applied sciences

MDPI

Article

Prediction of Ship-Unloading Time Using Neural Networks

Zhen Gao [1,2,*], Danning Li [1,2], Danni Wang [1,2], Zengcai Yu [1,2], Witold Pedrycz [3] and Xinhai Wang [4]

[1] National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Northeastern University, Shenyang 110819, China; danning1019@163.com (D.L.); danni_present@foxmail.com (D.W.); yu_zengcai@163.com (Z.Y.)
[2] Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, Shenyang 110819, China
[3] The Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6R 2V4, Canada; wpedrycz@ualberta.ca
[4] College of Sciences, Northeastern University, Shenyang 110819, China; 2270021@stu.neu.edu.cn
* Correspondence: gaozhen@ise.neu.edu.cn

Abstract: The prediction of unloading times is crucial for reducing demurrage costs and ensuring the smooth scheduling of downstream processes in a steel plant. The duration of unloading a cargo ship is primarily determined by the unloading schedule established at the raw materials terminal and the storage operation schedule implemented in the stockyard. This study aims to provide an accurate forecast of unloading times for incoming ships at the raw materials terminal of a steel plant. We propose three neural network-based methods: the Backpropagation Neural Network (BP), the Random Vector Functional Link (RVFL), and the Stochastic Configurations Network (SCN) for this prediction. This issue has not been previously researched using similar methods, particularly in the context of large-scale steel plants. The performance of these three methods is evaluated based on several indices: the Root Mean Square Error (RMSE), the quality of the best solution, convergence, and stability, which are employed for predicting unloading times. The prediction accuracies achieved by the BP, RVFL, and SCN were 76%, 85%, and 87%, respectively. These results demonstrate the effectiveness and potential applications of the proposed methods.

Keywords: prediction; ship-unloading; neural networks; steel plants

Citation: Gao, Z.; Li, D.; Wang, D.; Yu, Z.; Pedrycz, W.; Wang, X. Prediction of Ship-Unloading Time Using Neural Networks. *Appl. Sci.* **2024**, *14*, 8213. https://doi.org/10.3390/app14188213

Academic Editor: Rui Araújo

Received: 8 August 2024
Revised: 6 September 2024
Accepted: 11 September 2024
Published: 12 September 2024

1. Introduction

The prediction of unloading times is crucial for reducing delay costs and ensuring the smooth scheduling of downstream processes in a large steel plant [1,2]. In fact, accurately predicting unloading times can create favorable conditions for the more efficient allocation of unloading resources for subsequently arriving ships. Additionally, it can provide a buffer for subsequent storage operations in the storage yard. In a large-scale iron and steel enterprise, hundreds or even thousands of ships transport tens of millions of tons of raw materials to the terminal each year. These ships must be unloaded promptly to minimize delay costs and meet production demands for raw materials. However, significant demurrage costs—amounting to hundreds of millions annually—arise when many ships fail to unload on time due to various factors.

Two key factors influence the unloading time of ships: the ship-unloading schedule established at the raw materials terminal and the raw materials storage operation schedule implemented in the stockyard. The ship-unloading system of a steel plant is illustrated in Figure 1. Typically, this system consists of several berths, a limited number of ship unloaders, and a conveyor transmission system. The berths are designated for docking raw material ships, while the ship unloaders are responsible for unloading the raw materials. These unloaders can move along a fixed track. For smaller vessels, typically those under 50,000 tons, a single ship unloader is enough to meet unloading requirements. However, for larger vessels, generally exceeding 150,000 tons, it is necessary to allocate two to

three ship unloaders to facilitate the unloading operations. The conveyor system is installed throughout the entire port area, the stockyard, and certain production areas to enable the bidirectional transfer of raw materials and products among these locations. When a ship arrives at the raw materials terminal, a berth and one or more ship unloaders are assigned to it for the unloading operation. Simultaneously, storage space must be designated for the unloaded raw materials in the stockyard. Accurate prediction of ship-unloading times can create buffer space for subsequent dock-unloading schedules and storage space allocation. This improvement facilitates the execution of related operations, reduces ship delay costs, and enhances economic efficiency, making it a critical factor in port operations.

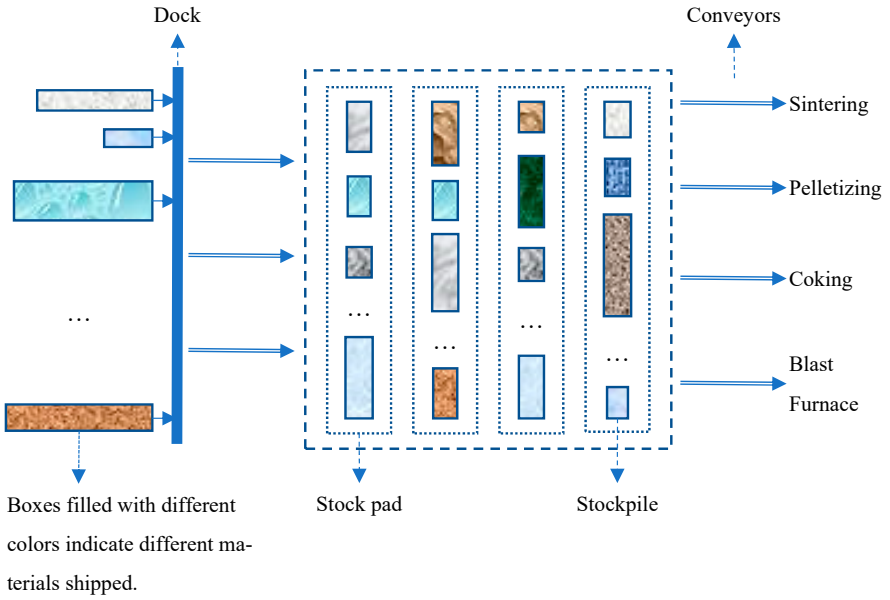

Figure 1. The ship-unloading system of a steel plant.

The prediction methods have become a prominent area of study in both industrial applications [3,4] and academic research [5,6]. These methods can be broadly categorized into three groups: statistical methods [7–9], mechanistic methods [10,11], and data-driven methods [12,13]. Statistical methods operate under the assumption that the observations of the predicted object follow specific statistical distributions, utilizing statistical theory to derive prediction results. In contrast, data-driven methods rely entirely on data, employing machine learning techniques to generate predictions. Mechanistic methods assume that the predicted object adheres to certain physical or chemical laws, including thermodynamics, dynamics, and fluid mechanics, and they derive predictions through the application of relevant mechanistic equations. However, data-driven methods have gained popularity due to their user-friendliness and lower implementation costs in industrial settings.

As far as we know, unloading time prediction is rarely studied for bulk carriers. However, there are several analogous studies in other industries. For instance, Zhang et al. [14] developed a fuzzy sequence-to-sequence network for forecasting unloading times in earthmoving projects. Liang and Wang [15] developed a modularized simulation method (MSM) for forecasting delivery times of priority lots in a 300 mm silicon wafer fabrication production line. Xu et al. [16] developed a robust berth-scheduling algorithm to address the uncertainty of vessel arrival delay and handling time. In response to this research gap, this study proposes a method that employs three neural network-based approaches: a Backpropagation Neural Network (BP), a Random Vector Functional Link Network (RVFL), and a Stochastic Configuration Network (SCN) model to predict unloading time at a steel plant raw materials terminal. There are several compelling reasons for selecting this prediction method:

Neural networks have been widely applied to prediction and classification and exhibited a great deal of success.

1. BP as a mature neural network and the ensuing learning method has been widely applied [17,18].
2. RVFL introduces an efficient way to find out the optimal network parameters and represents a richer structure variety of neural networks [19].
3. SCN exhibits a concise network structure and demonstrates a better application potential [20].

The primary contributions of this paper are:

1. Three neural network models-Backpropagation (BP), Random Vector Functional Link (RVFL), and Stochastic Configuration (SCN) have been developed for predictive applications.
2. Three prediction models are comprehensively evaluated based on their predictive accuracy, convergence, and stability, and are applied to a real-world scenario involving unloading-time prediction.

The paper is organized as follows. Section 2 covers the relevant studies. Section 3 includes a description of the relevant methods. Section 4 elaborates on the experimental results produced with the proposed methods. Section 5 includes the conclusions and identifies some future directions.

2. Literature Review

There are several studies related to predicting unloading time, which can be summarized from three perspectives: ship-unloading scheduling, stockyard storage operations, and the application of neural networks for prediction.

Ship-unloading scheduling: Dhingra et al. [21] proposed a two-level stochastic model to estimate a container ship's handling time. The higher-level model employed a continuous-time Markov chain to estimate the handling time for loading and unloading, while the lower-level model utilized a closed queuing network to provide the transition matrix as input for the higher-level model. Bish [22] addressed a container loading and unloading scheduling problem involving a set of ships, considering the storage location of each container, the assignment of vehicles to containers, and the scheduling of cranes for loading and unloading operations. The objective was to minimize the maximum time required to serve the set of ships. A heuristic approach was proposed to solve the problem, and the effectiveness of this heuristic was subsequently analyzed. Al-Dhaheri et al. [23] introduced a stochastic mixed-integer programming model aimed at minimizing container ship handling time. A genetic algorithm was developed to construct the crane schedule while adhering to the operational rules governing crane usage. Sun et al. [24] investigated a quay crane scheduling problem aimed at minimizing the completion time of loading and unloading operations for container ships. They formulated a mathematical programming model and proposed a Benders decomposition method to solve the problem. Sammarra et al. [25] introduced a tabu search heuristic for quay crane scheduling, focusing on minimizing the completion time of loading and unloading operations for container ships. They considered specific operational constraints for the quay cranes, decomposing the problem into a routing problem and a scheduling problem, which were addressed using a tabu search heuristic and a local search technique, respectively. Tang et al. [26] proposed a joint scheduling approach for quay cranes and trucks servicing a container ship. They developed a particle swarm optimization (PSO) method to tackle the scheduling problem. Kao et al. [27] presented a knowledge-based approach for ship unloading, aimed at minimizing total demurrage costs. Their knowledge base was designed to manage the sequencing of waiting ships and the transfer of ships between two docks. Kao et al. [28] proposed a heuristic approach for scheduling ship discharging. Their method considered the sequencing of holds and the assignment of ship unloaders, belt conveyor units, and stackers. Kao and Lee [29] introduced an integrated system for ship unloading, which coordinates two operations—dock assignment and ship discharging—to enhance the efficiency of unloading operations. Kim and Moon [30] developed a mixed-integer programming (MIP) model to address a berth-planning problem. In their approach, each ship was expected to remain for a fixed duration, irrespective of the berthing locations. A simulated annealing algorithm was devised to solve this problem. Kim et al. [31] investigated a ship-unloading

issue for a large-scale steel plant, aiming to minimize the total flow time of all ships at the seaport. They developed a heuristic to address this challenge. Gao et al. [1] examined the scheduling of ship unloading in a large-scale steel plant. They formulated a mathematical model and proposed a column-generation algorithm to solve the problem. Additionally, Gao et al. [2] introduced a differential evolution algorithm for ship-unloading scheduling, with special attention given to the assignment of belt conveyors.

Stockyard storage operations: Lee et al. [32] proposed a mixed-integer programming (MIP) model for yard truck scheduling and storage allocation. Their objective was to minimize the weighted total cost, which included penalties for total delays and the costs associated with total travel time. A constructive heuristic was developed for the MIP model, resulting in a solution gap of 10.27% when compared to CPLEX. The solution time for the heuristic was at the second level, while CPLEX required 30 to 40 h to find a solution. Tang et al. [33] investigated a stockyard storage space-allocation problem at a large iron ore terminal. They formulated a mixed-integer linear programming model aimed at minimizing the total travel distance for all incoming iron ores. The primary constraints included rules governing space allocation and the operations of stacker-reclaimers. A genetic algorithm-based heuristic was employed to solve the model. For smaller problems, the heuristic can achieve optimal solutions in seconds; however, for larger-scale problems, CPLEX is unable to find an optimal solution within 2 h, while the heuristic can provide near-optimal solutions in seconds. Li and Tang [34] addressed a storage space-allocation problem in an iron-steel stockyard by developing a nonlinear programming model. The objective was to minimize transportation and penalty costs, with constraints related to geometric factors (length, width, and height) and differences in material types for spatial allocation. This problem was solved using an improved tabu search algorithm. Kim et al. [35] examined a raw materials storage-allocation problem for a large-scale steelworks plant. They proposed a mixed-integer linear programming model, which was solved using CPLEX 9.02. The model's constraints included maintaining a safety distance between two stockpiles and ensuring materials balance. Their method enhanced current production practices.

Neural networks in prediction: Various neural network variants have been employed for prediction and classification tasks. Zhang and Shin [36] proposed a probabilistic neural network for monitoring manufacturing processes. This network featured Gaussian-mixture distributed parameters, which enhanced computational efficiency. Li et al. [37] developed a long short-term memory (LSTM) neural network for time-series prediction, utilizing partial least squares (PLS) to simplify the network architecture. The streamlined LSTM model effectively balanced strong generalization capabilities with a compact structure. Xiao et al. [12] employed a Backpropagation Neural Network (BP) combined with rough set theory for power load forecasting, using rough sets for dimensionality reduction. This approach yielded improved prediction results. Adelia and Panakkat [13] introduced a probabilistic neural network for predicting earthquake magnitudes. Kosanoglu [38] proposed an ensemble model that integrates time-series clustering techniques with deep learning methods for wind speed forecasting. The model utilized a Dirichlet mixture model and dynamic time warping to cluster features from the time-series data. The results indicated that this feature-clustering approach is a promising framework for forecasting. Hussein et al. [39] developed a machine learning method that combines the random vector functional link (RVFL) network with a moth search algorithm to predict missing values of total algal counts during water quality monitoring. The moth search algorithm optimized the input features for the RVFL network, resulting in predicted algal values that closely matched real observations. El-Said et al. [40] proposed four machine learning algorithms: RVFL, support vector machine, social media optimization, and k-nearest neighbors, to assess the impact of air injection and transverse baffles on the thermohydraulic performance of shell and tube heat exchangers. The results demonstrated that the RVFL model effectively identified nonlinear relationships between operating conditions and process responses. Wang and Wang [41] introduced an SCN model for predicting component concentrations in sodium aluminate liquor. This mechanistic model was integrated to elucidate the relationships

among conductivity, temperature, and component concentrations in Bayer alumina production, with experimental results indicating high prediction accuracy. Li et al. [42] presented an improved SCN model for predicting ammonia nitrogen concentrations in water quality monitoring. They introduced a new inequality during the network construction process and proposed a node-selection method. The experimental results confirmed the effectiveness of the enhanced SCN model.

In summary, while there is a substantial amount of research on predicting ship-unloading times, most studies focus on container ships rather than bulk carriers. Furthermore, existing research on bulk carriers primarily addresses unloading scheduling rather than unloading-time prediction, which does not adequately reflect the significance of this issue. The study aims to address this gap.

3. Related Methods

3.1. BP Neural Network

BP is a multilayer neural network whose effectiveness has been demonstrated in numerous studies [17]. Typically, BP consists of one input layer, one or more hidden layers, and one output layer, with full connectivity between adjacent layers. The BP learning algorithm operates iteratively. The network parameters, which include weights and biases, are propagated forward through the network and adjusted backward to minimize error.

Given a group of samples, $\left\{(x_i, y_i),\ i = 1, \ldots, N,\ x_i \in R^d, y_i \in R^l\right\}$, assuming there is one hidden layer in the BP (Backpropagation) network. containing J nodes, denoted as $v_1, v_2, \ldots, v_J$. The input weights, biases for the hidden nodes, output weights of the hidden nodes, and thresholds for the output nodes are $w_{ij}, b_j, w_{jk}, \theta_k, i = 1, \ldots, d$, j = $1, \ldots, J$, k = $1, \ldots, l$, respectively. Then,

1. Hidden layer outputs

$$v_j = f_1(\sum_{i=0}^{d} w_{ij} x_i),\ j = 0, 1, \ldots, J, \tag{1}$$

f_1 serves as the activation function for hidden node j, $x_0 = -1$, $w_{0j} = b_j$.

2. Output layer results

$$\overline{y}_k = f_2(\sum_{j=0}^{J} w_{jk} v_j),\ k = 0, 1, \ldots, l, \tag{2}$$

f_2 serves as the activation function for output node k, $v_0 = -1$, $w_{0k} = \theta_k$.

3. Updating of weights

$$\begin{aligned} &\Delta w_{jk} = \eta g_k v_j, \\ &j = 0, \ldots, J,\ k = 1, \ldots, l, \\ &g_k \triangleq \overline{y}_k(1 - \overline{y}_k)(y_k - \overline{y}_k). \end{aligned} \tag{3}$$

$$\begin{aligned} &\Delta w_{ij} = \eta e_j x_i, \\ &i = 0, \ldots, d,\ j = 1, \ldots, J, \\ &e_j \triangleq -v_j(1 - v_j) \sum_{k=1}^{l} w_{jk} g_k,\ g_k \triangleq \overline{y}_k(1 - \overline{y}_k)(y_k - \overline{y}_k). \end{aligned} \tag{4}$$

However, the Backpropagation (BP) algorithm is inherently gradient based, which means there is a possibility of converging to local minima. Nevertheless, it remains a robust method due to various strategies designed to mitigate the risk of local optimality. These strategies include the momentum method, adaptive learning rates, and the use of first-order approximations, such as the Newton and quasi-Newton methods, instead of relying solely on the gradient method. The BP algorithm is detailed in Algorithm 1.

Algorithm 1: BP algorithm

Input: samples $\{x_i, y_i\}$, $x_i \in R^d$, $y_i \in R^l$, $i = 1, \ldots, N$; J
Output: w_{ij}, w_{jk}, $i = 1, \ldots, d$; j=1, …, J; k = 1,…,l
Initialization:
For all i, j
w_{ij} = generateRandomNumber $(-1, 1)$; //generate random numbers in interval $[-1, 1]$;
For all j, k
w_{jk} = generateRandomNumber $(-1, 1)$; //generate random numbers in interval $[-1, 1]$;
$\eta = 0.8$; //learning rate
$L_{max} = 5000$; $L = 0$;
$\varepsilon = 0.001$; //error bound

While $L \leq L_{max}$ and $\|\mathrm{E}\|_F > \varepsilon$, do
1. Calculate hidden layer outputs with Formula (1);
2. Calculate prediction values with Formula (2);
3. Calculate error: $\mathrm{E} = \frac{1}{2}\sum_{k=1}^{l}(\overline{y}_k - y_k)^2$;
4. Update weights with Formulas (3) and (4);
5. L++;

End While
Return w_{ij}, w_{jk};

3.2. Random Vector Functional Link Network (RVFL)

RVFL is a single hidden layer feedforward neural network (SLFNN) that features direct connections between the input layer and the output layer nodes. The input weights from the input layer to the hidden layer are generated randomly within specified intervals, while the output weights are computed analytically using either the least-square method or the pseudo-inverse method.

Formally, the RVFL model with L hidden nodes can be described as:

$$f(x_i) = \sum_{k=1}^{d} \beta_k x_{ik} + \sum_{k=d+1}^{k+L} \beta_k g(\alpha_k^T X + b_k),\ i = 1, \ldots, N. \tag{5}$$

The optimization problem of the standard RVFL model can be written as:

$$\min_{\beta \in R^{d+L}} \frac{1}{2}\|\beta\|^2 + \frac{1}{2}\lambda\|H\beta - Y\|^2, \tag{6}$$

where, $H = [H_1\ H_2]_{N\times(d+L)}$,

$$H_1 = \begin{vmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nd} \end{vmatrix}, \tag{7}$$

$$H_2 = \begin{vmatrix} \theta(\alpha_1 \cdot x_1 + \sigma_1) & \theta(\alpha_2 \cdot x_1 + \sigma_2) & \cdots & \theta(\alpha_L \cdot x_1 + \sigma_L) \\ \vdots & \vdots & \ddots & \vdots \\ \theta(\alpha_1 \cdot x_N + \sigma_1) & \theta(\alpha_2 \cdot x_N + \sigma_2) & \cdots & \theta(\alpha_L \cdot x_N + \sigma_L) \end{vmatrix}, \tag{8}$$

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{d+L} \end{bmatrix}_{(d+L)\times m}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}_{N\times m}.$$

where, $\beta_k = [\beta_{k1}, \beta_{k2}, \ldots, \beta_{km}]$ is the output weight vector connecting the kth input (hidden) node to the output nodes, $1 \leq k \leq d + L$, and $\alpha_j = [\alpha_{j1}, \alpha_{j2}, \ldots, \alpha_{jd}]$ is the weight vector connecting the jth hidden node to the input nodes, $1 \leq j \leq L$. Additionally, $x_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$ is the ith sample. For the target matrix, $y_i = [y_{i1}, y_{i2}, \ldots, y_{im}]$, $1 \leq i \leq N$. Moreover,

$\theta(\cdot)$ and σ_i are the non-constant activation function and the bias term of ith hidden node, respectively. λ is a regularization coefficient.

The optimal solution to the problem (6) when $\delta = \frac{1}{\lambda} = 0$ reads as follows:

$$\beta = H^{+}\ Y, \tag{9}$$

where H^{+} represents the Moore-Penrose generalized inverse of the matrix H. The regularization term is employed to avoid the over-fitting issue. The RVFL algorithm is described in Algorithm 2.

Algorithm 2: RVFL algorithm

Inputs: Input $X = \{x_1, \ldots, x_N\}$, $x_i \in R^d$ and output $Y = \{y_1, \ldots, y_N\}$, $y_i \in R^m$; Bias $b = [\beta_1, \ldots, \beta_L]$; The number of nodes in hidden layer L; the range of weights σ; regularization parameter λ.
Outputs: W_0.
1: Initialize: randomly selecting the initial weights in range of $[-\sigma, \sigma]$. $W_{initial} \in [-\sigma, \sigma]^{L \times d}$;
2: Calculate activation values of the hidden layer: $Z_{pre} = f_a(W_{initial}\ X + b)$, where f_a is an activation function;
3: Establish enhancement layer to implement direct link: $H = [Z_{pre},\ X]$;
4: Calculate output weights through regularized least squares method:
$W_0 = Y \times H^T \times (\lambda \mathrm{I} + \mathrm{H}H^T)^{-1}$;
5: Return W_0, there by $\hat{Y} = W_0 \times H$ can be calculated.

3.3. Stochastic Configuration Networks

Like RVFL in structure, the SCN is also a single hidden-layer neural network; however, the number of nodes in its hidden layer is not fixed but is dynamically increased through a supervisory mechanism [20]. Additionally, the input parameters between the input layer and the hidden layer are generated randomly within a specified domain, and the output parameters from the hidden layer are computed using least squares. The key to the SCN algorithm is the introduction of a supervisory mechanism that facilitates the addition of hidden nodes, helping to minimize the loss function and ultimately achieve convergence.

Given a set of N samples $\{(x_i,\ y_i),\ x_i \in R^d,\ y_i \in R^m,\ i = 1, \ldots, N\}$. An SCN learning network with $L-1$ hidden nodes can be represented as follows:

$$f_{L-1}(X) = \sum_{j=1}^{L-1} \beta_j g_j(w_j, b_j, X), \tag{10}$$

and the current residual error is:

$$\boldsymbol{e}_{L-1} = f - \boldsymbol{f}_{L-1} = [\boldsymbol{e}_{L-1,1}, \boldsymbol{e}_{L-1,2}, \ldots, \boldsymbol{e}_{L-1,m}]. \tag{11}$$

where, $g_j(\cdot)$ is the activation function of hidden node j, β_j is the output weight vector connecting the jth hidden node and w_j and b_j are input weights randomly generated in specified intervals $[-\lambda, +\lambda]^d$ and $[-\lambda, +\lambda]$, λ is a positive scalar.

While a new hidden node is added to the network, the algorithm is considered convergent if the resulting sequence of residual errors is monotonically decreasing.

$$f_L = f_{L-1} + \beta_L g_L(w_L \cdot X + b_L),\ e_L = f - f_L,\ e_L \le e_{L-1}. \tag{12}$$

A result obtained from reference [20] satisfies the Formula (12), where:

$$\beta_L = \frac{\langle e_{L-1}, g_L \rangle}{\|g_L\|^2}. \tag{13}$$

where,

$$\begin{cases} \langle e_{L-1}, g_L \rangle^2 \geq b_g^2 \delta_L, \\ 0 < \|g_L\| < b_g, \\ \delta_L = (1 - r - u_L)\|e_{L-1}\|^2, \\ r > 0,\ u_L > 0,\ 1 - r - u_L > 0, \\ \lim\limits_{L \to +\infty} u_L = 0. \end{cases} \tag{14}$$

The learning problem (11) can be converted to the following optimization task:

$$[\beta_1^*, \beta_2^*, \cdots, \beta_L^*] = \underset{\beta}{\operatorname{argmin}} \|f - \sum_{j=1}^{L} \beta_j g_j\|. \tag{15}$$

Its equivalent matrix form reads as:

$$\beta^* = \underset{\beta}{\operatorname{argmin}} \|G_L \beta - Y\|^2 = G_L^+ Y, \tag{16}$$

where $G_L = [g_1, g_2, \ldots, g_L]$, G_L^+ is the Moore-Penrose generalized inverse of the matrix G_L.

The SCN algorithm is detailed in Algorithm 3. For clarity, several formulas are introduced below.

$$g_L(w_L, b_L, X) = [g_L(w_L^T x_1 + b_L),\ g_L(w_L^T x_2 + b_L), \cdots, g_L(w_L^T x_N + b_L)]^T, \tag{17}$$

$$\beta_{L,q} = \frac{e_{L-1,q}(X)^T g_L(w_L, b_L, X)}{g_L(w_L, b_L, X)^T g_L(w_L, b_L, X)},\ q = 1, 2, \cdots, m, \tag{18}$$

and let

$$\begin{aligned} \xi_{L,q} &= e_{L-1,q} - e_{L,q} \\ &= \beta_{L,q} - (1 - r - u_L) e_{L-1,q}(X)^T e_{L-1,q}(X) \end{aligned}. \tag{19}$$

Algorithm 3: SCN algorithm

Input: $X = \{x_1, x_2, \ldots, x_N\}$, $x_i \in R^d$ and outputs $Y = \{y_1, y_2, \ldots, y_N\}$, $y_i \in R^m$; $L_{\max}$, ε, $T_{\max}$; $\Lambda = \{\lambda_{\min}: \Delta\lambda: \lambda_{\max}\}$;
Output: β^*, w^*, b^*

1. Initialize e_0: = $[y_1, y_2, \ldots, y_N]^T$, $0 < r < 1$, two empty sets Ω and W;
2. While $L \leq L\max$ AND $\|e_0\| > \varepsilon$, Do

Phase 1: Stochastic parameters configuration (steps 3–17):

3. For $\lambda \in \Lambda$
4. For $k = 1, 2, \ldots, Tmax$, Do
5. Randomly assign v_L and b_L from $[-\lambda, \lambda]^d$ and $[-\lambda, \lambda]$, respectively;
6. Calculate g_L, $\xi_{L,q}$ based on Equations (17) and (19), and $\mu_L = (1 - r)/(L + 1)$;
7. If min $\{\xi_{L,1}, \xi_{L,2}, \ldots, \xi_{L,m}\} \geq 0$
8. Save w_L and b_L in W, $\xi_L = \sum\limits_{q=1}^{m} \xi_{L,q}$ in Ω, respectively;
9. Else go back to step 4;
10. End If
11. End For (corresponds to Step 4)
12. If W is not empty
13. Find $w_L{}^*$, $b_L{}^*$ that maximize ξ_L in Ω, and set $G_L = [g_1{}^*, g_2{}^*, \ldots, g_L{}^*]$;
14. Break (go to step 18);
15. Else randomly take $\tau \in (0, 1 - r)$, renew r: = $r + \tau$, return to step 4;
16. End If
17. End For (corresponds to step 3)

Phase 2: Output weights computation:

18. Obtain $G_L = [g_1{}^*, g_2{}^*, \ldots, g_L{}^*]$;
19. Calculate $\beta^* = [\beta_1{}^*, \beta_2{}^*, \ldots, \beta_L{}^*]^T = G_L{}^+ \cdot Y$;
20. Calculate $e_L = e_{L-1} - \beta L^* g_L{}^*$;
21. Renew e_0: $= e_L$; L: $= L + 1$;
22. End While
23. Return $\beta_1{}^*, \beta_2{}^*, \ldots, \beta_L{}^*$, $w^* = [w_1{}^*, w_2{}^*, \ldots, w_L{}^*]$, $b = [b_1{}^*, b_2{}^*, \ldots, b_L{}^*]$.

3.4. Evaluation Metrics

Three metrics RMSE, MAPE, and R^2 are used to evaluate the performance of the proposed models, where RMSE is the root mean square error used to compare the prediction accuracy of different models, and MAPE is the mean absolute percentage error used to avoid positive and negative error cancellation on a prediction accuracy measurement. R^2 is the coefficient of determination, the larger the better. All results are computed based on the average values over 100 independent trials.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{\mathrm{N}}(y_i - \hat{y}_i)^2}, \tag{20}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right|, \tag{21}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y}_i)^2}, \tag{22}$$

where y_i is the actual value of the sample, $\overline{y}_i$ is the predicted value of the sample, $\hat{y}_i$ is the mean of the actual values of samples, and N is the total number of samples.

4. Experimental Results

To evaluate the performance of the proposed models, we tested the influence of different structures and initial weights on the model performance, stability, convergence, and prediction accuracy of different models. Four datasets-DB1, DB2, DB3, and DB4 were selected for benchmarking, where DB1 and DB2 are single-input-single-output datasets, but DB3 and DB4 have multiple inputs and a single output. All datasets were divided into 70% as training sets and 30% as test sets.

(1) DB1

$$f(x) = \sin(x) \cdot \cos(x).$$

(2) DB2

$$f(x) = x^2 - x + 13 - 5 \cdot \sin(2\pi x^2).$$

(3) DB3 (stock) is a standard dataset for regression prediction from the KEEL website. The data is the daily stock prices of 10 aerospace companies from January 1988 to October 1991. The task is to predict the price of the 10th company, given the prices of the rest of the companies.

(4) DB4 (concrete) is a standard dataset for regression prediction from the KEEL website. The concrete compressive strength is a nonlinear function of age and ingredients. The task is to predict the concrete compressive strength.

4.1. Network Structure

Network structure is critical for models based on neural networks. Different network architectures can significantly impact a model's performance. However, the network structure of a Stochastic Configuration Network (SCN) is determined by the algorithm itself and is not predetermined. Therefore, discussions regarding various network structures are relevant only to Backpropagation (BP) and Random Vector Functional Link (RVFL) networks.

For the Backpropagation (BP) model, we have selected a double-hidden-layer architecture as the foundational structure, with a primary focus on determining the optimal number of nodes in each layer. The testing process is conducted repeatedly, varying the

number of nodes in the two hidden layers, ensuring that the maximum number of nodes per hidden layer does not exceed 50. The results of the tests are assessed using the root mean square error (RMSE) metric. The findings are presented in Table 1.

Table 1. BP network structure on datasets.

Dataset	Number 1	Number 2	Range
DB1	5	30	[0, 50]
DB2	10	40	[0, 50]
DB3	50	10	[0, 50]
DB4	30	5	[0, 50]

Remark: Number 1: the number of nodes in the first hidden layer; Number 2: the number of nodes in the second hidden layer; Range: the range of the number of nodes.

For the RVFL, the objective of the test is to determine the optimal number of nodes in the hidden layer and the appropriate range of random weights based on the RMSE measure. The results are presented in Table 2.

Table 2. RVFL network structure on datasets.

Dataset	Number	Range-1	Range-2
DB1	80	[−40, 40]	[−50, 50]
DB2	100	[−20, 20]	[−50, 50]
DB3	100	[−5, 5]	[−50, 50]
DB4	100	[−10, 10]	[−50, 50]

Remark: Number: the number of nodes in the hidden layer; Range-1: the range of the random weights; Range-2: the maximum range of the permitted random weights.

4.2. Performance

We evaluated the performance of three models in conjunction with two widely recognized algorithms, Support Vector Regression (SVR) and Random Forest (RF), across four datasets. A total of 100 independent trials were conducted for each model and dataset. The evaluation metrics used include Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and R^2. The test results are presented in Table 3, while the fitting curves are illustrated in Figures 2–5. Based on the evaluation results, we found that the SCN model outperformed all other models across every dataset.

Table 3. Performance comparison among different models on four datasets.

Dataset	Model	RMSE		MAPE		R^2
		Mean ± (SD)	Best	Mean ± (SD)	Best	
DB1	SVR	0.0703 ± 0.0002	0.0700	30.1491 ± 0.9152	28.3137	0.0045
	RF	0.0085 ± 0.0007	0.0076	7.7704 ± 2.7491	3.7731	0.9994
	BP	0.0122 ± 0.0123	0.0115	37.6948 ± 23.3176	13.7574	0.9976
	RVFL	0.0484 ± 0.0006	0.0484	66.1857 ± 14.4562	39.9376	0.9824
	SCN	0.0074 ± 0.0024	0.0024	5.6983 ± 2.6124	0.8440	0.9995
DB2	SVR	0.1996 ± 0.0009	0.1982	21.4102 ± 0.1986	21.1481	0.0594
	RF	0.0064 ± 0.0005	0.0056	0.4135 ± 0.0150	0.3944	0.9993
	BP	0.0761 ± 0.0100	0.0601	6.2884 ± 1.9444	6.4121	0.9063
	RVFL	0.1084 ± 0.0005	0.1084	7.3193 ± 0.0225	7.3171	0.8197
	SCN	0.0008 ± 0.0039	0.0008	0.0050 ± 0.0008	0.0038	0.9999
DB3	SVR	0.0536 ± 0.0027	0.0492	2.7469 ± 0.1562	2.5131	0.9460
	RF	0.03968 ± 0.0036	0.0333	1.9413 ± 0.1555	1.6562	0.9649
	BP	0.0602 ± 0.0155	0.0386	10.4826 ± 1.9601	7.5725	0.9454
	RVFL	0.0687 ± 0.0001	0.0686	14.2589 ± 0.0001	14.2578	0.9170
	SCN	0.0388 ± 0.0012	0.0357	1.8707 ± 0.0483	1.7650	0.9681
DB4	SVR	0.0885 ± 0.0045	0.0823	20.8771 ± 1.0710	19.1571	0.8147
	RF	0.0961 ± 0.0080	0.0785	18.3751 ± 1.1851	16.4298	0.7839
	BP	0.0911 ± 0.0087	0.0796	19.5973 ± 1.7310	17.1850	0.8029
	RVFL	0.1932 ± 0.0001	0.1922	41.6601 ± 2.4803	33.1910	0.4144
	SCN	0.0870 ± 0.0060	0.0767	21.8238 ± 1.2261	19.3793	0.8135

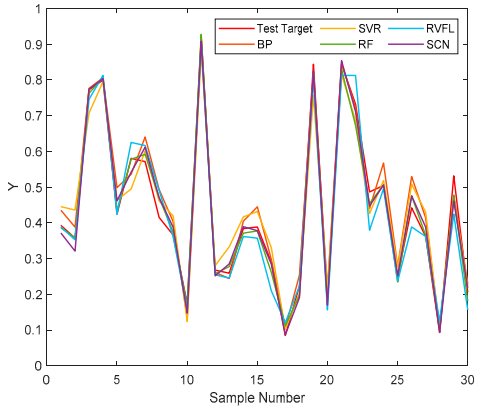

Figure 2. Fitting curves on DB3.

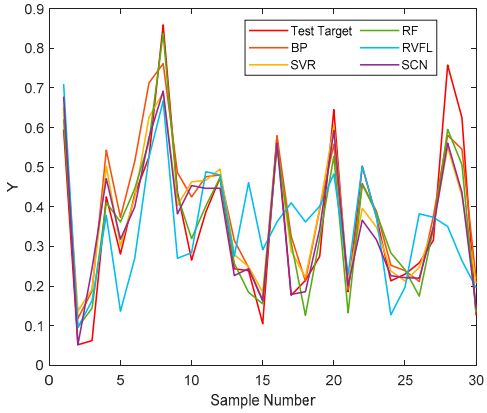

Figure 3. Fitting curves on DB4.

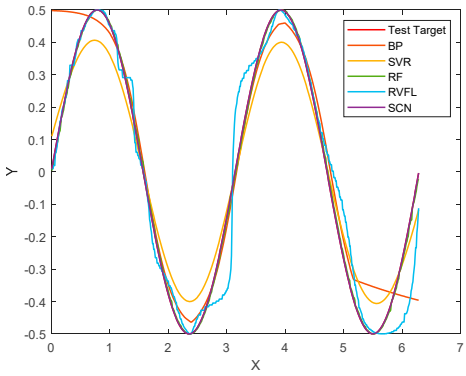

Figure 4. Fitting curves on DB1.

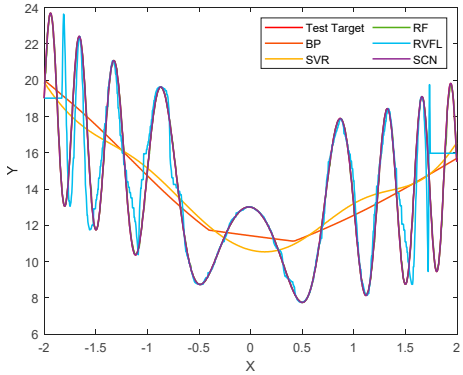

Figure 5. Fitting curves on DB2.

4.3. Convergence

Convergence is a crucial metric for assessing the performance of a model. The loss curves for the Backpropagation (BP) and the SCN were drawn in Figures 6–9 for four different data sets. It is important to note that the Random Vector Functional Link (RVFL) curves are not included, as this model does not involve an iterative process. As shown in Figures 6–9, the SCN exhibits a steeper decline in loss compared to the BP during iterations, ultimately resulting in a lower loss for the SCN across all data sets. This indicates that the SCN outperforms the BP in terms of model efficacy.

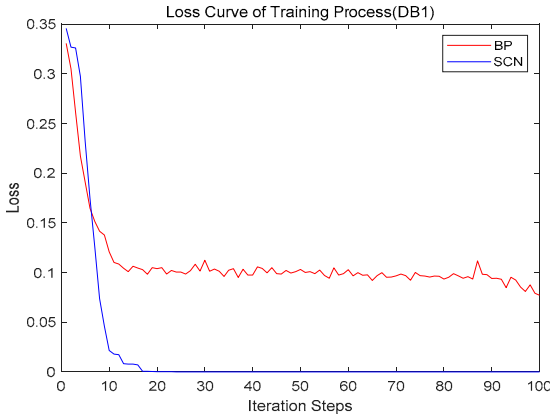

Figure 6. Loss curve of training (DB1).

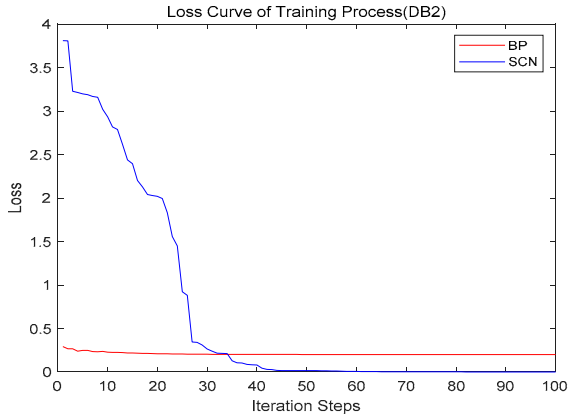

Figure 7. Loss curve of training (DB2).

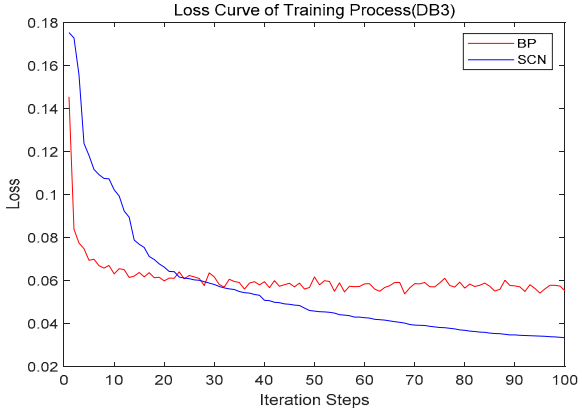

Figure 8. Loss curve of training (DB3).

Figure 9. Loss curve of training (DB4).

4.4. Stability

Stability is an important measure of an algorithm's performance. For the same reason, the RVFL is not inspected; only the BP and the SCN are discussed for model stability. Ibid., four data sets are used for testing, and 100 trials are executed. The RMSE is used as the evaluation measure. The computational results are shown in Figures 10–13. Compared with the BP, the RMSE curves of the SCN have smaller values and fluctuations, which indicates that the SCN has better stability and prediction accuracy with the algorithm proceeding.

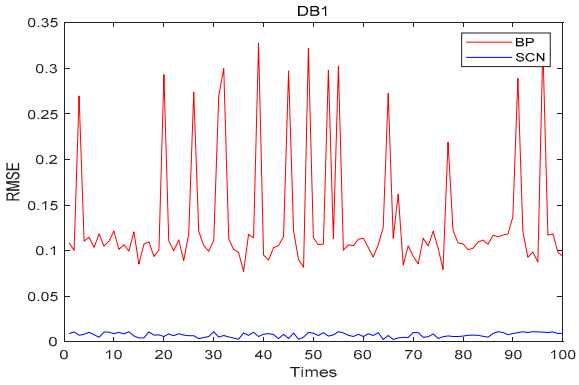

Figure 10. RMSE within 100 trials (DB1).

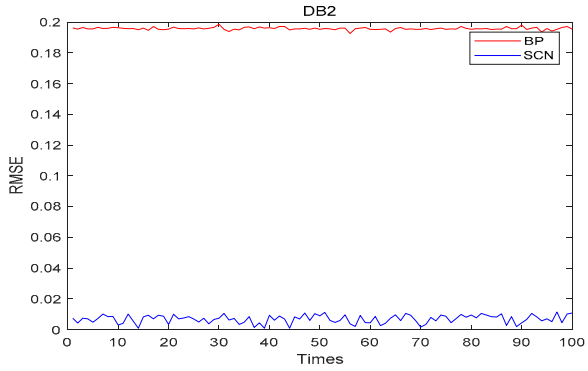

Figure 11. RMSE within 100 trials (DB2).

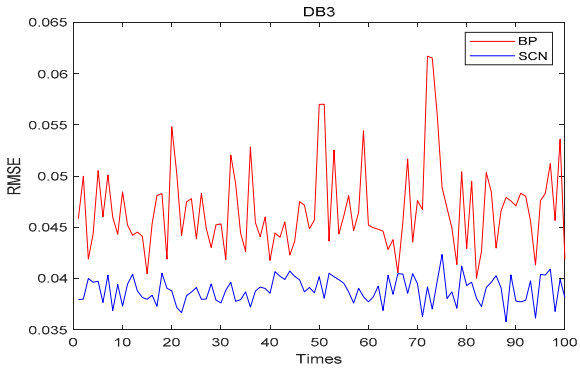

Figure 12. RMSE within 100 trials (DB3).

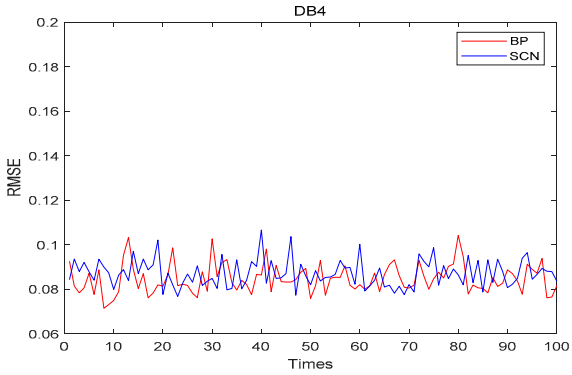

Figure 13. RMSE within 100 trials (DB4).

4.5. Performance with Various Initial Weights

The range of initial weights greatly influences the learning capability and generalization property. We select 11 intervals in the range of [−500,500] to find the best range of initial weights. The computational results (RMSE) are shown in Tables 4–7 and summarized in Table 8. The results show that the BP is the most sensitive to the changes in the range of weights; when and only when the weights are small enough, the BP can maintain stability. Specially, when the range of weights becomes large, the BP becomes extremely unstable. The RVFL performs a little better than the BP, but its accuracy is easily influenced by the range of weights. The SCN gains the best result with the test data, and the change in the accuracy is quite small compared with the other two algorithms.

Table 4. Performance with various weights on DB1.

Range of Weights	DB1		
	RVFL	BP	SCN
[−1, 1]	0.28148	0.14596	0.00586
[−5, 5]	0.13976	0.49169	0.00678
[−10, 10]	0.08151	0.56205	0.00748
[−20, 20]	0.05453	0.59831	0.00877
[−30, 30]	0.04694	0.61142	0.00837
[−40, 40]	0.04878	0.61564	0.00895
[−50, 50]	0.05661	0.56205	0.00897
[−60, 60]	0.16208	0.56205	0.00968
[−80, 80]	0.21759	0.60772	0.00995
[−100, 100]	0.16081	0.61610	0.00954
[−500, 500]	0.25634	0.61374	0.01169

Table 5. Performance with various weights on DB2.

Range of Weights	DB2		
	RVFL	**BP**	**SCN**
[−1, 1]	0.18356	0.20413	0.00474
[−5, 5]	0.12167	0.52918	0.00704
[−10, 10]	0.09808	0.54208	0.00817
[−20, 20]	0.09679	0.53724	0.00582
[−30, 30]	0.08942	0.57866	0.00853
[−40, 40]	0.09056	0.60472	0.00962
[−50, 50]	0.10127	0.56014	0.00937
[−60, 60]	0.14049	0.59212	0.00942
[−80, 80]	0.16943	0.56078	0.00966
[−100, 100]	0.16799	0.51073	0.01021
[−500, 500]	0.31912	0.54241	0.01235

Table 6. Performance with various weights on DB3.

Range of Weights	DB3		
	RVFL	**BP**	**SCN**
[−1, 1]	0.07325	0.04593	0.03709
[−5, 5]	0.04759	0.50602	0.07701
[−10, 10]	0.04287	0.57113	0.07612
[−20, 20]	0.07698	0.04606	0.07848
[−30, 30]	0.09410	0.55866	0.10125
[−40, 40]	0.10406	0.52318	0.09185
[−50, 50]	0.12092	0.53317	0.08473
[−60, 60]	0.12194	0.53291	0.08901
[−80, 80]	0.13108	0.53027	0.08757
[−100, 100]	0.12379	0.54040	0.08571
[−500, 500]	0.13252	0.55592	0.08917

Table 7. Performance with various weights on DB4.

Range of Weights	DB4		
	RVFL	**BP**	**SCN**
[−1, 1]	0.18387	0.09637	0.10241
[−5, 5]	0.13977	0.40924	0.10093
[−10, 10]	0.14436	0.57055	0.09312
[−20, 20]	0.17075	0.55035	0.10734
[−30, 30]	0.19714	0.56227	0.13241
[−40, 40]	0.20660	0.53162	0.12498
[−50, 50]	0.21126	0.55677	0.10198
[−60, 60]	0.21676	0.55245	0.10081
[−80, 80]	0.22001	0.56567	0.10748
[−100, 100]	0.23056	0.55152	0.09669
[−500, 500]	0.24482	0.50343	0.10091

Table 8. Algorithm stability with various initial weights.

Algorithm	DB1		DB2		DB3		DB4	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
RVFL	0.13695	0.08663	0.14349	0.06779	0.09719	0.03276	0.19690	0.03396
BP	0.54425	0.13744	0.52384	0.10962	0.44942	0.20026	0.49548	0.14010
SCN	0.00873	0.00160	0.00863	0.00213	0.08164	0.01646	0.10628	0.01192

4.6. Case Study

A real-world application of the unloading-time prediction will be addressed using the three models in this section. The data originates from the raw materials terminal of the studied steel plant, covering the years 2018 to 2020. Each ship-unloading record includes the following fields: BN (Berth Number), SN (Ship Name), RMN (cargo loaded), QRM (Quantity of Raw Materials), UD (Unloading Duration), ST (Starting Time of unloading), ET (End Time of unloading), SL1 (Ship Length), and SL2 (Storage Location in the stockyard). Additionally, there are other relevant fields not listed, such as berthing time, empty berthing time, and delay time, which can be appropriately incorporated into the calculations. Some unloading records are presented in Table 9.

Table 9. Ship-unloading records.

BN	SN	RMN	QRM	UD	ST	ET	SL1	SL2
7	FS8	Ore fines 1	23,585	37	5 November 2018 15:00	7 November 2018 4:00	173	OB24
1	MT	OYD-S/ONM-N	51,500	23	5 November 2018 9:00	6 November 2018 8:00	292	OA16
1	BY	OHY-S	90,000	34	7 November 2018 10:00	8 November 2018 20:00	320	OA18
2	ZC258	Ore fines 4	33,000	35	5 November 2018 9:00	6 November 2018 20:00	189	OB32
4	BXH6	dolomite	1100	3	5 November 2018 13:00	5 November 2018 16:00	50	OC23
5	JN9	Limestone	5005	18	6 November 2018 2:00	6 November 2018 20:00	110	OC11
2	ZC258	Ore fines 4	12,000	13	6 November 2018 9: 00	6 November 2018 22:00	189	OB16
2	BY	OHY-S	90,000	34	7 November 2018 3:30	8 November 2018 20:00	320	OA36
4	JN9	Limestone	5005	17	6 November 2018 9:00	7 November 2018 2:00	110	OC26

A total of 70% of the collected data was utilized as the training set, while the remaining 30% served as the test set. The computational results are presented in Figure 14, which displays the prediction outcomes for 30, 50, and 100 ships. The experimental findings indicate that the SCN model achieves an average prediction accuracy of 87%, the RVFL model 85%, and the BP model 76%. Additionally, the running times for all three models are in seconds.

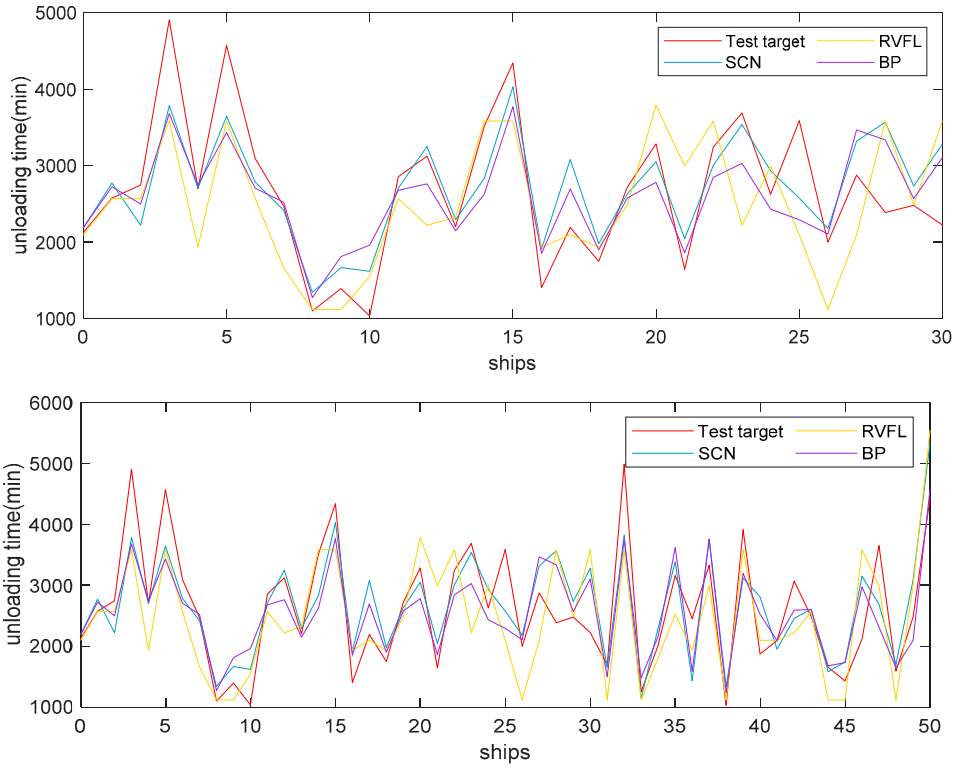

Figure 14. *Cont.*

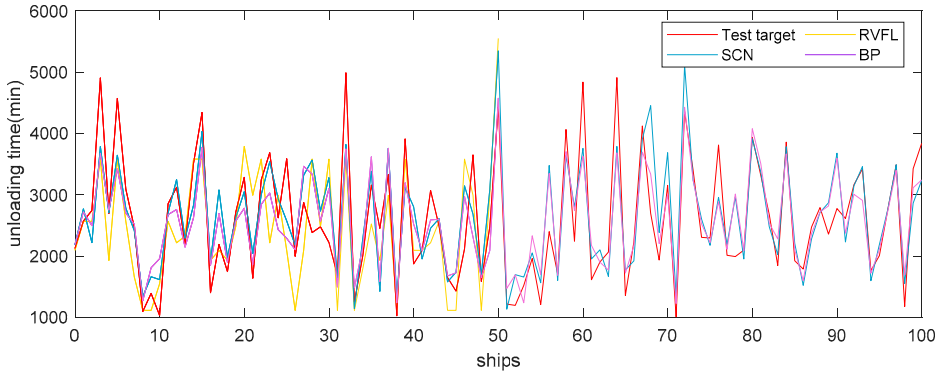

Figure 14. Unloading time predictions for 30, 50, and 100 ships.

5. Conclusions

The prediction of ship-unloading time is important for improving the efficiency of ship-unloading operations and reducing the delay cost for a steel plant. In this paper, three neural network-based methods-BP, RVFL, and SCN are proposed for the prediction purpose. The performance of these models is evaluated and tested using the benchmark data sets, which are from open databases and published literature. Finally, the three models are used for the prediction of ship unloading time. The experiment results indicate that the three models are very effective. The prediction accuracies of the SCN, RVFL, and BP are 87%, 85%, and 76%, respectively. That shows that our methods are effective and exhibit application potential.

In this study, we developed neural networks to predict the unloading time at the raw materials terminal of a steel plant. This method holds significant practical implications, not only for the operation of the steel plant's raw materials terminal but also for similar industries and industrial processes that are heavily reliant on upstream operation times. We must acknowledge the limitations of our method. While the results presented in this paper are satisfactory, every method has its constraints. As discussed in the algorithm testing section, each algorithm requires meticulous design and debugging to fulfill its intended purpose. Furthermore, varying data sets can also influence the outcomes. Future research will concentrate on integrating data processing technologies with forecasting methods to address practical industrial challenges.

Author Contributions: Conceptualization: Z.G. and D.L.; data curation: D.L.; formal analysis: D.W.; methodology: Z.G. and W.P.; experimental design and testing: D.L.; software: D.W. and Z.Y.; supervision: Z.G.; validation: Z.Y. and X.W.; writing original draft: Z.G.; writing review and editing: W.P. and D.W. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the Major Program of the National Natural Science Foundation of China (72192830, 72192831) and the 111 Project (B16009).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

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

References

1. Gao, Z.; Sun, D.; Zhao, R.; Dong, Y. Ship-unloading scheduling optimization for a steel plant. *Inf. Sci.* **2021**, *544*, 214–226. [CrossRef]
2. Gao, Z.; Zhang, M.; Zhang, L. Ship-unloading optimization with differential evolution. *Inf. Sci.* **2022**, *591*, 88–102. [CrossRef]
3. Zhang, X.F.; Chen, S.N.; Zhang, J.X. Adaptive sliding mode consensus control based on neural network for singular fractional order multi-agent systems. *Appl. Math. Comput.* **2022**, *434*, 127442. [CrossRef]
4. Zhang, X.F.; Chen, Y.Q. Admissibility and robust stabilization of continuous linear singular fractional order systems with the fractional order α: The $0 < \alpha < 1$ case. *ISA Trans.* **2018**, *82*, 42–50. [PubMed]

5. Yan, H.; Zhang, J.X.; Zhang, X.F. Injected infrared and visible image fusion via L1 decomposition model and guided filtering. *IEEE Trans. Comput. Imaging* **2022**, *8*, 162–173. [CrossRef]
6. Tang, L.; Meng, Y. Data analytics and optimization for smart industry. *Front. Eng. Manag.* **2021**, *8*, 157–171. [CrossRef]
7. Kong, D.; Zhu, J.; Duan, C.; Liu, L.; Chen, D. Bayesian linear regression for surface roughness prediction. *Mech. Syst. Signal Process.* **2020**, *142*, 106770. [CrossRef]
8. An, X.; Jiang, D.; Liu, C.; Zhao, M. Wind farm power prediction based on wavelet decomposition and chaotic time series. *Expert Syst. Appl.* **2011**, *38*, 11280–11285. [CrossRef]
9. Rojas, I.; Valenzuela, O.; Rojas, F.; Guillen, A.; Herrera, L.J.; Pomares, H.; Marquez, L.; Pasadas, M. Soft-computing techniques and ARMA model for time series prediction. *Neurocomputing* **2008**, *71*, 519–537. [CrossRef]
10. Kundrat, D.M. Conceptual model of the iron blast furnace considering thermodynamic and kinetic constraints on the process. *Metall. Trans. B* **1989**, *20*, 205–218. [CrossRef]
11. Gan, L.; Lai, C. A general viscosity model for molten blast furnace slag. *Metall. Mater. Trans. B* **2014**, *45*, 875–888. [CrossRef]
12. Xiao, Z.; Ye, S.J.; Zhong, B.; Sun, C.X. BP neural network with rough set for short term load forecasting. *Expert Syst. Appl.* **2009**, *36*, 273–279. [CrossRef]
13. Adelia, H.; Panakkat, A. A probabilistic neural network for earthquake magnitude prediction. *Neural Netw.* **2009**, *22*, 1018–1024. [CrossRef] [PubMed]
14. Zhang, Y.; Wang, X.; Yu, J.; Zeng, T.; Wang, J. Multistep prediction for earthworks unloading duration: A fuzzy Att-Seq2Seq network with optimal partitioning and multi-time granularity modeling. *Neural Comput. Appl.* **2023**, *35*, 21023–21042. [CrossRef]
15. Liang, S.K.; Wang, C.N. Modularized simulation for lot delivery time forecast in automatic material handling systems of 300 mm semiconductor manufacturing. *Int. J. Adv. Manuf. Technol.* **2005**, *26*, 645–652. [CrossRef]
16. Xu, Y.; Chen, Q.; Quan, X. Robust berth scheduling with uncertain vessel delay and handling time. *Ann. Oper. Res.* **2012**, *192*, 123–140. [CrossRef]
17. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
18. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
19. Malik, A.K.; Gao, R.; Ganaie, M.A.; Tanveer, M.; Suganthan, P.N. Random vector functional link network: Recent developments, applications, and future directions. *Appl. Soft Comput.* **2023**, *143*, 110377.
20. Wang, D.; Li, M. Stochastic configuration networks: Fundamentals and algorithms. *IEEE Trans. Cybern.* **2017**, *47*, 3466–3479. [CrossRef]
21. Dhingra, V.; Roy, D.; de Koster, R.B.M. A cooperative quay crane-based stochastic model to estimate vessel handling time. *Flex. Serv. Manuf. J.* **2017**, *29*, 97–124. [CrossRef]
22. Bish, E.K. A multiple-crane-constrained scheduling problem in a container terminal. *Eur. J. Oper. Res.* **2003**, *144*, 83–107. [CrossRef]
23. Al-Dhaheri, N.; Jebali, A.; Diabat, A. A simulation-based genetic algorithm approach for the quay crane scheduling under uncertainty. *Simul. Model. Pract. Theory* **2016**, *66*, 122–138. [CrossRef]
24. Sun, D.; Tang, L.; Baldacci, R. A benders decomposition-based framework for solving quay crane scheduling problems. *Eur. J. Oper. Res.* **2019**, *273*, 504–515. [CrossRef]
25. Sammarra, M.; Cordeau, J.F.; Laporte, G.; Monaco, M.F. A tabu search heuristic for the quay crane scheduling problem. *J. Sched.* **2007**, *10*, 327–336. [CrossRef]
26. Tang, L.; Zhao, J.; Liu, J. Modeling and solution of the joint quay crane and truck scheduling problem. *Eur. J. Oper. Res.* **2014**, *236*, 978–990. [CrossRef]
27. Kao, C.; Li, D.C.; Wu, C.; Tsai, C.C. Knowledge-based approach to the optimal dock arrangement. *Int. J. Syst. Sci.* **1990**, *21*, 2209–2215. [CrossRef]
28. Kao, C.; Wu, C.; Li, D.C.; Lai, C.Y. Scheduling ship discharging via knowledge transformed heuristic evaluation function. *Int. J. Syst. Sci.* **1992**, *23*, 631–639. [CrossRef]
29. Kao, C.; Lee, H.T. Coordinated dock operations: Integrating dock arrangement with ship discharging. *Comput. Ind.* **1996**, *28*, 113–122. [CrossRef]
30. Kim, K.H.; Moon, K.C. Berth scheduling by simulated annealing. *Transp. Res. Part B* **2003**, *37*, 541–560. [CrossRef]
31. Kim, B.I.; Chang, S.Y.; Chang, J.; Han, Y.; Koo, J.; Lim, K.; Shin, J.; Jeong, S.; Kwak, W. Scheduling of raw-material unloading from ships at a steelworks. *Prod. Plan. Control* **2011**, *22*, 389–402. [CrossRef]
32. Lee, D.H.; Cao, J.X.; Shi, Q.; Chen, J.H. A heuristic algorithm for yard truck scheduling and storage allocation problems. *Transp. Res. Part E* **2009**, *45*, 810–820. [CrossRef]
33. Tang, X.; Jin, J.G.; Shi, X. Stockyard storage space allocation in large iron ore terminals. *Comput. Ind. Eng.* **2002**, *164*, 107911. [CrossRef]
34. Li, S.; Tang, L. Improved tabu search algorithms for storage space allocation in integrated iron and steel plant. In Proceedings of the 2005 ICSC Congress on Computational Intelligence Methods and Applications, Istanbul, Turkey, 15–17 December 2005; p. 6. [CrossRef]
35. Kim, B.I.; Koo, J.; Park, B.S. A raw material storage yard allocation problem for a large-scale steelwork. *Int. J. Adv. Manuf. Technol.* **2009**, *41*, 880–884. [CrossRef]
36. Zhang, B.; Shin, Y.C. A probabilistic neural network for uncertainty prediction with applications to manufacturing process monitoring. *Appl. Soft Comput.* **2022**, *124*, 108995. [CrossRef]

37. Li, W.; Wang, X.X.; Han, H.G.; Qiao, J. A PLS-based pruning algorithm for simplified long-short term memory neural network in time series prediction. *Knowl.-Based Syst.* **2022**, *254*, 109608. [CrossRef]
38. Kosanoglu, F. Wind speed forecasting with a clustering-based deep learning model. *Appl. Sci.* **2022**, *12*, 13031. [CrossRef]
39. Hussein, A.M.; Elaziz, M.A.; Abdel Wahed, M.S.M.; Sillanpää, M. A new approach to predict the missing values of algae during water quality monitoring programs based on a hybrid moth search algorithm and the random vector functional link network. *J. Hydrol.* **2019**, *575*, 852–863. [CrossRef]
40. El-Said, E.M.S.; Elaziz, M.A.; Elsheikh, A.H. Machine learning algorithms for improving the prediction of air injection effect on the thermohydraulic performance of shell and tube heat exchanger. *Appl. Therm. Eng.* **2021**, *185*, 116471. [CrossRef]
41. Wang, W.; Wang, D. Prediction of component concentrations in sodium aluminate liquor using stochastic configuration networks. *Neural Comput. Appl.* **2020**, *32*, 13625–13638. [CrossRef]
42. Li, K.; Yang, C.; Wang, W.; Qiao, J. An improved stochastic configuration network for concentration prediction in wastewater treatment process. *Inf. Sci.* **2023**, *622*, 148–160. [CrossRef]