ISSN1027-5495. Functional Materials, 29, No.2 (2022), p. 279-290
doi:http://dx.doi.org/10.15407/fm29.02.279 

# Prediction of mechanical properties of hot rolled strip based on DBN and composite expectile regression

***Siyu Huang[1,2], Xiaoxia He[1,2], Xin Zhang[1,2], Weigang Li[3]***

[1]College of Science, Wuhan University of Science and Technology, Wuhan 430065, Hubei, China
[2]Hubei Province Key Laboratory of Systems Science in Metallurgical Process, Wuhan University of Science and Technology, Wuhan 430065, Hubei, China
[3]Engineering Research Center of Metallurgical Automation and Measurement Technologyof Ministry ofEducation,Wuhan University of Science and Technology, Wuhan 430081, Hubei, China.

*Received February 10, 2022*

In this paper, a DBN-$CER_{19}$ model (deep belief network and composite expectile regression) is proposed to predict the tensile strength of hot rolled strip. The model takes full advantage that the quadratic loss function used by expectile regression can be solved by standard gradient optimization algorithm, and that DBN can learn more abstract hidden layer information from the underlying data. At the same time, the model combines the CS algorithm (Cuckoo Search) to select the number of DBN hidden layer nodes to improve the network structure. In order to demonstrate the superiority of this method, we use root mean squared error (RMSE), mean absolute percentage error (MAPE) and mean absolute error (MAE)as measurement indicators for empirical analysis. The results of the proposed model on the test set are 21.4381, 2.5251and 13.5035, respectively. The empirical results show that the DBN-$CER_{19}$ model has higher prediction performance than previous models such as BP neural network(BPNN), quantile regression neural network(QRNN), expectile regression neural network(ERNN), and DBN.

**Keywords:** deep belief network; composite expectile regression; CS algorithm; mechanical properties of steel

**Прогноз механических свойств горячекатаной полосы на основе DBN и композитной ожидаемой регрессии**. *Сию Хуан, Сяося Хэ, Синь Чжан, Вейган Ли*

Предлагается модель DBN-CER19 (сеть глубокого доверия и составная регрессия ожиданий) для прогнозирования прочности на растяжение горячекатаной полосы. Модель использует то, что квадратичная функция потерь, используемая в ожидаемой регрессии, может быть решена с помощью стандартного алгоритма оптимизации градиента, и что DBN может получать более абстрактную информацию о скрытом слое из базовых данных. В то же время модель объединяет алгоритм CS (Cuckoo Search) для выбора количества узлов скрытого слоя DBN для улучшения структуры сети. Чтобы продемонстрировать превосходство этого метода, мы используем среднеквадратичную ошибку (RMSE), среднюю абсолютную процентную ошибку (MAPE) и среднюю абсолютную ошибку (MAE) в качестве показателей измерения для эмпирического анализа. Результаты предлагаемой модели на тестовом наборе составляют 21,4381, 2,5251 и 13,5035 соответственно. Эмпирические результаты показывают, что модель DBN-CER19 имеет более высокую эффективность прогнозирования, чем предыдущие модели, такие как нейронная сеть BP (BPNN), нейронная сеть квантильной регрессии (QRNN), нейронная сеть регрессии ожидания (ERNN) и DBN.

## 1. Introduction

The prediction of mechanical properties of hot rolled strip steel is a very complex metallurgical frontier technology that can provide a great impetus to long-term development of the steel industry. Since there is a complex and definite correspondence between the internal organization structure, strip composition and the corresponding production process that determines the mechanical properties of strip steel, it has been a difficult problem of concern for the steel metallurgical industry to establish a prediction model based on the quantitative relationship between these process parameters to further optimize the mechanical properties of steel.

In response to the modeling difficulties such as many influencing factors and complex interactions of the steel product performance index forecasting problem, researchers have carried out numerous related research works. Borisade S.G et al[1]used a polynomial regression mathematical model to predict the mechanical propertiesand experimentally demonstrated the authenticity of the model. Wu et al[2] used the MIV method to guide the selection of input neurons for the bayesian NN model to establish the mechanical performance prediction model. Khalaj G et al[3] used an artificial NN model to predict the ultimate tensile strength of APIX70 steel and achieved a high prediction accuracy. Chou et al[4] used TPSO algorithm to optimize BPNN for modeling and optimization of yield strength with high computational efficiency and strong robustness. Adel Saoudi et al[5]developed an artificial neural network (ANN)model to predict tensile and impact properties of APIX70 pipeline steel based upon its chemical composition. Hu[6]constructed a mechanical property prediction model for hot-rolled strip steel based on a deep feedforward NN and a convolutional neural network(CNN) fused with LeNet.5 and GoogLeNet. Xie Qian et al[7] designed a deep learning model to predict the mechanical properties of industrial steel plates based on the process parameters and composition of crude steel and applied it online in a real steel manufacturing plant.

Compared to the mean regression analysis method, the QR proposed by Koenker[8] provides a more comprehensive picture of the potential relationships between response variables and covariates when faced with asymmetrically distributed or larger scattered data. Tian et al[9] developed a generalized RBF neural network quantile regression model for predicting the mechanical properties of hot-rolled strip steel. Dadabada et al[10] proposed a QRNN based on particle swarm optimization (PSO)-trained to forecast volatility from financial time series. Zanget al[11] extracted the higher order features of the data based on CNN and further used gated recurrent NN for quantile regression modeling to predict the bus load values under different quantile conditions at any future moment. Li et al[12] proposed a long short-term memory NN (LSTM) quantile regression probability density prediction method to predict wind power, wind power interval and probability distribution of wind power.

For the detection loss function of quantile is not differentiable at zero, Newey and Powell[13] proposed the idea of expectile regression, where the detection loss function of expectile is everywhere differentiable, which can not only attenuate the influence of outliers on statistical inference and have the advantages of both mean regression and quantile regression, but also can characterize the whole distribution more comprehensively while having higher estimation efficiency. Hu et al[14] introduced an expectile-based VaR risk measure tool to obtain a risk measure based on a semi-parametric conditional autoregressive model, using expectile to estimate the model parameters, which makes fuller use of information and can better measure Chinese stock market risk. Jiang et al[15] proposed the ERNN model, which adds a NN structure to the expectile regression method, and since the expectile model uses an asymmetric square loss function, it can be easily estimated by standard gradient-based optimization algorithms and directly outputs the conditional expectation function, inheriting the ability of NN to handle nonlinear problems. Jiang[16] proposed a ERNN model with an increased penalty term, which can flexibly express the nonlinear relationship between the explanatory and response variables, and can completely examine the variation pattern of the conditional distribution of the response variables and comprehensively improve its predictive capability. Liu et al[17] proposed a class of semi-parametric variable coefficient composite expectile regression models, while establishing the large sample nature of their estimation. Numerical simulations show that the obtained estimates are more effective due to the consideration of information from multiple expectiles.

Currently, deep learning is successfully applied in various fields ranging from natural language recognition, environment perception to human behavior recognition with its efficient feature extraction ability and powerful modeling capability. As one of the widely used deep learning methods, DBN is able to automatically learn and extract essential features from massive data using its deep structure from bot-

tom-up. And it can largely solve the problems of shallow learning and improve the prediction performance by the feature extraction capability of Restricted Boltzmann Machine (RBM). Bao et al[18] proposed an improved deep learning structure based on DBN and support vector machines (SVR) for traffic prediction under severe weather. Yu et al[19] combined symbolic and classification rules with deep networks and proposed a knowledge-based deep trust network model (KBDBN) with good feature learning performance to predict the surface roughness of workpieces. Xing et al[20] proposed a temperature-based DBN in which temperature is considered in the learning process of DBN, and the introduction of temperature parameters improves the learning ability of DBN by changing the transfer function of neurons and activation conditions. Abdel-Zaher et al[21] proposed a computer-aided diagnosis scheme for breast cancer detection based on DBN, which provided an effective classification model for breast cancer.

When constructing DBN models, the choice of network structure and the number of hidden layers have a great impact on the prediction results of model. However, the network structures of DBN models are commonly obtained empirically or through time-consuming multiple tuning of parameters. In this paper, we study a DBN composite expectile regression model optimized with the CSalgorithm and use the model for regression prediction of steel mechanical properties. The model fully applies the advantage that the quadratic loss function used in the expectile regression can be used to solve the model with standard gradient optimization algorithms, and the feature that DBN can continuously learn from input data to discover more abstract hidden layer information. Further, the number of DBN hidden layer nodes is selected using the CS algorithm to improve the network structure. Finally, the DBN model with optimal structure is used for feature extraction and regression prediction of steel mechanics data. The empirical results show that the improved DBN composite expectile regression model proposed in this paper outperforms prediction models such asBPNN, QRNN, ERNN and DBN.

## 2. Composite Expectile regression and DBN model

### 2.1. Composite Expectile regression

Newey and Powell[13] established the asymmetric quadratic loss detection function based on the $l_2$ parametric in 1987 and proposed the idea of expectile regression: the overall distribution of the dependent variable is estimated by regressing the independent variables on the conditional expectile of the dependent variable to obtain a regression model based on all expectiles.It is a one-to-one mapping relationship with the quantile, which makes it inherently similar to the quantile and can give a complete portrayal of the conditional distribution. The computational advantage of the squared coefficient in the loss function makes the expectile have better properties than the quantile. Specifically, for a random variable $Y$, its $\theta$ -expectile can be obtained by optimizing the following expected loss function:

$$Expectile_Y(\theta) = arg\min_{\upsilon \in R} E\left[\rho_\theta(Y-\upsilon)\right], \quad (1)$$

where $\rho_\theta(u)$ is the asymmetric squared loss function:

$$\rho_\theta(u) = \begin{cases} \theta u^2, u \geq 0 \\ (1-\theta)u^2, u < 0, \end{cases} \quad (2)$$

where $\theta \in (0,1)$ is the expectile condition, which determines the degree of asymmetry of the loss function. $E[\cdot]$ denotes the expectation operator, and $u$ is the residual value of $Y - \upsilon$.

Consider the explanatory variable $X_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$, $p$ represents the dimensionality, and $Y$ is the response variable, assuming that the functional relationship between $X$ and $Y$ can be expressed as:

$$Y = f(X, \beta) + \varepsilon, \quad (3)$$

where $f(\cdot)$ is the nonlinear function, $\beta$ is the parameter to be estimated, and $\epsilon$ is the error term. The corresponding empirical loss function is defined as:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\rho_\theta\left(Y_i - f(x_i, \beta_\theta)\right), \quad (4)$$

where $N$ is the sample size, $f(x_i, \beta_\theta)$ is the conditional expectile value at different θ levels, and $\beta_\theta$ is the coefficient related to $\theta$, $\theta \in (0,1)$.

Liu and Zhou[22] proposed the composite expectile regression and proved that the composite expectile regression has good statistical properties relative to the expectile regression.

For a given $K$, let $\theta_k = \frac{k}{K+1}, k = 1, 2, \ldots, K$ and $c_{\theta_k}$ be the $\theta_k$-th expectile value of $\epsilon_i = Y_i - f(x_i, \beta_{\theta_k})$. The following composite expectile regression loss function can be obtained:

$$R(\beta) = \frac{1}{K}\sum_{k=1}^{K}L(\theta_k) = \quad (5)$$
$$= \frac{1}{KN}\sum_{k=1}^{K}\sum_{i=1}^{N}\rho_{\theta_k}\left(Y_i - f(x_i, \beta_{\theta_k}) - c_{\theta_k}\right).$$

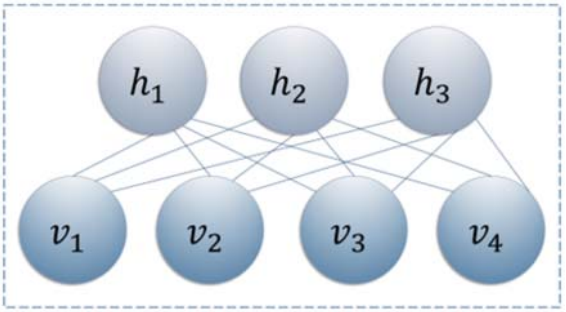


Fig.1. A constrained Boltzmann machine with 7 variables

We can solve the parameter problem by minimizing.

### *2.2. Deep Belief Network*

Over the past few decades, deep neural networks(DNN), inspired by the structure of the brain, have been used in a wide range of practical problems. The DBN proposed by Hinton et al[23], which consists of a stack of multiple RBMs, combines the powerful learning capability of DNN with the interpretability of inference-based processes and is considered one of the most effective DNN.

A DBN is a feed-forward NN with a deep structure, which consists of an input layer, multiple hidden layers and finally an output layer. The input layer receives the input data and transmits the data to the hidden layers to complete the learning process. The hidden layers are created by several stacked RBMs, thus forming a network capable of capturing underlying patterns and invariants directly from the input data.

RBM is an undirected graph model with a bipartite graph structure, and a typical RBM model topology is shown in Fig. 1.

The observable and hidden layers are used to represent the observable and hidden variables in the RBM . The observable and hidden layers are fully connected by symmetric undirected weights, but there is no connection between nodes in the same layer. The RBM is an energy-based model with weights and biases that determine the energy of the joint configuration of the observable and hidden layer nodes $E(\boldsymbol{v},\boldsymbol{h})$ :

$$E\left(\boldsymbol{v},\boldsymbol{h}\,|_{\dagger}\right) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i=1}^{m}\sum_{j=1}^{n} v_i w_{ij} h_j = \\ = -\boldsymbol{a}^T\boldsymbol{v} - \boldsymbol{b}^T\boldsymbol{h} - \boldsymbol{v}^T\boldsymbol{W}\boldsymbol{h}, \tag{6}$$

where

$$\boldsymbol{v} = \left[v_1, v_2, \ldots, v_m\right],$$

$$\boldsymbol{h} = \left[h_1, h_2, \ldots, h_n\right],$$

$$\boldsymbol{W} = \begin{bmatrix} w_{11}\, w_{12} \cdots w_{1n} \\ w_{21}\, w_{22} \cdots w_{2n} \\ \vdots \quad \vdots \quad \cdots \quad \vdots \\ w_{m1}\, w_{m2} \cdots w_{mn} \end{bmatrix},$$

$$\boldsymbol{a} = \left[a_1, a_2, \ldots, a_m\right], \quad \boldsymbol{b} = \left[b_1, b_2, \ldots, b_n\right], \tag{7}$$

$$\dagger = \begin{bmatrix} a_1 \quad a_2 \quad \ldots a_m \\ b_1 \quad b_2 \quad \ldots b_n \\ w_{11}\, w_{12} \cdots w_{1n} \\ w_{21}\, w_{22} \cdots w_{2n} \\ \vdots \quad \vdots \quad \cdots \quad \vdots \\ w_{m1}\, w_{m2} \cdots w_{mn} \end{bmatrix}. \tag{8}$$

Assume that $\boldsymbol{I} = \{\boldsymbol{a},\boldsymbol{b},\boldsymbol{W}\}$ is the RBM model parameters, $\boldsymbol{v}$ is the output of the visible layer, $\boldsymbol{h}$ is the output of the hidden layer, $\boldsymbol{W}$ is the weight matrix between the hidden layer and the visible layer, $\boldsymbol{a}$ is the bias vector of the visible layer, and $\boldsymbol{b}$ is the bias vector of the hidden layer. $v_i$ is the output value of the $i$-th visible neuron, $h_j$ is the output value of the $j$-th hidden neuron, $a_i$ is the bias of the $i$-th visible neuron, $b_i$ is the bias of the $j$-th hidden neuron, and $w_{ij}$ is the weight of the edge between the $i$-th visible neuron and the j-th hidden neuron in the RBM. $m$ is the total number of neurons in the visible layer and $n$ is the total number of neurons in the hidden layer.

The joint probability distribution $p(\boldsymbol{v},\boldsymbol{h})$ of the RBM is defined as:

$$p\left(\mathbf{v},\mathbf{h}\,|_{\dagger}\right) = \frac{1}{Z_\beta} e^{-E\left(\boldsymbol{v},\boldsymbol{h}|_{\dagger}\right)} = \frac{1}{Z_\beta} e^{\boldsymbol{a}^T\boldsymbol{v}} e^{\boldsymbol{b}^T\boldsymbol{h}} e^{\boldsymbol{v}^T\boldsymbol{W}\boldsymbol{h}}, \tag{9}$$

$$Z = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E\left(\boldsymbol{v},\boldsymbol{h}|_{\dagger}\right)}, \tag{10}$$

where $Z$ is the normalization factor, often referred to as the partition function. The conditional probability of each observable and hidden variable in a RBM is described as:

$$p\left(h_j = \mathbf{1} \mid \boldsymbol{v}\right) = \sigma\left(b_i + \sum_{i=1}^{m} v_i w_{ij}\right), \tag{11}$$

$$p\left(v_i = \mathbf{1} \mid \boldsymbol{h}\right) = \sigma\left(a_i + \sum_{j=1}^{n} h_j w_{ji}\right), \tag{12}$$

where $p\left(h_j = \mathbf{1} \mid \boldsymbol{v}\right)$ is the activation probability of the $j$-th hidden layer neuron, $p\left(v_i = \mathbf{1} \mid \boldsymbol{h}\right)$ is the activation probability of the $i$-th visible neuron, $\sigma$ is the logistic function, and $\sigma(x) = \mathbf{1} / \left(\mathbf{1} + e^{-x}\right)$.

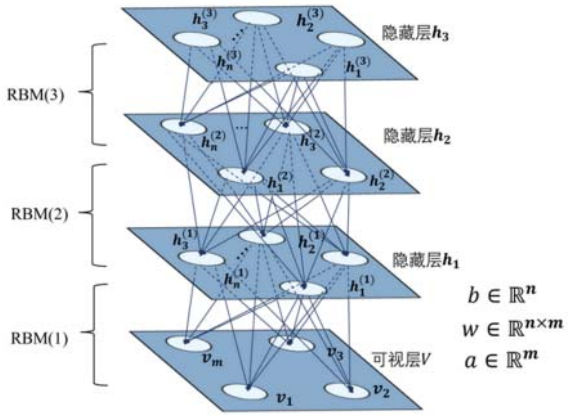


Fig. 2. A typical DBN model structure

The DBN consists of multiple RBMs stacked from bottom to top, with the hidden layer of the $l$ -th RBM serving as the observable layer of the $(l+1)$ -st RBM. A typical DBN structure model is shown in Fig. 2.

The joint probability of all variables in DBN can be decomposed as:

$$p\left(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \cdots, \boldsymbol{h}^{(L)}\right) = \\ = p\left(\boldsymbol{v} \mid \boldsymbol{h}^{(1)}\right)\left(\prod_{l=1}^{L-2} p\left(\boldsymbol{h}^{(l)} \mid \boldsymbol{h}^{(l+1)}\right)\right) p\left(\boldsymbol{h}^{(L-1)}, \boldsymbol{h}^{(L)}\right) \\ = \left(\prod_{l=1}^{L-1} p\left(\boldsymbol{h}^{(l)} \mid \boldsymbol{h}^{(l+1)}\right)\right) p\left(\boldsymbol{h}^{(L-1)}, \boldsymbol{h}^{(L)}\right), \quad (13)$$

where $p\left(\boldsymbol{h}^{(l)} \mid \boldsymbol{h}^{(l+1)}\right)$ is a Sigmoid-type conditional probability distribution:

$$p\left(\boldsymbol{h}^{(l)} \mid \boldsymbol{h}^{(l+1)}\right) = \sigma\left(\boldsymbol{a}^{l} + \boldsymbol{W}^{(l+1)} \boldsymbol{h}^{(l+1)}\right), \quad (14)$$

where $\sigma(\cdot)$ is the Logistic function, $\boldsymbol{a}^{l}$ is the bias parameter, and $\boldsymbol{W}^{(l+1)}$ is the weight parameter.

### 2.3. Composite Expectile Regression Based on DBN

DBN can continuously learn from input data to discover more abstract hidden layer information, while the layer-by-layer pre-training of RBM has the feature of producing very good initial values of parameters. In order to fully apply the quadratic loss function used in the expectile regression, which can be solved by the standard gradient optimization algorithm, and the advantage that multiple expectile regressions can improve the efficiency of parameter estimation, we propose a prediction model(DBN-CER) combining DBN and composite expectile regression, and use DBN as the nonlinear model at $f(\cdot)$ in

$$R(\beta) = \frac{1}{K}\sum_{k=1}^{K} L(\theta_k) = \\ = \frac{1}{KN}\sum_{k=1}^{K}\sum_{i=1}^{N} \rho_{\theta_k}\left(Y_i - g\left(x_i, \beta_{\theta_k}\right) - c_{\theta_k}\right), \quad (15)$$

where $\dagger = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{W}\}$ is the parameter of the model, $N$ is the number of samples, and for a given $K$, let $\theta_k = \frac{k}{K+1}, k = 1, 2, \ldots, K$, $\theta \in (0,1)$, $c_{\theta_k}$ be the $\theta_k$ -thexpectile value of $\epsilon_{ij} = Y_i - f\left(x_i, \beta_{ij}\right)$. $g\left(x_i, \beta_{\theta_k}\right)$ is the conditional expectile value of the DBN prediction model at different $\theta$ levels.

## 3. Solution Algorithms

The design of the network structure is a key issue in the application of DBN.Here we use a swarm intelligence optimization algorithm-CS algorithm proposed by Prof. Xin-She Yang and S. Deb[24] at the University of Cambridge, UK, to find the optimal number of hidden layer units.The training process of DBN prediction model can be divided into two phases: layer-by-layer pre-training and fine-tuning.First, the underlying RBM is unsupervised pre-trained by the Contrastive Divergence(CD) algorithm to obtain highly abstract reconfiguration features and better initial values of parameters. The learned reconstructed features are then used as the input of the DBN prediction network, and the BP algorithm is used to perform supervised fine-tuning on the labeled samples, and then a parameter-sharing DBN prediction model is obtained to fit the output.

### 3.1. Unsupervised pre-training and fine-tuning of DBN

The training process of DBN prediction model is divided into two phases: pre-training and fine-tuning.

The first phase is the forward-stacking RBM learning process: unsupervised layer-by-layer learning is used to initialize the parameters of the NN. As shown in Fig. 3, in order to adjust parameters such as weights and biases of the network, greedy pre-training by unsupervised means is required in the first phase, and the unsupervised learned model parameters are used as the initial values of the supervised learning model to provide a priori knowledge of the input data for the subsequent learning process in phase 2.

In 2002, Hinton[25] proposed the CD algorithm, a fast learning algorithm for RBM. In the CD algorithm, the probabilities of all hidden variables are first calculated using (11), and based on this distribution, a hidden vector is sampled from it by the Gibbs sampling

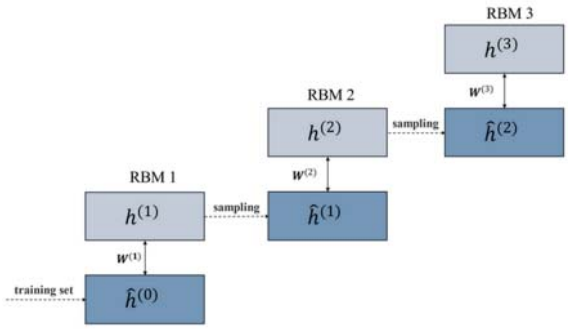


Fig. 3. Layer-by-layer pre-training process for DBN

method. Then, based on this hidden vector, the probabilities of the observable variables are calculated using (12), and based on this distribution, the input vector of the visible layer is reconstructed by Gibbs sampling again. This mapping and reconstruction process between the visible and hidden layers is repeatedly performed, and the values of weights a, b and W are continuously updated during each reconstruction. The update criteria for each parameter are:

$$\Delta W_{ij} = e\left(\langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{recon}\right), \quad (16)$$

$$\Delta a_i = e\left(\langle v_i\rangle_{data} - \langle v_i\rangle_{recon}\right), \quad (17)$$

$$\Delta b_j = e\left(\langle h_j\rangle_{data} - \langle h_j\rangle_{recon}\right), \quad (18)$$

where $e$ is the learning rate and $e \in (0,1)$. $\langle\cdot\rangle_{data}$ denotes the mathematical expectation defined by the input dataset and $\langle\cdot\rangle_{recon}$ denotes the mathematical expectation defined by the model after a one-step reconstruction.

Phase 2 is the backward fine-tuning learning process of DBN: global fine-tuning of network parameters using the back-propagation algorithm.As in Fig. 4, after each RBM is trained layer by layer, a regression layer is added to the top of the resulting DBN as the output layer, and then the weights of the entire network are fine-tuned using the BP algorithm.Starting from the last layer of the DBN, the model parameters are fine-tuned to the previous layer using known labels, and then the model parameters are further optimized globally.For the regression problem, the top layer of the network is generally chosen as the regression layer with only one node. This avoids the disadvantages of BP algorithms that tend to fall into long training timesandlocal optimum due to the random initialization of the weight parameters, and not only improves the performance of the modelwhile speeding up the convergenceof the tuning phase.

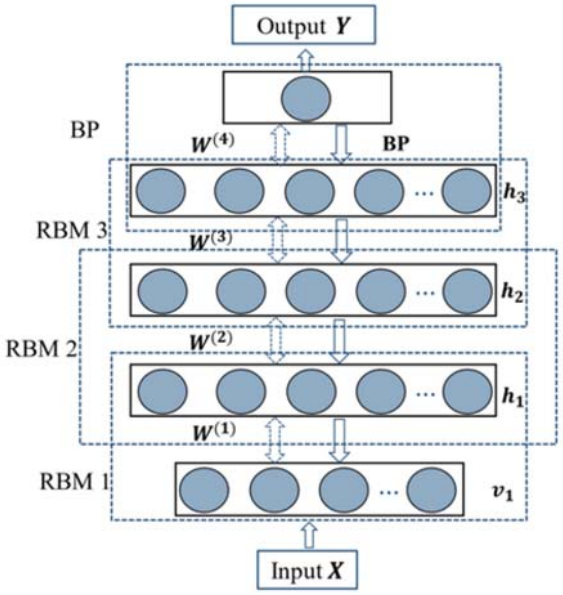


Fig. 4. Pre-training and fine-tuning of DBN

### *3.2. CS optimized DBN structure*

The research team of Bengio[26] proved after extensive experiments:DBNcontaining multiple hidden layers are more effective than single hidden layer networks.In another paper, Bengio and Larochelle[27] further elaborated on this issue based on a series of experimental results.The results show that the fitting effect of the DNN model increases as the number of hidden layers increases. However, when the number of hidden layers is increased to more than 4, the fit of the model weakens and the generalization performance decreases. The unsupervised pre-training process has also been shown to contribute to the optimization of deep structure.Therefore, we choose the DBN structure with 3 RBMs here. The number of hidden layer neurons also affects the performance and fitting ability of the model.If the number of hidden layer neurons is too small, the model has difficulty learning useful features and cannot fit the data effectively. If the number of hidden layer neurons is too large, the model tends to be overfitted.Moreover, the more the number of hidden layer neurons, the more parameters the model needs to calculate, leading to a longer training process. Therefore, selecting the right number of hidden layer neurons is a key step in building the network model.

Intelligent optimization algorithms are a class of meta-heuristic optimization algorithms inspired by biota in nature and summarized with powerful computational, optimization-seeking capabilities.With its universal applicability it is widely used for network structure design and parameter optimization problems in NN.Among the many intelligent optimization

algorithms, the CS algorithmis widely used in practical problems because of its simple structure, easy implementation, few parametersand good search performance (easy convergence to global optimum and fast convergence, excellent search path).Chen et al[28]proposed a proton exchange membrane fuel cell (PEMFC) degradation prediction method based on wavelet NN and CS algorithm, which greatly improved the prediction accuracy of PEMFC.Preeti Malhotra et al[29] used the CS algorithm to optimize a hybrid model combining discrete cosine transform (DCT) and principal component analysis (PCA) to improve the accuracy of face recognition and further optimize the face recognition system.Yang et al[30]combined the CS algorithm with finite element analysis to propose a new method for optimizing the positioning layout of sheet metal fixtures based on the "N-2-1" positioning principle.

The CS algorithm is based on the parasitic incubation behavior of cuckoos and simulates the biological behavior of cuckoos searching for the optimal host nest to lay their eggs, this nest representing the new or better solution. In addition, to simulate the cuckoo's nest search, the algorithm improves the optimal location based on the Lévy flight path principle rather than a simple isotropic random wander.The CS algorithmsets up three ideal states as follows:

(1) Cuckoos lay one egg per generation at a time and hatch randomly in 1 nest.

(2) A greedy selection strategy. Cuckoos always retain the best nest as a host until a better nest is found, and the nest most suitable for egg-laying will be retained for the next generation.

(3) The number of available nests is fixed for each generation and the probability of a host bird finding a parasitic egg is set to $Pa \in [0,1]$.

The CS algorithm first defines the objective function and initializes this function to generate initial positions randomly, setting parameters such as maximum discovery probability *Pa*, problem dimension, population size, and maximum number of iterations. In the search processes, the cuckoo first generates a new location based on the current location by randomly swimming in a Lévy flight, and selects a better search location by greedy means according to the fitness function. To increase the diversity of the search, some of the newly generated positions are discarded according to a certain probability $Pa$. The same number of new positions as the discarded positions are regenerated using the preference random tour, and the better search positions are preserved according to the fitness value evaluation, completing a round of the search for superiority.

On this consideration, the cuckoo's path of discovery is as follows,

$$X_i^{t+1} = X_i^t + \alpha \oplus L(\lambda), \qquad (19)$$

where $i \in [1,2,\ldots,n]$, $n$ is the number of cuckoo nests. $X_i^t$ denotes the position vector of the $i$-th nest in generation $t$. $X_i^{t+1}$ denotes the position vector of the $i$-th nest in generation $t+1$ based on $X_i^t$. $\alpha$ is the control step of the random search range, $\alpha = \alpha_0 (X_i^t - X_b)$, $\alpha_0 = 0.01$, $X_b$ is the current optimized best position. $\oplus$ is the point-to-point multiplication. $L(\lambda)$ is the random search path and its step size obeys the Lévy distribution.

In Levy flight, short and long steps are alternated. Levy flight prefers a random wander search strategy that mixes variational and crossover operations throughto produce each new position of the random wander.

$$X_i^{t+1} = X_i^t + r \cdot (X_j^t - X_k^t), \qquad (20)$$

where $r$ denotes the scaling factor and $r = rand(0,1)$. $X_j^t$ and $X_k^t$ are the randomly selected nest locations, respectively.

In the CS algorithm, a suitable number of hidden layer nodes are selected for the network by assigning the number of neurons in each hidden layer to the optimal bird nest location. The CS algorithm uses the Levy flight principle to update the candidate populations, which expands the search range and increases the diversity of populations through random tour and preference random tour search strategies. This effectively avoids falling into local minima on the DBN-CER gradient curve, while effectively balancing the global search and local search for the optimal solution.Meanwhile, in the process of global search for optimal solutions, the CS algorithm has a faster speed, which can accelerate the gradient descent of DBN-CER and quickly obtain the approximate global optimal solution. For the DBN model with three hidden layers selected in this paper, each layer has $m_1, m_2, m_3$ neurons respectively, and the position of each cuckoo in the CS is set as a three-dimensional vector $X(m_1, m_2, m_3)$.

### *3.3. Algorithm flow*

The CS algorithm was used to select the number of DBN hidden layer network nodes and optimize the DBN-CER model structure. The flow chart is shown in Fig.5. The specific steps are as follows.

Step 1: The data set is divided into a training set and a test set with a certain ratio.

Step 2: Set parameters such as the number of nests, problem dimension, maximum discovery probability *Pa*, and maximum number of iterations.

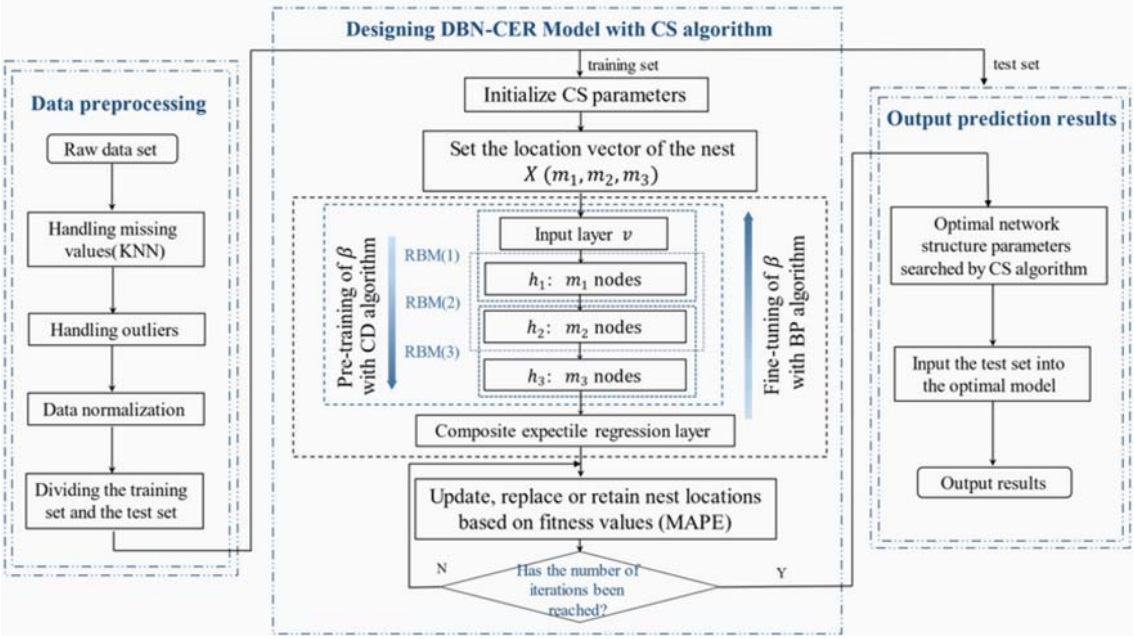


Fig. 5. Optimizing DBN-CER model with the CS algorithm

Step 3: The number of nodes in the DBN-CER hidden layer is encoded and the position vector is set to $X(m_1, m_2, m_3)$, which is searched using the CS algorithm.

Step 4: MAPE of the DBN-CER model was used to construct the objective function, fitness. where the training process of the DBN consisted of two phases, pre-training and fine-tuning. The pre-training phase uses the CD algorithm to train each layer of the RBM in a layer-by-layer manner. The fine-tuning phase uses the BP algorithm to propagate backwards the error between the actual output and the true value layer by layer to optimize the parameters of the whole DBN model. The objective function is calculated and the initial optimal value is saved as the current global optimum.

Step 5: Updating the nest positions by (19) and (20), discarding some of the newly generated positions according to a certain probability *Pa*, calculating the fitness value and comparing it with the optimal value in step 3, replacing the global optimal weight if the contemporary optimal value is better than the global optimal weight, and recording the final optimal nest position left behind.

Step 6: If the optimal nest position obtained reaches the set number of iterations, the value of the optimal position is assigned to the number of nodes of the DBN-CER model, otherwise return to step 5.

Step 7: Training of the DBN-CER model optimized with the CS algorithm.

Step 8: Input test samples and output regression results.

## 4. Real data analysis

### 4.1. Data sources and pre-processing

The experiments were conducted on a 64-bit 8-core R7-4800U, benchmark frequency of 1.80 GHz and a computer configuration with 16 GB of RAM. Both python 3.7 and Tensorflow 2.2 were used.

The research data for this paper is derived from a hot-rolled strip production line at a steel company and includes twenty influencing factors such as process parameter factors: FT (heating furnace temperature), FET (final rolling inlet temperature), FDH (processed thickness), CT (coiling temperature), RT (roughing exit temperature) and chemical factors: Mn (manganese), Si (silicon), Ti (titanium), V (vanadium), NbN (niobium oxide), Nbc (niobium carbide), Cs (carbon residue).The data was divided into a training set and a test set in the ratio of 7:3. The K nearest neighbour algorithm (KNN) was used to process the collected data for missing values. The R language lm() function was used for outlier detection. Finally, data normalization is performed by equation, which leads to the final modeling data.

$$\widehat{x_i} = \frac{x_i - x_{min}}{x_{max} - x_{min}}. \qquad (21)$$

To evaluate the predictive performance of each model, three performance metrics were used to measure the performance of the models, includingRMSE, MAPE and MAE.

Table 2 results of different $k$

| $k$ | Train | | | Test | | |
|---|---|---|---|---|---|---|
| | RMSE | MAPE | MAE | RMSE | MAPE | MAE |
| 1 | 20.5264 | 2.4580 | 12.9981 | 21.6788 | 2.5632 | 13.7474 |
| 5 | 20.5184 | 2.4563 | 12.9831 | 21.5830 | 2.5445 | 13.6300 |
| 9 | 20.4896 | 2.4531 | 12.9810 | 21.4967 | 2.5313 | 13.5429 |
| 19 | 20.4295 | 2.4478 | 12.9499 | 21.4381 | 2.5251 | 13.5035 |

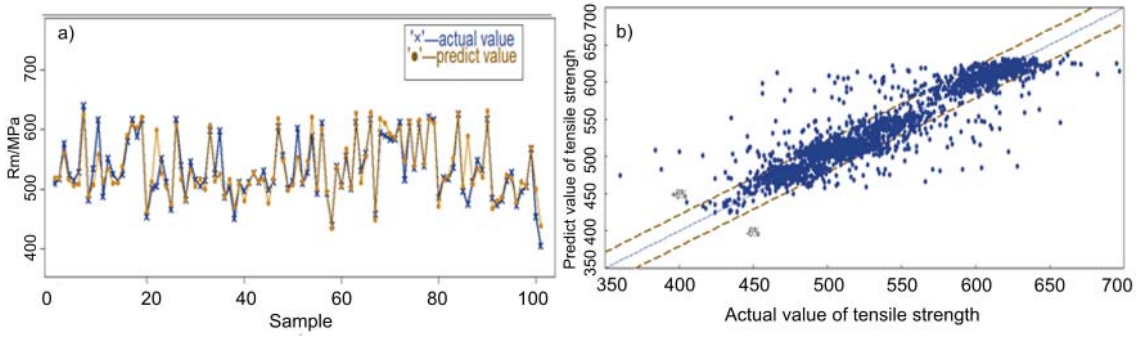


Fig. 6. DBN-CER19 training results (a) Fitting plots of the true and predicted values of the training set (b) Error intervals of the true and predicted values of the training set.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(g_i - y_i)^2}, \quad (22)$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|g_i - y_i|}{y_i}\right), \quad (23)$$

$$MAE = \sum_{i=1}^{n}|g_i - y_i|, \quad (24)$$

where $y$ represents the sample value, $g$ represents the predicted value and n is the sample size.

### 4.2. Experimental results and analysis

*4.2.1. Parameters selection*

We take the tensile strength (TS) of the steel mechanical properties index as the dependent variable and the twenty influencing factors studied as the independent variables, and the optimal parameters of the DBN prediction model obtained after the CS search are shown in Table 1.

Using these parameter values for training the model, the following table shows the composite results for the training and test sets when $k$ =1, 5, 9, 19, respectively.

The results show that the accuracy of the training and test sets graduallyimproves as $k$ increases, and for the proposed DBN-CER$_k$ model, the model predicts the most accurate results when $k$ is 19, i.e., the efficiency of prediction can be improved by using multiple expectile regressions simultaneously.

*4.2.2. Analysis of results*

We substitute the data set into the trained DBN-CER$_{19}$ model and obtain the prediction results as shown in Figures 6 and 7. In order to highlight the prediction effect of the training and test sets more intuitively, we randomly select 100 samples from the training and test sets respectively, and compare the predicted values with the corresponding actual values. From Fig. 6(a) and Fig. 7(a), it can be seen intuitively that the predicted values represented by the yellow dots cover a large area of the real values represented by the blue dots, and it can be seen that the fitting effect of this model is significant.

Fig. 6(b) and Fig. 7(b) show the true values of the training and test sets as the horizontal coordinates and their predicted values as the vertical coordinates, respectively. The two side lines represent the range of ±6% interval. The closer the point to the middle diagonal line, the better the prediction of the model is. The graph shows that the vast majority of points are distributed within the ±6% interval range, indicating that the true and predicted values are very close to each other and the model has a better prediction.

*4.2.3. Performance comparison with different models*

To validate the performance of the proposed improved DBN-CER19 model, it was compared with BPNN, QRNN, ERNN, DBN, and deep

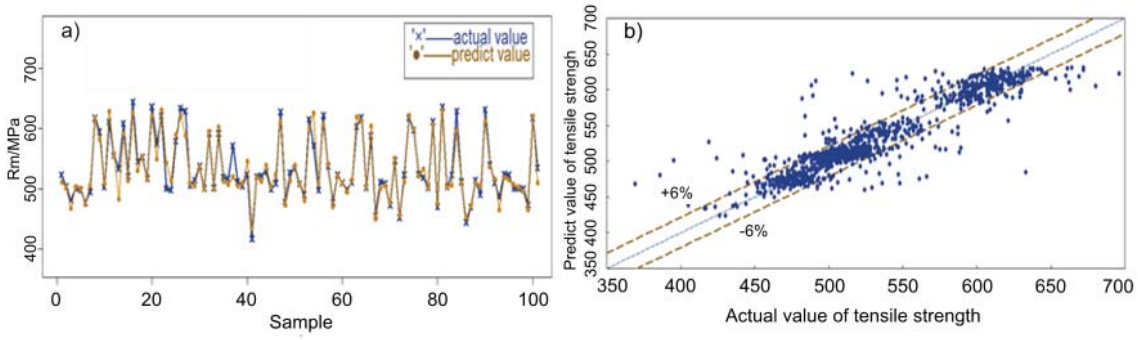


Fig. 7. DBN-CER19 prediction results. (a) Fitting plot of the true and predicted values of the test set (b) Error interval of the true and predicted values of the test set.

Table 3. The statistical results of different models

| Model | Train | | | Test | | |
|---|---|---|---|---|---|---|
| | MAE | MAPE | RMSE | MAE | MAPE | RMSE |
| BPNN | 13.2937 | 2.5128 | 20.7431 | 14.1365 | 2.6302 | 23.6983 |
| QRNN | 13.2100 | 2.5015 | 21.1377 | 13.9400 | 2.6052 | 21.9033 |
| ERNN | 13.0375 | 2.4655 | 20.5016 | 13.9956 | 2.6088 | 23.8001 |
| DBN | 13.3423 | 2.5230 | 20.8438 | 13.8297 | 2.5768 | 21.7370 |
| DBN-QR | 13.2353 | 2.5034 | 21.2197 | 13.6978 | 2.5653 | 21.8364 |
| DBN-CER19 | 12.9499 | 2.4478 | 20.4295 | 13.5035 | 2.5251 | 21.4381 |

belief network quantile regression (DBN-QR) models.For the fairness of comparison, the network structure of these comparison models was kept consistent with DBN-CER19, which are 20-12-36-23-1. The statistical results of different NN models on the training and test sets are shown in Table 3.

Among them, the QRNN model and the ERNN model are the predicted and true value errors under the condition of $\tau = \mathbf{0.5}$. It can be seen that the DBN-CER19 model outperforms the error metrics in both the training and test sets.

Fig. 8 makes a comparison plot of each model on the test set data under the three performance metrics.

From Fig. 8, it is clear that DBN-CER19 is the most accurate and precise model in our experiments. Meanwhile, the pre-trained DBN and DBN-QR models perform second best on MAPE and RMSE, while QRNN and ERNN perform very unstable under different metrics and BPNN performs the worst in the experiments. Overall, the pre-trained DBN model performs better than the model without pre-training because DBN is able to continuously learn more abstract high-level information from low-level data, and the layer-by-layer pre-training of RBM in it can produce very good initial values of parameters, which can achieve smaller empirical errors and stronger generalization ability. And the improved DBN-CER19 using CS algorithm proposed in this paper utilizes multiple expectile regression can improve the efficiency of prediction, and shows obvious superiority under MAE, MAPE, and RMSE metrics.

## 5. Conclusion

This paper investigates the quantitative relationship between the tensile mechanical properties of hot-rolled steel and 20 process and chemical composition parameters. A deep learning model is combined with expectile regression to establish the DBN-CER19 model, which provides a more accurate model prediction method for steel mechanical strength prediction.

The CD algorithm is first used for unsupervised pre-training of the underlying RBM to obtain highly abstract reconstructed features and better initial values of the parameters. The learned reconstructed features are then used as the input of the DBN prediction network, and the BP algorithm is used to perform supervised fine-tuning on the labeled samples, and then the DBN prediction model with shared parameters is obtained to fit the output, avoiding the

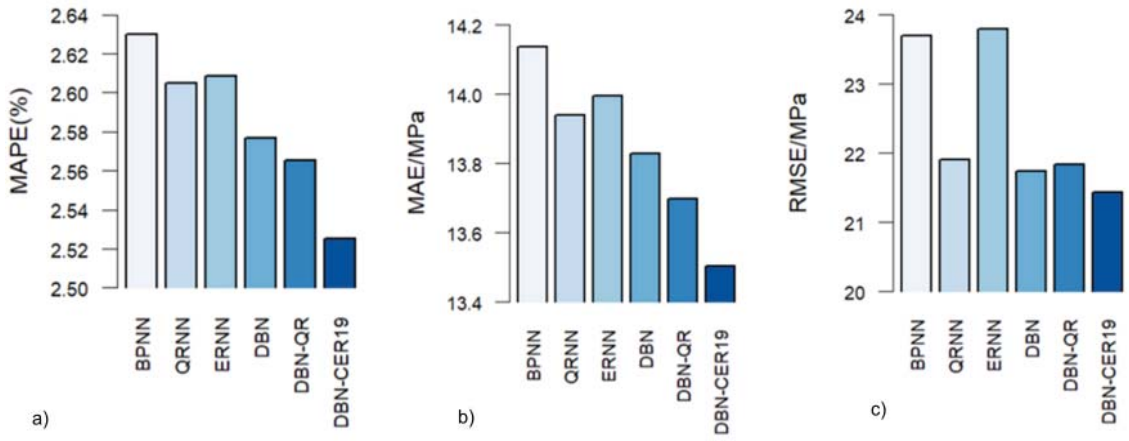


Fig. 8. Performance of each model on the test set

drawback that the BP algorithm is prone to fall into local optimum and long training time due to the random initialization of the weight parameters. Meanwhile, the number of DBN hidden layer network nodes is selected using the CS algorithm to optimize the DBN-CER model structure, which improves the performance of the model while speeding up the convergence of the tuning phase. Expectile regression can handle data heterogeneity well, and it is insensitive to outliers and has good robustness. Moreover, it can consider the overall distribution, and multiple expectile regressions can be used to further improve the prediction efficiency of steel mechanical properties.

Through empirical analysis, the proposed model is compared with BPNN, QRNN, ERNN, and DBN, and the effectiveness of the method is demonstrated. This is important for improving the forecast accuracy of hot rolled steel and provides a degree of theoretical help for high quality production.

***Acknowledgements***

The paper is supported by National Social Science Foundation of China (Grant No.11201356, 51774219), Educational Commission of Hubei Province of China (Grant No.B2020013), and the Hubei Province Key Laboratory of Systems Science in Metallurgical Process (Wuhan University of Science and Technology) (Grant No.201715).

## *References*

[1] BorisadeS.G., AjibolaO.O., Adebayo A.O., OyetunjiA.. *Materials Today: Proceedings*, **38**, 1133, 2021.

[2] WuSiwei, Liu Zhenyu, Zhou Xiaoguang, Shi Naian,.*J.Iron Steel Res,* **28(**12), 1, 2016.

[3] Khalaj G, Azimzadegan T, M Khoeini, *Neural Computing & Applications*, **23**(7-8): 230, 2013.

[4] Chou P Y, Tsai J T, Chou J H. , *IEEE Access*, **4**, 585, 2017.

[5] AdelSaoudi,MamounFellah, NaouelHezil,Djah idaLerari, FaridaKhamouli, L'hadiAtoui, et.al., *Intern. J.Pressure Vessels Piping*, **186**, 104153, 2020.

[6] Hu Shixiong.Mechanical Property Prediction of HotRolled Strip Based on Deep Learning. Wuhan University of Science and Technology, 2019.

[7] XieQian, SuvarnaManu, LiJiali, ZhuXinzhe, CaiJiajia, WangXiaonan. *Mater. & Design*, **197**, 109201, 2021.

[8] Roger Koenker. , *J .Statist. Planning Infer.,* **143**(7),1134, 2013.

[9] Tong Tian, Si Yu Huang, Key Eng Mater, **6234**(1794), 29, 2021.

[10] DadabadaPradeepkumar, Vadlamani Ravi,. *Appl. Soft Computing*, **58**,35, 2017.

[11] ZangHaixiang, LiuChongchong, Teng Jun, Kong Bojun, *Smart Power*, **48**(08), 24, 2020.

[12] Li Bin, PengShurong, Peng Junzhe, Huang Shijun, ZhengGuodong. *Electr. Power Autom. Equipment*, **38**(09),15, 2018. .

[13] Newey W K, Powell J L., *J.Econometric Soc.,* **819**, 1987.

[14] HU Zongyi, Li Yi, Wan Chuang, TangJianyang, *Statist.Inform. Forum*, **34**(04), 19, 2019.

[15] Jiang Cuixia, Jiang Ming, Xu Qifa, Huang Xue,. Neurocomputing, **247,** 73, 2017

[16] JiangMin. Expectile regression neural network-mode1 with applications. Hefei University of Technology, 2018.

[17] LiuXiaoqian, Zhou Yong, *Systems Engin. Theory Practice*, 2020, **40**(08), 217, 2020.

[18] Bao Xuexin, JiangDan, YangXuefeng, Wang Hongmei, *Alexandria Engin J*, **60**(1): 413, 2021

[19] Yu Jianbo, Liu Guoliang,. *Computers Industry*, **121**, 103262, 2020.

[20] Xing Haixia, WangGongming, LiuCaixia, SuoMinghe, *Neural Networks*, 2021, **133**,157, 2021.

[21] Abdel Zaher, Ahmedm., Eldeib, Ayman M. *Expert Systems Applic.* 2016, **46**(15): 139, 2016.
[22] Zou Hui, Yuan Ming, *The Annals Statis.,* **36**(3),1108, 2008.
[23] Hinton G E, Salakhutdinov RR., *Science*, **313**(5786), 504, 2006.
[24] Yang Xinshe, Suash DEB. Cuckoo Search via Levy Flights. 2009World Congress on Nature & Biologically Inspired Computing, 210, 2009.
[25] CarreiraPerpinan M A, Hinton G E, On contrastive divergence learning. Proceedings of artificial intelligence & statistics, 2005.
[26] Nicolas Le Roux, Bengio Y. , *Neural Computation*, **20**(6): 1631, 2008.
[27] Larochelle H, Bengio Y, Louradour J, et al, *J.Machine Learning Res.,* 2009(10), l.
[28] Chen Kui, Laghrouche Salah, DjerdirAbdesslem, *ISA Transactions*, **113**, 175, 2021.
[29] Preeti Malhotra and Dinesh Kumar,. *J.Intelligent Systems,* **28**(2), 321, 2019.
[30] Bo Yang, ZhongqiWang, YuanYang, YonggangKang, Cheng Li, Adv. Mechan. Engin., **9(**6): 145, 2017.