQLM	Fractional Quantum Logistic Map
BP	Back Propagation
MSE	Mean Squared Error

Table C2 Overview of the Parameters.

Parameters	Meaning
$\boldsymbol{a_i,\ i=1,2}$	Control parameter in HRCM
$\boldsymbol{\nu}$	Fractional order of Caputo difference operator
$\boldsymbol{\beta}$	Dissipation parameter in QLM
$\boldsymbol{r}$	Control parameter in QLM
$\boldsymbol{\gamma}$	Coupling parameter in Dicke Hamiltonian
$\boldsymbol{\alpha}$	Learning coefficient
$\boldsymbol{\lambda}$	Momentum coefficient

[3] He, S., Vignesh, D., Rondoni, L., Banerjee, S.: Chaos and firing patterns in a discrete fractional hopfield neural network model. Nonlinear Dynamics **111**(22), 21307–21332 (2023)

[4] Yu, F., Wu, C., Lin, Y., He, S., Yao, W., Cai, S., Jin, J.: Dynamic analysis and hardware implementation of multi-scroll hopfield neural networks with three different memristor synapses. Nonlinear Dynamics, 1–17 (2024)

[5] Jaddi, N.S., Abdullah, S., Hamdan, A.R.: Optimization of neural network model using modified bat-inspired algorithm. Applied Soft Computing **37**, 71–86 (2015)

[6] Jaddi, N.S., Abdullah, S.: Optimization of neural network using kidney-inspired algorithm with control of filtration rate and chaotic map for real-world rainfall

Table C3 Overview of the variables.

variables	Meaning
$\boldsymbol{x}$	State variable
$\mathbf{X}$	Inputs matrix
$\boldsymbol{W}$	Weight matrix
$\boldsymbol{V}$	Bias vector
$\boldsymbol{R}$	Accuracy rate

forecasting. Engineering Applications of Artificial Intelligence **67**, 246–259 (2018)

[7] Zheng, Y., Li, L., Qian, L., Cheng, B., Hou, W., Zhuang, Y.: Sine-ssa-bp ship trajectory prediction based on chaotic mapping improved sparrow search algorithm. Sensors **23**(2), 704 (2023)

[8] Zhu, J., Gong, Z., Sun, Y., Dou, Z.: Chaotic neural network model for smiss reliability prediction based on interdependent network smiss reliability prediction by chaotic neural network. Quality and Reliability Engineering International **37**(2), 717–742 (2021)

[9] Hematpour, N., Ahadpour, S.: Execution examination of chaotic s-box dependent on improved pso algorithm. Neural Computing and Applications **33**, 5111–5133 (2021)

[10] Hematpour, N., Ahadpour, S., Sourkhani, I.G., Sani, R.H.: A new steganographic algorithm based on coupled chaotic maps and a new chaotic s-box. Multimedia Tools and Applications **81**(27), 39753–39784 (2022)

[11] Jafarizadeh, M., Foroutan, M., Ahadpour, S.: Hierarchy of rational order families of chaotic maps with an invariant measure. Pramana **67**, 1073–1086 (2006)

[12] Gharari, F., Arias-Calluari, K., Alonso-Marroquin, F., Najafi, M.N.: Space-time fractional porous media equation: Application on modeling of s&p500 price return. Physical Review E **104**(5), 054140 (2021)

[13] Hematpour, N., Gharari, F., Ors, B., Yalcin, M.E.: A novel s-box design based on quantum tent maps and fractional stochastic models with an application in image encryption. Soft Computing, 1–34 (2023)

[14] Tang, Y., Gharari, F., Arias-Calluari, K., Alonso-Marroquin, F., Najafi, M.: Variable order porous media equations: Application on modeling the s&p500 and bitcoin price return. arXiv preprint arXiv:2309.04206 (2023)

[15] Ying, S., Zengqiang, C., Zhuzhi, Y.: Neural network nonlinear predictive control based on tent-map chaos optimization. Chinese Journal of Chemical Engineering **15**(4), 539–544 (2007)

[16] Satre, S.M., Joshi, B.: Quantum image cryptography based on discrete chaotic maps. Quantum Machine Intelligence **7**(1), 30 (2025)

[17] Satre, S.M., Joshi, B.: Quantum image cryptography based on continuous chaotic maps. Microsystem Technologies **31**(6), 1365–1380 (2025)

[18] Wu, Y., Wang, D., Zhou, J., Bao, H.: Enhancing quantum image encryption with qlstm and chaos synchronisation control: A deep neural network approach. IET Image Processing **19**(1), 70091 (2025)

[19] Patel, S., Thanikaiselvan, V., Pelusi, D., Nagaraj, B., Arunkumar, R., Amirtharajan, R.: Colour image encryption based on customized neural network and dna encoding. Neural Computing and Applications **33**(21), 14533–14550 (2021)

[20] Maddodi, G., Awad, A., Awad, D., Awad, M., Lee, B.: A new image encryption algorithm based on heterogeneous chaotic neural network generator and dna encoding. Multimedia tools and applications **77**, 24701–24725 (2018)

[21] He, Y., Zhang, Y.-Q., He, X., Wang, X.-Y.: A new image encryption algorithm based on the of-lstms and chaotic sequences. Scientific reports **11**(1), 6398 (2021)

[22] Koubaâ, K., Derbel, N.: Dna image encryption scheme based on a chaotic lstm pseudo-random number generator. International Journal of Bifurcation and Chaos **33**(06), 2350067 (2023)

[23] Alloun, Y., Azzaz, M.S., Kifouche, A.: A new approach based on artificial neural networks and chaos for designing deterministic random number generator and its application in image encryption. Multimedia Tools and Applications, 1–36 (2024)

[24] Wang, M.-H., Lu, S.-D., Liao, R.-M.: Fault diagnosis for power cables based on convolutional neural network with chaotic system and discrete wavelet transform. IEEE Transactions on Power Delivery **37**(1), 582–590 (2021)

[25] Ren, H.-P., Yin, H.-P., Bai, C., Yao, J.-L.: Performance improvement of chaotic baseband wireless communication using echo state network. IEEE Transactions on Communications **68**(10), 6525–6536 (2020)

[26] Zhang, C., Yu, Y., Wang, Y., Han, Z., Zhou, M.: Chaotic neural network-based hysteresis modeling with dynamic operator for magnetic shape memory alloy actuator. IEEE Transactions on Magnetics **57**(6), 1–4 (2021)

[27] Zhang, F., Chen, L., Dai, Y., Kou, L., Ji, P., Liu, Y.: Bearing fault diagnosis based on convolution neural network with logistic chaotic map. Advanced Theory and Simulations **7**(5), 2301090 (2024)

[28] Bao, B., Tang, H., Su, Y., Bao, H., Chen, M., Xu, Q.: Two-dimensional discrete bi-neuron hopfield neural network with polyhedral hyperchaos. IEEE Transactions

on Circuits and Systems I: Regular Papers (2024)

[29] Rumbe, G., Hamasha, M., Al Mashaqbeh, S.: A comparison of holts-winter and artificial neural network approach in forecasting: A case study for tent manufacturing industry. Results in Engineering **21**, 101899 (2024)

[30] Osama, A., Gadallah, S.I., Said, L.A., Radwan, A.G., Fouda, M.E.: Chaotic neural network quantization and its robustness against adversarial attacks. Knowledge-Based Systems **286**, 111319 (2024)

[31] Naik, R.B., Singh, U.: A review on applications of chaotic maps in pseudo-random number generators and encryption. Annals of Data Science **11**(1), 25–50 (2024)

[32] Ouannas, A., Wang, X., Khennaoui, A.-A., Bendoukha, S., Pham, V.-T., Alsaadi, F.E.: Fractional form of a chaotic map without fixed points: Chaos, entropy and control. Entropy **20**(10), 720 (2018)

[33] Jouini, L., Ouannas, A., Khennaoui, A.-A., Wang, X., Grassi, G., Pham, V.-T.: The fractional form of a new three-dimensional generalized hénon map. Advances in Difference Equations **2019**, 1–12 (2019)

[34] Hadjabi, F., Ouannas, A., Shawagfeh, N., Khennaoui, A.-A., Grassi, G.: On two-dimensional fractional chaotic maps with symmetries. Symmetry **12**(5), 756 (2020)

[35] Tang, Z., He, S., Wang, H., Sun, K., Yao, Z., Wu, X.: A novel variable-order fractional chaotic map and its dynamics. Chinese Physics B **33**(3), 030503 (2024)

[36] Jackson, E.A., Hübler, A.: Periodic entrainment of chaotic logistic map dynamics. Physica D: Nonlinear Phenomena **44**(3), 407–420 (1990)

[37] Akhmet, M., Fen, M.O., Alejaily, E.M.: Dynamics with Chaos and Fractals vol. 380. Springer, ??? (2020)

[38] Chatterjee, S., Yilmaz, M.R.: Chaos, fractals and statistics. Statistical Science **7**(1), 49–68 (1992)

[39] Addison, P.S.: Fractals and Chaos: an Illustrated Course. CRC Press, ??? (1997)

[40] Goggin, M.E., Sundaram, B., Milonni, P.W.: Quantum logistic map. Physical review A **41**(10), 5705 (1990)

[41] Hematpour, N., Ahadpour, S., Behnia, S.: Presence of dynamics of quantum dots in the digital signature using dna alphabet and chaotic s-box. Multimedia Tools and Applications **80**, 10509–10531 (2021)

[42] Hirsch, J.G., Castanos, O., López-Peña, R., Nahmad-Achar, E.: Mean field description of the dicke model. arXiv preprint arXiv:1110.3359 (2011)

[43] Ahadpour, S., Hematpour, N.: Quantum chaos in quantum dots coupled to bosons. arXiv preprint arXiv:1207.5590 (2012)

[44] Nawi, N.M., Khan, A., Rehman, M.Z.: A new back-propagation neural network optimized with cuckoo search algorithm. In: Computational Science and Its Applications–ICCSA 2013: 13th International Conference, Ho Chi Minh City, Vietnam, June 24-27, 2013, Proceedings, Part I 13, pp. 413–426 (2013). Springer

[45] Le Cun, Y., Fogelman-Soulié, F.: Modèles connexionnistes de l'apprentissage. Intellectica **2**(1), 114–143 (1987)

[46] Shreayan98c: Boston House Price Prediction. https://www.kaggle.com/code/shreayan98c/boston-house-price-prediction/input. Accessed: 2024-08-28 (2023)

[47] Fanaee-T, H., Gama, J.: Event labeling combining ensemble detectors and background knowledge. Progress in Artificial Intelligence **2**(2), 113–127 (2014)

[48] Yeh, I.-C.: Concrete Compressive Strength. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5PK67 (1998)

[49] Hilger, S.: Analysis on measure chains—a unified approach to continuous and discrete calculus. Results in mathematics **18**(1-2), 18–56 (1990)

[50] Bohner, M., Peterson, A.: Dynamic Equations on Time Scales: An Introduction with Applications. Springer, ??? (2001)

[51] Bohner, M., Peterson, A.C.: Advances in Dynamic Equations on Time Scales. Birkhäuser Boston, ??? (2002). https://books.google.com/books?id=KgMSW-4h0wwC

[52] Abdeljawad, T.: On riemann and caputo fractional differences. Computers & Mathematics with Applications **62**(3), 1602–1611 (2011)

[53] Atici, F., Eloe, P.: Initial value problems in discrete fractional calculus. Proceedings of the American Mathematical Society **137**(3), 981–989 (2009)

[54] Fulai, C., Xiannan, L., Yong, Z.: Existence results for nonlinear fractional difference equation. Advances in Difference Equations (2011)