Journal of Computers Vol. 36 No. 1, February 2025, pp. 89-112
https://doi.org/10.63367/199115992025023601007

Regularized Total Least Squares Broad Learning System for Regression

Ke-Jia Xiong[1], Guan-Ci Yang[2*], Tao Zhou[1], and Zhen-Qiang Xie[3]

[1] State Key Laboratory of Public Big Data, Guizhou University,
Guiyang City 550025, Guizhou Province, China

[2] Key Laboratory of Advanced Manufacturing Technology of the Ministry of Education, Guizhou University,
Guiyang City 550025, Guizhou Province, China

[3] National Engineering Research Center of Big Data Application to The Improvement of Governance Capacity,
Guiyang City 550000, Guizhou Province, China
369115381@qq.com, gcyang@gzu.edu.cn, 15284755690@qq.com, xzq521@163.com

Received 8 May 2024; Revised 13 September 2024; Accepted 27 September 2024

Abstract. The Broad Learning System (BLS) has been extensively developed and applied across various fields due to its significant advantages, including high efficiency, strong generality, and scalability. However, in practical applications where both inputs and outputs are contaminated by noise, traditional BLS demonstrates suboptimal performance in handling sample data. This study introduces a novel regularized total least squares broad learning system (RTLS-BLS) designed to enhance the robustness of BLS when noisy values are present in both the input and the output of the training set. Initially, the *k*-support norm is integrated into the BLS-based autoencoder (BLS-AE), embedded within BLS, to extract robust features from the original input data. The BLS-AE equipped with the k-support norm effectively addresses or mitigates issues related to overly sparse (based on L_1-norm) or overly dense (based on L_2-norm) regularization of extracted features. Subsequently, regularized total least squares (RTLS) are employed to assess the output weights of BLS, further enhancing its robustness. Moreover, simultaneous perturbation measures for the coefficient matrix and the output are provided. Experimental results from two function approximation tasks, eight benchmark regression tasks, and two network interface flow datasets demonstrate that the proposed RTLS-BLS achieves significantly robust performance under noisy conditions compared to other methods.

Keywords: broad learning system, regularized total least squares, *k*-support norm, BLS-based autoencoder

1 Introduction

Over the past decades, Deep Learning (DL) has exhibited excellent performance across various domains such as computer vision [1], natural language processing [2], speech recognition [3], recommendation systems [4], and autonomous driving [5]. DL techniques are a type of representation-learning approach that involves multiple layers of representation. These layers are created by combining simple yet non-linear modules, each responsible for transforming the representation at one level (starting from the initial input) into a slightly more abstract representation at a higher level [6]. However, the training process of DL models typically involves massive training data to tune the model parameters iteratively and complicated network architecture. The time-consuming iterative adjustment process poses a significant challenge to its application in scenarios with high real-time requirements and limited computational resources.

To address the limitations of deep learning models, Chen et al. [7] proposed a novel shallow neural network based on random vector functional link neural networks (RVFLNN), termed the broad learning system (BLS). This model utilizes a flattened network architecture, significantly enhancing the speed and efficiency of data modeling. The sparse autoencoder is employed to fine-tune input samples and extract sparse features. These features are then used to generate enhancement nodes, with randomly generated weights, biases, and activation

* Corresponding Author

functions. The output weights are determined by computing the pseudo-inverse of the combined output matrix from the feature and enhancement layers. Chen et al. [7] describe three incremental learning strategies within BLS: 1) increasing the number of feature mapping nodes, 2) adding additional enhancement nodes, and 3) expanding the input data. Notably, the BLS model allows for incremental reconstruction without the need for retraining, should network expansion become necessary.

To leverage the powerful feature extraction capabilities of convolutional neural networks (CNN), a cascade CNN was integrated into the broad learning system (BLS) by Chen et al. [8]. This integration facilitates the generation of a series of convolutional feature mapping nodes through the use of convolution and pooling operations. Yang et al. [9] proposed the CNN-based broad learning system, which employs CNN to replace the traditional feature extraction methods and utilizes principal component analysis (PCA) for data dimensionality reduction. Li Ting et al. [10] integrated CNN with the Adam algorithm, leveraging CNN for feature extraction while employing the Adam algorithm to update the feature weights. Additionally, for tasks involving fuzzy uncertain data modeling, the Takagi-Sugeno (TS) fuzzy system was embedded into BLS to enhance its fuzzy inference capabilities. In Fuzzy BLS [11], feature nodes were replaced with fuzzy subsystems, and the k-means algorithm was employed to determine both the number of fuzzy rules and the centers of the Gaussian membership functions. Huang et al. proposed a BLS with manifold regularized sparse features (BLS-MS) [12]. This model employs a manifold regularized sparse autoencoder based on extreme learning machine (MS-ELM-AE) for the feature mapping process. Ding et al. [13] introduced a greedy learning strategy and proposed the greedy broad learning system. This approach addresses the issue of redundancy in hidden layer neurons within the BLS framework, offering an effective solution to optimize the network structure. In an effort to enrich the diversity of BLS variants, Zhang et al. [14] introduced four novel variants of BLS. These variants are characterized by distinct types of nodes (feature nodes and enhancement nodes) and various cascading manners (pyramid, dropout, and dense). Zhao et al. [15] proposed a semi-supervised broad learning system (SS-BLS), which constructs a Laplacian matrix using a manifold regularization framework. By combining feature nodes, enhancement nodes, and the Laplacian matrix to formulate the objective function, this method extends the application of BLS to semi-supervised learning tasks. Fan et al. [16] proposed a class-specific weighted broad learning system (CSWBLS), which constructs the least squares error term on a class-specific basis and imposes weight constraints to modulate the contribution of each class to the model, thereby enhancing the accuracy of heartbeat classification in medical images. In [17] a Broad Learning System employing the maximum correlation criterion (C-BLS) along with its corresponding training algorithm was presented to mitigate the negative effects of outliers on BLS performance. Huang et al. [18] integrated random projection with residual reduction to generate enhancement nodes, thereby proposing a novel bidirectional broad learning system (B-BLS). This approach not only enhances computational efficiency but also effectively reduces the number of hidden layer nodes.

BLS has demonstrated outstanding performance in diverse real world data modeling tasks, such as predicting the interactions between lncRNA and protein [19], wear fault diagnosis of aircraft engines [20], multi-step-ahead wind speed prediction [21], emotion recognition [22], traffic predictors [23], etc.

The standard BLS typically assumes that the training data are noise-free. However, in many real-world scenarios, the collected sample data may suffer from contamination caused by noise or outliers, leading to a significant decline in the performance and stability of BLS. To enhance model robustness, Jin et al. [24] proposed a graph regularized BLS (GRBLS) by incorporating manifold learning into the objective function of BLS. This approach constrains the output weights to extract more discriminative information from image data, addressing the issue of label noise in raw data. Feng et al. [25] employed the random perturbation approximation to discard features that are not related to low-dimensional manifold embedding, thereby improving the robustness of BLS in chaotic time series prediction. To enhance the stability and generalizability of BLS in modeling tasks involving uncertain data, Jin et al. [26] introduced three robust versions of BLS (RBLS), each penalized by the L_1 norm on the data fidelity term, while the regularization terms are individually subjected to L_1 norm, L_2 norm, and elastic net regularization, respectively. Guo et al. [27] introduced M-estimator robust learning techniques into the BLS. By integrating diverse M-estimator cost functions and their corresponding weighting strategies, they performed inverse weighting calculations on samples to mitigate or eliminate the adverse effects of outlier errors on the learning model. In [28], a novel adaptive weight generation mechanism based on imbalanced data distribution was designed, assigning weights to each training sample through a generalized weighting scheme. This approach aims to enhance the classification accuracy of BLS on imbalanced datasets, ensuring more effective handling of the prevalent class imbalance issues and improving overall predictive performance.

The aforementioned BLS variants exhibit superior robustness compared to the original BLS when data samples are corrupted by noise or outliers. However, there still exists a significant limitation: the existing BLS-based methods primarily focus on the approximation problem of sample data contaminated by either input or output noises, rather than tackling the error-in-variables (EIV) model that involves noise in both input and output values. To address this issue, a novel regularized total least squares broad learning system (RTLS-BLS) is proposed in this study. In RTLS-BLS, the k-support norm is ingeniously integrated into the objective function of BLS-AE, serving as a powerful tool for extracting robust features from raw input data. Simultaneously, by incorporating regularized total least squares, the weight estimation process is further optimized, ensuring the model's stability and accuracy in noisy environments. In summary, the primary contributions of this study can be summarized as follows:

1) To extract high-level abstract features from input data, a *k*-support norm regularization term is embedded into the loss function of the BLS-based autoencoder, resulting in the k-support norm regularized BLS-AE (KSP-BLS-AE). Subsequently, KSP-BLS-AE is cleverly integrated into the BLS system, serving as a key component for extracting robust features from raw input data. This design enhances the feature extraction performance of the BLS model in noisy environments.

2) The RTLS-BLS model integrates the essence of BLS with the advantages of regularized total least squares. It first employs the BLS concept to select the hidden layer weights, then utilizes regularized total least squares to estimate the output weights, achieving an efficient approximation of the EIV model. Additionally, by accounting for potential disturbances in both the hidden outputs and observed values, the model is optimized to enhance its robustness and adaptability, ensuring more stable and accurate performance when dealing with complex data.

3) Extensive experiments were conducted to validate the performance of our proposed RTLS-BLS in regression tasks with noise in both input and output samples. The experimental results indicate that RTLS-BLS exhibits superior performance compared to other benchmark methods.

The rest of the paper is organized as follows: Chapter 2 briefly reviews BLS and BLS-based autoencoder. Chapter 3 presents the proposed KSP-BLS-AE and RTLS-BLS algorithm. The test performance of RTLS-BLS on various benchmarks is detailed in Chapter 4. Finally, Chapter 5 concludes this article.

2 Related Work

This section provides a brief review of the classical BLS framework and the single hidden layer autoencoder constructed based on BLS (BLS-AE).

2.1 Broad Learning System

The Broad Learning System (BLS) is an innovative shallow neural network that is based on the random vector functional-link neural network (RVFLNN). Its primary objective is to address the challenges of high computational requirements and lengthy training times in deep learning [7]. As depicted in Fig. 1, the hidden layer of BLS consists of a single-layer structure comprising the feature layer and the enhancement layer.

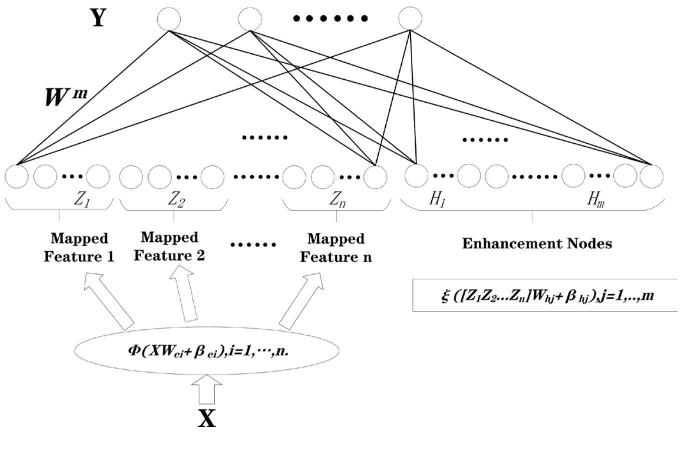


Fig. 1. The network structure of BLS [7]

Assuming a training dataset $\{X,Y\} \in \mathbb{R}^{N\times(d+c)}$. For n feature mappings, each mapping generates k nodes, can be represented as the equation of the form:

$$Z_i = \phi_i(XW_{e_i} + \beta_{e_i}), i = 1,2,\ldots,n. \quad \textbf{(1)}$$

where W_{e_i} and β_{e_i} are randomly generated weights and biases, respectively. $\phi_i(\cdot)$ is a nonlinear activation function, such as ReLU or Tanh.

Assuming the network contains m groups of enhancement nodes, each group contains p nodes. The jth group of enhancement nodes is represented as

$$H_j = \xi_j(Z^n W_{h_j} + \beta_{h_j}), j = 1,2,\ldots,m. \quad \textbf{(2)}$$

where $Z^n = [Z_1, Z_2, \ldots, Z_n]$, W_{h_j} and β_{h_j} are randomly generated weights and biases, respectively. $\xi_j(\cdot)$ is a nonlinear activation function. Denote all the enhancement nodes as $H^m = [H_1, H_2, \ldots, H_m]$.

Therefore, the broad learning system can be represented as

$$Y = [Z_1, Z_2, \ldots, Z_n \mid H_1, H_2, \ldots, H_m]W^m = [Z^n \mid H^m]W^m. \quad \textbf{(3)}$$

where the $W^m = [Z^n|H^m]^+Y$ is the output weights calculated by generalized inverse with the ridge regression approximation [7].

2.2 BLS-Based Autoencoder

An autoencoder (AE) [29] is recognized as an unsupervised learning model, primarily designed to learn the representation of input information. In this model, the input data itself serves as the reconstruction target. The model consists of two principal components: the encoder and the decoder. The encoder is used to map the input data to a lower-dimensional feature space, while the decoder is employed to map these low-dimensional feature vectors back to the original data space.

Based on BLS theory, BLS-AE [30] incorporates randomly generated hidden layer, and its output weight matrix tends to produce results similar to the input. Fig. 2 illustrates the fundamental structure of a BLS-based autoencoder. BLS-AE can also be stacked to create a deep architecture.

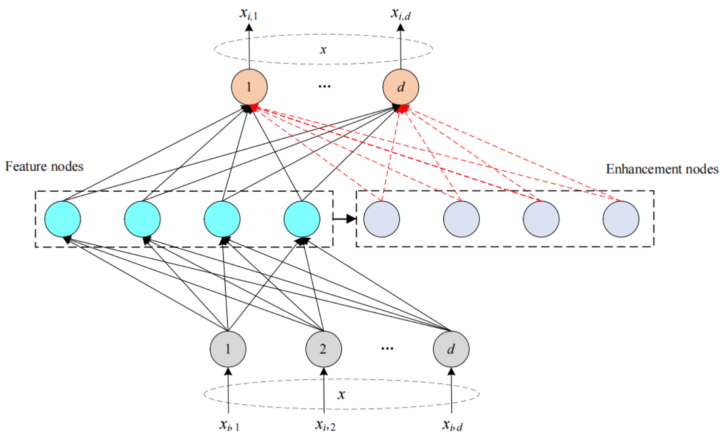


Fig. 2. The network structure of BLS-based autoencoder

Given the input data X, BLS-AE generates the feature layer and the enhancement layer through Eq.(1) and Eq.(2), respectively.

The feature layer and enhancement layer are combined to form the hidden layer, which is expressed as follows:

$$A = [Z_1, Z_2, ..., Z_n \mid H_1, H_2, ..., H_m] = [Z^n \mid H^m]. \tag{4}$$

Ridge regression is employed to achieve rapid reconstruction. The output weight matrix W is optimized by solving the subsequent minimization problem:

$$\underset{W}{\arg\min} : \|AW - X\|_2^2 + \lambda \|W\|_2^2. \tag{5}$$

By setting the gradient to zero, the W is approximated as

$$W = (\lambda I + A^T A)^{-1} A^T X. \tag{6}$$

Afterwards, the original data X can be projected using W^T. In other words, XW^T is the learned representation.

2.3 The Key Research Problem

Existing variants of the BLS are predicated on the assumption that input variables are completely accurate, with noise only present in the output variables, as depicted in Fig. 3. In practice, however, noise not only affects the output variables but also impacts the input variables, corresponding to an EIV model. Consequently, developing a robust BLS framework capable of universally handling the EIV model becomes the central research focus of this paper. This is critically important for enhancing the accuracy of data processing and model predictions.

In the RTLS-BLS framework, we initially embed the KSP-BLS-AE into the BLS for feature extraction, enhancing the model's ability to extract features in noisy environments. The KSP-BLS-AE incorporates a k-support norm regularization term into the objective function of the BLS-based autoencoder, facilitating the extraction of more robust features. The solution process for KSP-BLS-AE efficiently leverages the Alternating Direction Method of Multipliers (ADMM) and the precision of the k-support norm proximal operator, ensuring the accuracy and robustness of feature extraction. Subsequently, we employ regularized total least squares (R-TLS) to estimate the output weights, thereby achieving an effective approximation of the EIV model. Furthermore, to enhance the model's robustness and adaptability, potential disturbances in the hidden output matrix and observed values are explicitly considered and incorporated into the model optimization process, ensuring that RTLS-BLS exhibits robust and accurate performance when confronted with complex real-world data.

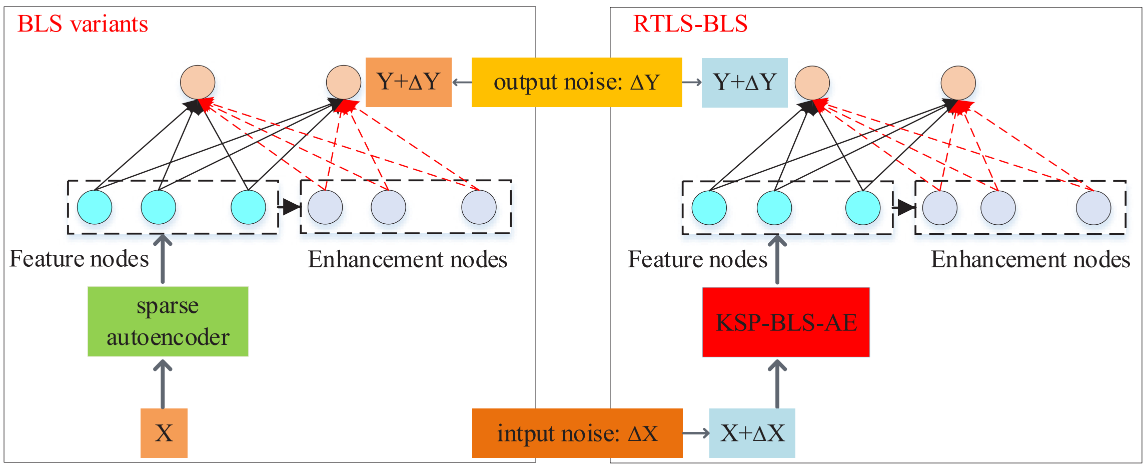


Fig. 3. The difference between the proposed RTLS-BLS framework and other variants of BLS

3 The Proposed Total Least Squares Broad Learning System

3.1 *k*-Support Norm Regularized BLS-Based Autoencoder

In this subsection, the *k*-support norm regularized BLS-based autoencoder (KSP-BLS-AE) is introduced to extract more effective features and enhance the model's robustness. The k-support norm is considered the tightest relaxation of sparsity and L_2 constraints. It overcomes the limitations of existing matrix norms in terms of sparsity and/or low-rank, including: 1) excessive sparsity/density, and/or 2) column independence [31].

Theorems 1. [32] for $\forall W \in \mathbb{R}^d$, let $|W|_0^{\downarrow} \to -\infty$, t is $\{0, ..., j-1\}$ the unique integer that satisfies the following conditions:

$$|W|_{j-t-1}^{\downarrow} > \frac{1}{t+1}\sum_{i=j-t}^{d} |W|_i^{\downarrow} > |W|_{j-t}^{\downarrow} . \tag{7}$$

In that case, the *k*-support norm can be expressed as:

$$\|W\|_k^{sp} = (\sum_{i=1}^{j-t-1} (|W|_i^{\downarrow})^2 + \frac{1}{t+1}(\sum_{i=k-j}^{d} |W|_i^{\downarrow})^2)^{\frac{1}{2}}. \tag{8}$$

where $|W|_i^{\downarrow}$ represents the value of the *i*th element of vector *W* after sorting it in descending order.

The optimization model of KSP-BLS-AE, based on the *k*-support norm regularization term, can be denoted by

$$\arg\min_{W} : \|AW - X\|_2^2 + \frac{\lambda}{2}(\|W\|_k^{sp})^2. \tag{9}$$

where *X* is the input data, *A* is the output of the hidden layer, λ is a regularization parameter. *W* is the output layer weight that needs to be determined.

We use the ADMM [33] algorithm and proximity operator to solve Eq.(9). Introducing an intermediate variable *o* in Eq.(9), we transform *W*=*o* into a constrained model:

$$\min_{\mathrm{W}} \| AW - X \|_2^2 + \frac{\lambda}{2}(\| o \|_k^{sp})^2 \quad s.t.\ W = o. \tag{10}$$

According to the ADMM optimization method, we first use the Lagrange method to transform the constrained Eq.(10) into an unconstrained model:

$$L(W, o, \mu) = \| AW - X \|_2^2 + \frac{\lambda}{2}(\| o \|_k^{sp})^2 + < \mu, W - o > . \tag{11}$$

where μ is the Lagrange multiplier that needs to be iterated. Then, we add a penalty term $\frac{\rho}{2}\| W - o \|_2^2$ to Eq.(11), resulting in the augmented Lagrange function for the original problem:

$$\begin{aligned} L_\rho(W, o, \mu) &= \| AW - X \|_2^2 + \frac{\lambda}{2}(\| o \|_k^{sp})^2 + < \mu, W - o > \\ &+ \frac{\rho}{2}\| W - o \|_2^2 = \| AW - X \|_2^2 + \frac{\lambda}{2}(\| o \|_k^{sp})^2 \\ &+ \frac{\rho}{2}\| W - o + \frac{\mu}{\rho} \|_2^2 - \frac{\| \mu \|_2^2}{2\rho}. \end{aligned} \tag{12}$$

where $\rho > 0$ is penalty parameter.

Taking the derivatives with respect to W, o, and μ , and setting them equal to zero, we obtain:

$$W^i = (2A^T A + \rho I)^{-1}(2A^T X + \rho o^{i-1} - \mu^{i-1}). \tag{13}$$

$$o^i = prox_{\frac{\lambda}{2\rho}\|\cdot\|_k^{sp2}}(W^i + \frac{\mu^{i-1}}{\rho}). \tag{14}$$

$$\mu^i = \mu^{i-1} + \rho(W^i - o^i). \tag{15}$$

where $prox_{\frac{\lambda}{2\rho}\|\cdot\|_k^{sp2}}(\cdot)$ is proximity operator. The proximity operator method and its corollaries are represented as follows.

Lemmas 1. [32] If function $f(x) = \frac{1}{2\rho}(\|x\|)^2$, the proximal operator of it is given by:

$$prox_f(x) = \begin{cases} \frac{\rho}{\rho+1}, \quad i = 1, ..., j-t-1 \\ z_i - \frac{T_{t,l}}{l-j+(\rho+1)t+\rho+1}, \quad i = j-t, ..., l \\ 0, \quad i = l+1, ..., d \end{cases} \tag{16}$$

where $z := |x|^{\downarrow}, z_0 := +\infty, z_{d+1} := -\infty, T_{t,l} := \sum_{i=j-l}^{l} z_i$, and $r \in \{0, ..., j-1\}$, $l \in \{j, ..., d\}$ satisfies:

$$\frac{1}{\rho+1} z_{j-t-1} > \frac{T_{t,l}}{l-j+(\rho+1)t+\rho+1} > \frac{1}{\rho+1} z_{j-t}. \tag{17}$$

$$z_l > \frac{T_{t,l}}{l-j+(\rho+1)t+\rho+1} > z_{l+1}. \tag{18}$$

Table 1 exhibits pseudo-code of the *k*-support norm regularized BLS-Based Autoencoder (KSP-BLS-AE).

Table 1. *k*-support norm regularized BLS-based autoencoder

```
Algorithm 1.
Input     The training data $X$, activation function $\varphi$ and $\xi$ ;
Output    The abstract feature $XW^T$ ;
1         While $i \le n$
2             Randomly construct weights $W_{e_i}$ and biases $\beta_{e_i}$ ;
3             Calculate the feature nodes by Eq.(1);
4         End While
5         Set the feature group $Z^n = [Z_1, Z_2, ..., Z_n]$
6         While $j \le m$
7             Randomly construct weights $W_{h_j}$ and biases $\beta_{h_j}$ ;
8             Calculate the enhancement nodes by Eq.(2);
9         End While
10        Set the enhancement group $H^m = [H_1, H_2, ... , H_m]$
11        Splicing all the nodes to get the hidden layer nodes by Eq.(4);
12        While $t \le MaxIt$
13            Update $W$ according to Eq.(13);
14            Update $o$ according to Eq.(14);
15            Update $\mu$ according to Eq.(15);
16        End While
```

3.2 Regularized Total Least Squares Broad Learning System

As noise is commonly present in both the input and output variables in practical applications, the existing BLS methods only consider the case of noise in the output variables. To address this issue, we introduce a regularized total least squares broad learning system (RTLS-BLS) to handle the errors-in-variables (EIV) model.

Considering a set of N training pairs $(x_i, y_i)_{i=1}^{N} \in \mathbb{R}^d \times \mathbb{R}^c$ with input noise Δx_i and output noise Δy_i , where $i = 1, ..., N$. The BLS with a nonconstant bounded feature mapping φ and activation function ξ can be equivalently denoted as:

$$\begin{aligned} y+\Delta y &= \sum_{i=1}^{n*k} W_i \varphi((x+\Delta x)W_{e_i} + \beta_{e_1}) + \sum_{j=1}^{m*q} W_{nk+j}\zeta((Z+\Delta Z)W_{h_j} + \beta_{h_j}) \\ &= \sum_{i=1}^{n*k} W_i \varphi((x+\Delta x)W_{e_i} + \beta_{e_1}) + \sum_{j=1}^{m*q} W_{nk+j}\zeta(x+\Delta x; \{\varphi, W_{h_j}, \beta_{h_j}\}). \end{aligned} \quad \textbf{(19)}$$

where $Z+\Delta Z = [\varphi((x+\Delta x)W_{e_1} + \beta_{e_1}), ..., \varphi((x+\Delta x)W_{e_{nk}} + \beta_{e_{nk}})]$, and output layer weights $W = (W_1, ..., W_{nk+mq})$, W_{e_i} and β_{e_i} are the randomly generated weights and biases of the feature nodes, W_{h_j} and β_{h_j} are the randomly generated weights and biases of the enhancement nodes. Eq.(19) can be written compactly as

$$(A+\Delta A)W = Y + \Delta Y. \quad \textbf{(20)}$$

where A is the output of the hidden layer, ΔA is the hidden perturbation matrix.

Now, Eq.(19) is transformed into how to compute the output weight vector W under the linear system with the unknown perturbation matrix ΔA and ΔY. As is well known, BLS determines the output weight matrix W by solving the linear system.

$$AW = Y + \Delta Y. \quad \textbf{(21)}$$

The least squares method is utilized to minimize the 2-norm of the residual $\Delta Y = Y - AW$. However, in Eq.(19), both A and Y are affected by perturbations, and relying solely on the least squares method may not sufficiently account for generalization. The Total Least Squares (TLS) [34] can effectively address Eq.(19). Taking into account that the singular values of $[A\ Y]$ may gradually diminish towards zero, implying an ill-posed problem, we employ regularized TLS (R-TLS) [35], which integrates Tikhonov regularization into TLS to tackle the ill-posed nature of the problem. Hence, we employ R-TLS to determine the optimal output weights W as well as the perturbation matrices ΔA and ΔY, such that

$$\min_{W,\Delta A,\Delta Y} \| [\Delta A \ \ \Delta Y] \|_F \ \ \textit{subject to} \ (A+\Delta A)W = Y+\Delta Y, \| W \|_2 \le \delta. \quad \textbf{(22)}$$

where $\|\cdot\|_F$ is the Frobenius norm, δ is a given positive constant.

$$\begin{aligned} &\| [\Delta A\ \Delta Y] \|_F^2 = tr([\Delta A\ \Delta Y]^T \cdot [\Delta A\ \Delta Y]^T) = tr(\Delta A \Delta A^T + \Delta Y \Delta Y^T) \\ &= vec(\Delta A)^T vec(\Delta A) + \Delta Y^T \Delta Y. \end{aligned} \quad \textbf{(23)}$$

The corresponding Lagrange multiplier formulation is

$$L(E_A, \Delta Y, W, \rho, \mu) \ = E_A^T E_A + \Delta Y^T \Delta Y + \rho(\| W \|_2^2 - \delta^2) + 2\mu(Y + \Delta Y - AW - \Delta AW). \quad \textbf{(24)}$$

where ρ and μ are Lagrange multiplier, $E_A = vec(\Delta A)$ and $\Delta AW = (W^T \otimes I)\ E_A$, $\otimes$ represents the Kronecker-Zehfuss product, and $vec(\cdot)$ is the flattening transformation of a matrix.

Let $H = [A^T A + \lambda I\ A^T Y]$, where λ is the regularization parameter. Its SVD decomposition is given by $H = U \sum V$, where $\sum = \text{diag}(\delta_1, ..., \delta_{L+s})$. We can partition $\sum$ and V as

$$\sum = \begin{array}{cc} \left[\begin{array}{cc} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{array}\right] & \begin{array}{c} L \\ s \end{array} \\ \begin{array}{cc} L & s \end{array} & \end{array}, V = \begin{array}{cc} \left[\begin{array}{cc} V_{11} & V_{12} \\ V_{21} & V_{22} \end{array}\right] & \begin{array}{c} L \\ s \end{array} \\ \begin{array}{cc} L & s \end{array} & \end{array}. \quad (25)$$

If V_{22} is nonsingular, then the output weight matrix is given by

$$\hat{W} = -V_{12}V_{22}^{-1}. \quad (26)$$

Taking the partial derivatives of E_A, ΔY, and μ with respect to Eq.(24), and setting them to zero, we can rearrange to obtain:

$$\Delta Y = -(Y - AW)(1 + W^T W)^{-1}. \quad (27)$$

$$\Delta A = (Y - AW)(1 + W^T W)^{-1} W. \quad (28)$$

$$\mu = (Y - AW)(1 + W^T W)^{-1}. \quad (29)$$

Therefore, the correction matrix is given by

$$\Delta Y = -(Y - A\hat{W})\,(1 + \hat{W}^T\,\hat{W})^{-1}. \quad (30)$$

$$\Delta A = (Y - A\hat{W})\,(1 + \hat{W}^T\,\hat{W})^{-1}\,\hat{W}. \quad (31)$$

Table 2 presents the pseudocode for the regularized total least squares broad learning system (RTLS-BLS).

Table 2. Regularized total least squares broad learning system

Algorithm 2.	
Input	The training dataset: $\{X, Y\} \in \mathbb{R}^{N\times(d+c)}$, activation function φ and ξ;
Output	W, ΔA, ΔY;
1	While $i \le n$
2	Obtain XW^T according to Algorithm 1;
3	Calculate $Z_i = \varphi(XW^T)$;
4	End While;
5	The feature mapping nodes as $Z^n = [Z_1, ..., Z_n]$;
6	While $j \le m$
7	Randomly construct weights W_{h_j} and biases β_{h_j} ;
8	Calculate the enhancement nodes by Eq.(2) and make the concatenation;
9	End While
10	Splicing all the nodes to get the hidden layer output matrix A by Eq.(4);
11	Performing the SVD decomposition on the matrix $H = [A^TA + \lambda I\ A^TY] = U \sum V^T$. Divide $\sum$ and V by Eq.(25);
12	If V_{22} is nonsingular then
13	Update W according to Eq.(26);
14	Update ΔY according to Eq.(30);
15	Update ΔA according to Eq.(31);
16	End If

3.3 Connections to Other Methods

In both the standard BLS and robust BLS variants such as GRBLS [24] and weighted BLS [28], the least squares method or weighted least squares method is employed as the objective function. In cases where only the output variable contains noise, the model parameters estimated via least squares possess desirable statistical properties such as being unbiased, consistent, and having minimal variance. However, in practical scenarios, the coefficient matrix is also affected by noise. If least squares is applied directly under these conditions, the estimated parameters no longer exhibit optimal statistical characteristics.

Three versions of RBLS [26] (L1RBLS, L2RBLS, and ENRBLS) utilize the L_1 loss function, which in certain scenarios offers advantages over the least squares method, such as when a sparse solution is required. However, since the L_1 norm is non-differentiable at zero, solving problems that minimize the L_1 loss function is generally more complex than solving least squares problems. Additionally, RBLS only considers scenarios where noise is present in the output variables, without accounting for cases where both input and output variables are noisy. Consequently, RBLS does not achieve satisfactory results when dealing with EIV models.

The RTLS-BLS method we propose employs total least squares as the objective function to establish an EIV model, taking into account noise in the output variables as well as in the coefficient matrix. Additionally, RTLS-BLS utilizes KSP-BLS-AE for feature extraction, which offers greater robustness compared to the sparse autoencoder used in other BLS implementations. This increased robustness stems from the *k*-support norm used in KSP-BLS-AE, which overcomes the limitations of the L_1 norm in terms of sparsity and/or low-rank. Consequently, RTLS-BLS exhibits significant advantages when handling data samples with noise in both inputs and outputs. The differences between RTLS-BLS and other variants of BLS are shown in Table 3.

Table 3. Differences between BLS variants

Algorithms	Autoencoder	Loss function	Regularization term	Whether to consider input noise	Whether to consider output noise
BLS	sparse autoencoder	least squares	L_2 regularization	No	No
GRBLS	sparse autoencoder	least squares	manifold regularization	No	Yes
L1RBLS	sparse autoencoder	L_1 loss function	L_1 regularization	No	Yes
L2RBLS	sparse autoencoder	L_1 loss function	L_2 regularization	No	Yes
ENRBLS	sparse autoencoder	L_1 loss function	elastic-net regularization	No	Yes
Weighted BLS	sparse autoencoder	weighted least squares	L_2 regularization	No	Yes
RTLS-BLS	KSP-BLA-AE	total least squares	Tikhonov regularization	Yes	Yes

4 Experimental

4.1 Experimental Setup

Parameter Setting. Several experiments were conducted to verify the performance of the proposed RTLS-BLS. The RTLS-BLS was compared with the standard BLS [7], BLS with L_1 regularization (denoted as BLS-L1), RBLS with L_1 and L_2 regularizations [26] (denoted as L1RBLS and L2RBLS, respectively), and broad stochastic configuration networks [36] (denoted as BSCN). All programs were implemented using MATLAB 2022b on a PC equipped with an Intel(R) Core(TM) i7-10875H CPU at 2.30GHz, an NVIDIA GeForce RTX 2060, and 16GB of RAM. The primary parameter configurations for each model are outlined as follows.

Grid search is employed to determine the optimal number of feature mapping nodes N_w, and the number of mapping nodes per group N_f. The grid search range of N_f spans from 1 to 15 with a step of 1. Similarly, the grid search range for N_w spans from 1 to 20 with a step of 1. The number of enhanced nodes N_e is set at a fixed value of 100. The output weights of BLS-L1 is evaluated using Eq.(32). The number of enhanced node groups in BSCN is set to 1, with each group consisting of 50 enhanced nodes. The tolerance threshold is fixed at 1e-10, and the maximum number of candidate nodes is set to 200. A set of positive scalars, denoted by γ , is defined within the range of $\{1,5,10,\ldots,30,50,100,150,200\}$. The regularization parameter λ is set to 2^Q, and the grid search range for Q spans from -8 to 8 with a step size of 1.

To ensure fairness in the experiment, the sigmoidal function (Eq.(33)) is selected as the activation function for BLS, BLS-L1, BSCN, L1RBLS, L2RBLS, and RTLS-BLS.

$$\arg\min_{W} \| AW - Y \|_2^2 + \lambda \| W \|_1 . \tag{32}$$

$$f(x) = \frac{1}{1+e^{-x}} . \tag{33}$$

Benchmark Datasets. To assess the regression performance of the RTLS-BLS, experiments were conducted on two function approximation problems, represented by Eq.(34) and Eq.(35), as well as on several real-world datasets sourced from the KEEL and UCI repositories. The specifications of these benchmark regression datasets are listed in Table 4. To demonstrate the effectiveness of the proposed RTLS-BLS for the Errors-in-Variables (EIV) model, noises and outliers were introduced into the training dataset in accordance with Eq.(36) and Eq.(37), while the test dataset was maintained uncontaminated.

Table 4. Specifications of benchmark datasets

Datasets	Type	Attributes	Outputs	Training	Test
concrete	Regression	8	1	721	309
ele-2	Regression	4	1	740	316
friedman	Regression	5	1	840	360
plastic	Regression	2	1	1155	495
stock	Regression	9	1	665	285
laser	Regression	4	1	696	297
autoMPG8	Regression	7	1	275	117
wankara	Regression	9	1	1127	482

Function approximation

$$y_1 = sin(2\pi x), x \in [0,1]. \tag{34}$$

In this experiment, we randomly generated 600 training and 600 test samples from the uniform distribution in the range of [0, 1], respectively.

$$y_2 = \frac{sin(3x)}{3x}, x \in [-4,4]. \tag{35}$$

For the second function approximation tasks, both the training and the test dataset consist of 1000 grid points uniformly distributed over the interval [-4, 4].

Real-world datasets

For the real-world datasets, normalization preprocessing is applied to minimize the impact of varying data scales, converting the input and output vectors into the [0, 1] range. Subsequently, the datasets are divided into training and test sets, with 70% of the samples allocated to training and the remaining 30% reserved for testing purposes.

Noise and outliers

To demonstrate the approximation effectiveness of the proposed RTLS-BLS for the Errors-in-Variables (EIV) model, experiments were conducted using a subset of function approximation training samples, with 10% randomly selected, as well as different proportions (10%, 20%, 30%, 40%) of the training samples from the real dataset. The input and target outputs of the selected samples were preprocessed according to Eq.(36) and Eq.(37), while the test dataset was kept unaffected.

$$x_{i,Outlier} = x_i + 1.0 \times rnd - 0.5. \tag{36}$$

$$y_{i,Outlier} = y_i + 1.0 \times rnd - 0.5. \tag{37}$$

where, x_i represents the attribute of the selected sample, while y_i denotes the target output of the selected sample. $x_{i,Outlier}$ and $y_{i,outlier}$ are used to represent an outlier, and *rnd* signifies a random number within the range (0,1).

Evaluation Indicator. In the evaluation of the proposed RTLS-BLS model, several widely recognized regression performance metrics were utilized, specifically mean absolute error (MAE) (Eq.(38)), root mean squared error (RMSE) (Eq.(39)), mean squared error (MSE) (Eq.(40)), and determination coefficient (R^2) (Eq.(41)). Smaller values of MAE, MSE, and RMSE indicate higher accuracy in regression. The R2 value, ranging between 0 and 1, reflects the model fit, with values closer to 1 indicating better fit. To ensure experimental validity, the RTLS-BLS model and relevant comparative models were independently executed 100 times. The generalization capability of the proposed model was assessed by calculating the average values of the evaluation metrics and their corresponding standard deviations.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \widehat{y_i}|. \tag{38}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y_i})^2}. \tag{39}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y_i})^2. \tag{40}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \widehat{y_i})^2}{\sum_{i=1}^{N}(\overline{y_i} - \widehat{y_i})^2}. \tag{41}$$

where y_i is the label of sample i, $\widehat{y_i}$ represents the regression result of sample i, N indicates the total number of samples.

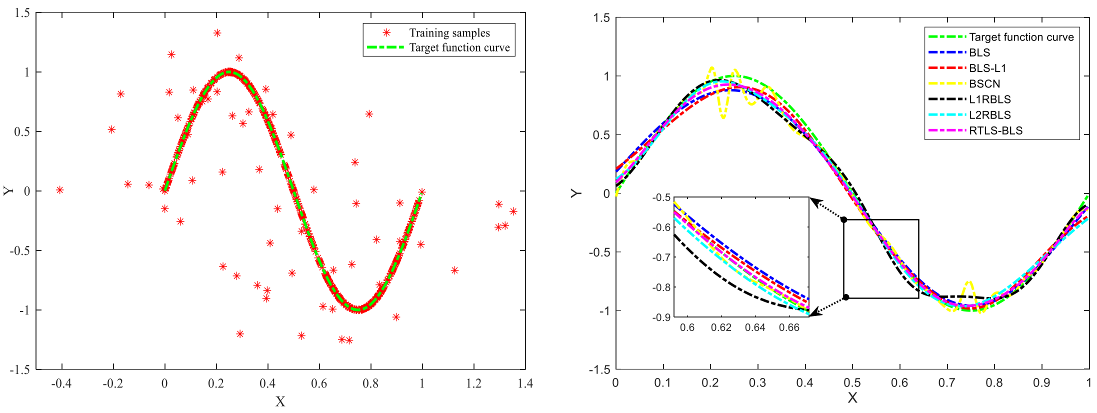
4.2 Experimental Results

Performance on The Function Approximation Problems. Table 5 presents the average test results for the function approximation problem at a pollution rate of 10%. The values shown are the averages and standard deviations of the evaluation metrics. From an examination of Table 5, it is observed that the RTLS-BLS model achieves the smallest MAE, MSE, and RMSE values, as well as the largest R^2 value among all models. These findings suggest that the RTLS-BLS model surpasses other models in regression accuracy and exhibits a superior approximation effect for the EIV model.

Table 5. Performance comparison of different models on function approximation with a contamination rate of 10%

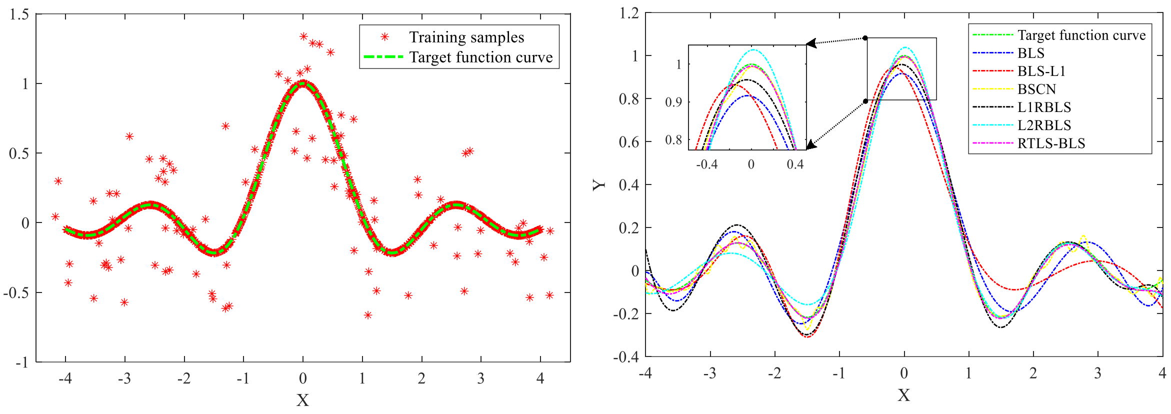
y	Algorithms	MAE±STD	RMSE±STD	MSE±STD	R^2±STD
y_1	BLS	0.0494±0.0043	0.0568±0.0046	0.0033±0.0005	0.9938±0.0010
	BLS-L1	0.0453±0.0031	0.0562±0.0029	0.0032±0.0003	0.9939±0.0006
	BSCN	0.0499±0.0012	0.0744±0.0025	0.0055±0.0004	0.9894±0.0007
	L1RBLS	0.0428±0.0029	0.0554±0.0040	0.0031±0.0005	0.9941±0.0009
	L2RBLS	0.0425±0.0023	0.0543±0.0023	0.0030±0.0003	0.9943±0.0005
	RTLS-BLS	**0.0418±0.0014**	**0.0527±0.0013**	**0.0028±0.0001**	**0.9947±0.0003**
y_2	BLS	0.0127±0.0087	0.0156±0.0094	0.0003±0.0005	0.9968±0.0044
	BLS-L1	0.0185±0.0108	0.0230±0.0134	0.0007±0.0010	0.9933±0.0096
	BSCN	0.0120±0.0009	0.0157±0.0012	0.0002±0.0001	0.9976±0.0003
	L1RBLS	0.0125±0.0047	0.0158±0.0058	0.0003±0.0001	0.9973±0.0024
	L2RBLS	0.0101±0.0031	0.0130±0.0036	0.0002±0.0001	0.9983±0.0015
	RTLS-BLS	**0.0086±0.0028**	**0.0115±0.0034**	**0.0001±0.0001**	**0.9986±0.0011**

Additionally, Fig. 4 and Fig. 5 display the regression results for the function approximation problem at a pollution rate of 10%. From these figures, it is observed that RTLS-BLS demonstrates a stronger fitting effect for the problem. This enhanced performance is attributed to the integration of the BLS-based autoencoder with the k-support norm, which enhances the robustness of feature extraction. Furthermore, the use of regularized total least squares to calculate output weights facilitates a better fit for the EIV model.



(a) Training samples and the target outputs (b) Test performance of the different algorithms

Fig. 4. Curves related to function y_1 with the contamination rate 10%



(a) Training samples and the target outputs (b) Test performance of the different algorithms

Fig. 5. Curves related to function y_2 with the contamination rate 10%

Performance on the Real-World Datasets. The test results for MAE, RMSE, MSE, and R^2 values, obtained using various algorithms across different datasets and pollution rates, are presented in Table 6 to Table 9, with the best results highlighted in bold. From these tables, the following conclusions can be drawn:

(1) Excellent performance has been achieved by the proposed RTLS-BLS among these algorithms. It has been found that RTLS-BLS achieved the lowest MAE, RMSE, MSE, and the highest R^2 across these datasets, indicating strong approximation capability for sample data with noise present in both input and output values.

(2) Higher values of MSE and RMSE, along with lower R^2 values, were yielded by the standard BLS and BSCN models compared to other models. These findings suggest that both the standard BLS and BSCN models exhibit sensitivity to noise and outliers.

(3) The average test performance of L1RBLS, L2RBLS, and BLS-L1 on the four evaluation indicators are consistently better than the standard BLS and BSCN models across different benchmark datasets. The observed phenomenon implies that the L_1 loss function is more robust and is generally less susceptible to the influence of outliers compared with the L_2 loss function.

Based on the above analysis, we can conclude that among all the compared methods, the proposed RTLS-BLS performs well for regression tasks with noisy input and output values.

In Fig. 6, we comprehensively present the performance evaluation results of six algorithms across eight datasets with varying degrees of anomalies. It is evident that our proposed RTLS-BLS algorithm significantly outperforms the other comparative methods in all test cases. The exceptional performance of RTLS-BLS is primarily attributed to its innovative design, which skillfully utilizes regularized total least squares to estimate output weights. This process not only thoroughly considers the impact of output noise Δ*y* but also, for the first time, incorporates considerations of input noise Δ*x*. This allows the RTLS-BLS algorithm to more precisely adapt to the EIV model, thereby displaying superior fitting capabilities and robustness when dealing with complex datasets containing noise in both inputs and outputs. In contrast, other algorithms, focusing solely on handling output noise Δ*y*, exhibit inadequate performance under such complex conditions, failing to achieve satisfactory results.

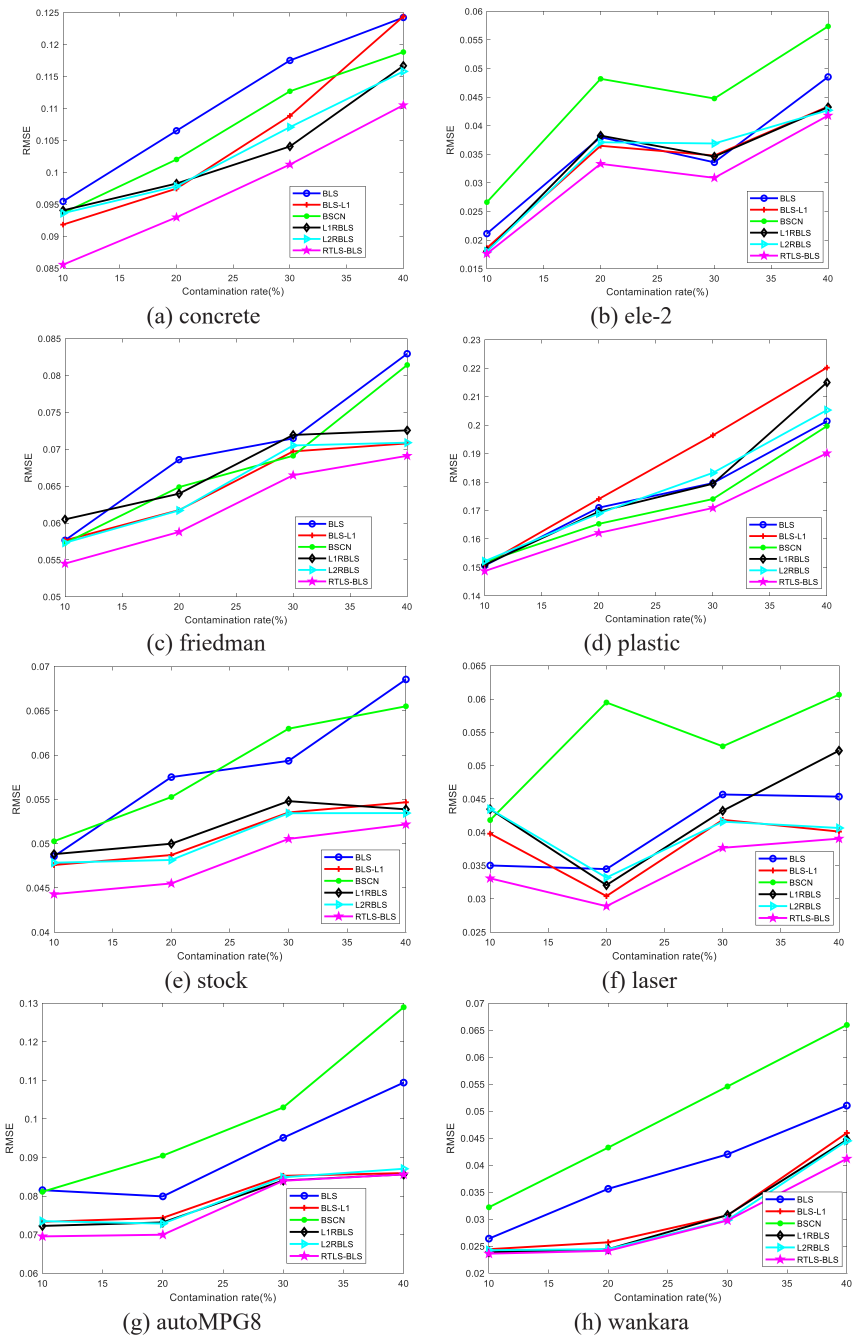


Fig. 6. Comparison testing RMSE on 8 data sets with different outlier level

To further demonstrate the advantages of RTLS-BLS, Fig. 7 clearly presents a comparison of the average test RMSE for six algorithms across eight datasets with a contamination rate of 30%. The results show that our proposed RTLS-BLS algorithm not only achieves lower RMSE values but also demonstrates faster convergence. This advantage is primarily due to the KSP-BLS-AE feature extraction method employed in the RTLS-BLS algorithm. Compared to the sparse autoencoders commonly used by other algorithms for feature extraction, KSP-BLS-AE leverages its unique advantages to extract more robust features. Specifically, the *k*-support norm mechanism built into KSP-BLS-AE effectively overcomes the limitations of traditional L_1 norms in terms of sparsity and/or low-rank characteristics, enabling the extraction of features that are not only sparse but also more representative. This capability better addresses noise and outliers in data. It is this feature that endows the RTLS-BLS algorithm with more efficient feature learning capabilities on complex datasets, thereby achieving faster convergence and lower prediction errors.

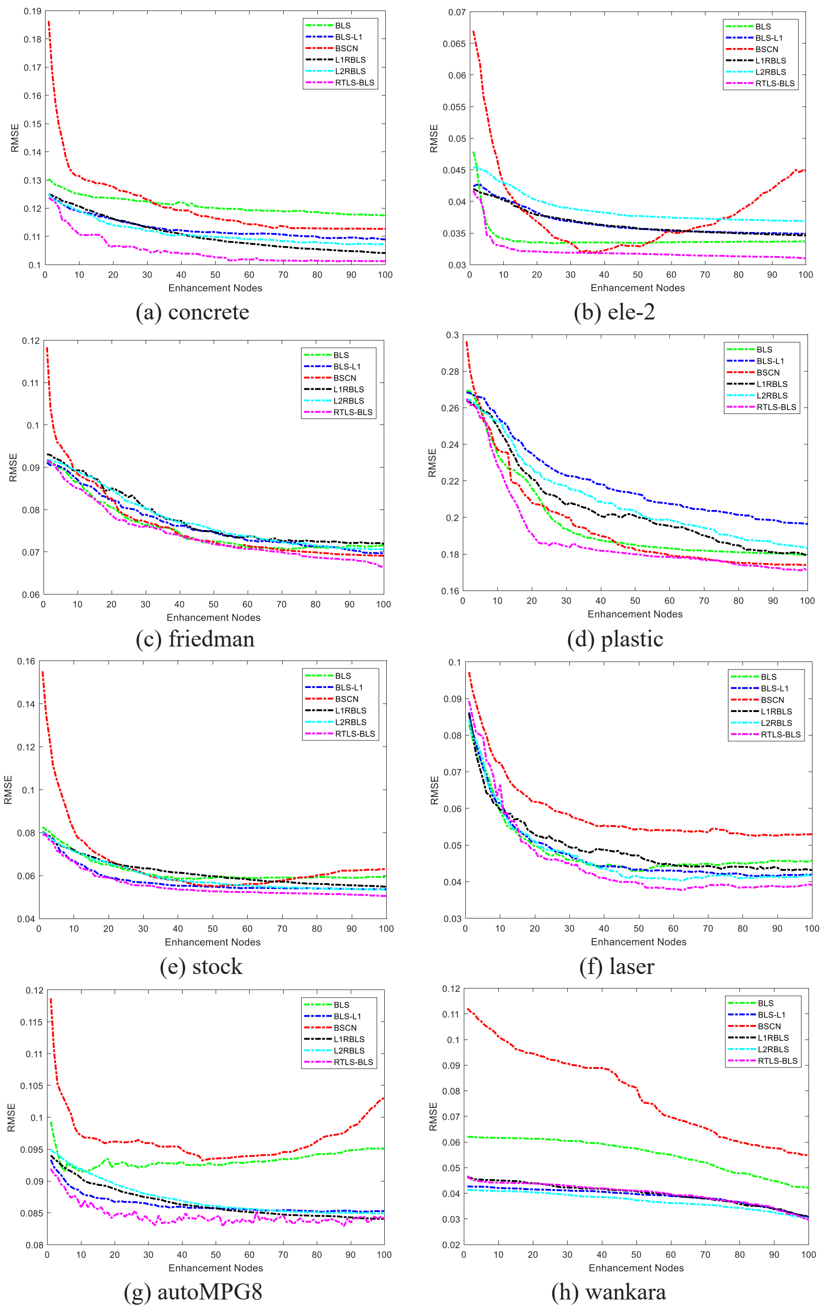


Fig. 7. Average test performance on regression datasets (Contamination rates=30%)

To comprehensively validate the efficacy of the RTLS-BLS algorithm, Table 10 is meticulously designed to compare RTLS-BLS with a series of other BLS variants on a benchmark dataset with a 10% contamination rate. In this experiment, the maximum number of enhancement nodes was set to 300. The desired error tolerance ε was adjusted based on the specific conditions of different datasets. Training was terminated upon reaching either the desired error tolerance or the maximum number of enhancement nodes. From the data in Table 10, it is observed, as expected, that different algorithms exhibit significant variations in training time and the number of nodes in the enhancement layers when faced with diverse datasets. Notably, the RTLS-BLS algorithm consistently displays the most streamlined configuration of hidden layer nodes across all evaluated datasets, directly reflecting its network's compactness and efficiency. Although the training time of RTLS-BLS is slightly longer due to the incorporation of the KSP-BLS-AE algorithm to enhance feature extraction robustness, this minor time cost is undoubtedly justified by its significant advantages in improving network efficiency and reducing model complexity. Overall, RTLS-BLS not only performs excellently in terms of efficiency but also provides robust support for handling complex data with its more compact network structure.

Ablation Studies. To evaluate the contribution of various improvement components within the RTLS-BLS model to the overall performance, ablation experiments were conducted on three datasets: concrete, friedman, and autoMPG8. RTLS-BLS, an enhanced version of the standard BLS, focuses its core improvements on the integration of the KSP-BLS-AE feature extraction module and the R-TLS output weight computation module. The experimental results, summarized in Table 11, clearly reveal several key findings: First, the integration of the KSP-BLS-AE module alone into BLS, leveraging its robust feature learning capabilities, significantly reduces the Root Mean Square Error (RMSE) compared to the standard BLS model, demonstrating the direct contribution of feature optimization to performance enhancement. Secondly, applying the R-TLS module alone to optimize output weight calculations in BLS also achieves superior performance relative to the basic BLS model, highlighting the effectiveness of the weight optimization strategy. Notably, although the individual applications of KSP-BLS-AE and R-TLS each enhance BLS performance, their combined application in the RTLS-BLS model results in a more significant performance leap, surpassing the effects achievable by using either module alone. These results not only validate the rationality of the RTLS-BLS design concept but also fully demonstrate the tremendous potential for overall performance improvement when its internal components work synergistically. Overall, the results of the ablation experiments robustly support the significant contributions of the improvement components in the RTLS-BLS model, showcasing its advantages and potential in handling Errors-in-Variables (EIV) models.

Forecasting Network Interface Flow. The experiment focuses on forecasting Network Interface Flow using two distinct datasets: the core network traffic dataset of European cities and the academic backbone network traffic dataset of the UK.

The core network traffic dataset of European cities captures the bit flow on the transatlantic link between 11 European cities during the period from June 7, 2005, 06:57, to July 31, 2005, 11:17. The data was sampled at a frequency of every *5* minutes. For the experiment, we divided the data into a training set (July 1 to July 19, 2005) and a test set (July 20 to July 28, 2005). In the experiment, the flow values within the time interval [T-11, T] were utilized as features to predict the flow at time T+1.

The UK academic backbone traffic dataset encompasses the aggregated flow of the UK academic network backbone spanning from November 19, 2004, 09:30, to January 27, 2005, 11:11. Sampling occurred at 5-minute intervals. For our experiment, we employed the data from January 1 to January 19, 2005, as the training set and the data from January 20 to January 27, 2005, as the test set. The flow values within the time range [T-11, T] were employed as features to predict the flow at time T+1 in the experiment.

Table 12 and Table 13 present the test results obtained from the core network traffic dataset of European cities and the academic backbone network traffic dataset of the UK, respectively. These results represent the average values and standard deviations of various evaluation metrics. Upon examining Table 12 and Table 13, it is observed that RTLS-BLS achieves the lowest MAE, MSE, RMSE, and the highest R2 value among all models. This performance indicates that RTLS-BLS excels in forecasting network interface flow compared to other models.

Performance Analysis. The experiments cited above demonstrate that the proposed RTLS-BLS model has achieved commendable results in handling errors-in-variables (EIV) models where both input and output values are affected by noise. This success is attributed to the utilization of the KSP-BLS-AE within RTLS-BLS for feature extraction, which facilitates the extraction of more effective features and enhances robustness against noise. Moreover, the use of regularized total least squares to evaluate the output weights of the hidden layer in RTLS-

BLS leads to improved fitting results for EIV models. In contrast to the least squares method, which solely considers noise in the output values, regularized total least squares account for noise in both input and output values, enabling RTLS-BLS to achieve superior fitting results for EIV models.

5 Conclusion

This paper proposes the RTLS-BLS to address the errors-in-variables (EIV) model in the presence of noisy input and output values. The approximation problem of the EIV model is explored, and the utilization of the RTLS-BLS algorithm is examined. The proposed method employs a *k*-support norm regularized BLS-based Autoencoder for feature extraction, adopts the BLS approach for selecting hidden layer weights, and utilizes regularized total least squares to compute the output weight, accounting for perturbations in both the hidden output matrix and output vector. Experimental results obtained from various function approximation problems and real-world datasets demonstrate the favorable performance of the proposed RTLS-BLS method in handling regression tasks with noisy input and output.

Table 6. Test MAE values of all algorithms on different datasets

Datasets	Algorithms	Contamination rates (MAE±STD ×10^{-2})			
		10%	20%	30%	40%
concrete	BLS	6.94±0.31	7.74±0.37	8.76±0.40	9.35±0.40
	BLS-L1	6.89±0.30	7.54±0.24	8.39±0.24	9.81±0.42
	BSCN	6.99±0.35	7.81±0.33	8.51±0.43	9.08±0.38
	L1RBLS	7.07±0.34	7.68±0.24	8.24±0.27	9.17±0.29
	L2RBLS	7.05±0.34	7.60±0.23	8.35±0.35	8.96±0.25
	RTLS-BLS	**6.41±0.37**	**7.16±0.28**	**7.91±0.34**	**8.63±0.29**
ele-2	BLS	1.59±0.10	2.96±0.12	2.63±0.09	3.79±0.13
	BLS-L1	1.48±0.08	2.86±0.12	2.83±0.14	3.28±0.09
	BSCN	1.90±0.14	3.26±0.18	2.99±0.23	4.06±0.29
	L1RBLS	1.45±0.05	2.94±0.25	2.86±0.07	3.32±0.25
	L2RBLS	1.46±0.05	2.81±0.16	3.13±0.16	3.29±0.11
	RTLS-BLS	**1.34±0.06**	**2.56±0.13**	**2.42±0.12**	**3.20±0.18**
friedman	BLS	4.45±0.16	5.25±0.26	5.38±0.23	6.21±0.37
	BLS-L1	4.47±0.21	4.89±0.22	5.28±0.24	5.58±0.19
	BSCN	4.43±0.16	5.05±0.20	5.17±0.20	6.26±0.24
	L1RBLS	4.68±0.23	5.04±0.26	5.56±0.24	5.75±0.32
	L2RBLS	4.45±0.16	4.88±0.21	5.44±0.20	5.57±0.19
	RTLS-BLS	**4.24±0.15**	**4.63±0.18**	**5.05±0.13**	**5.44±0.22**
plastic	BLS	12.02±0.05	14.18±0.67	15.28±0.35	17.12±0.42
	BLS-L1	12.03±0.10	14.41±0.53	16.71±1.04	18.80±1.25
	BSCN	12.24±0.12	13.51±0.12	14.61±0.20	16.50±0.26
	L1RBLS	12.04±0.13	13.87±0.44	15.07±0.86	18.27±1.67
	L2RBLS	12.19±0.22	13.84±0.40	15.46±0.83	17.36±1.30
	RTLS-BLS	**12.03±0.06**	**13.37±0.07**	**14.47±0.25**	**16.10±0.33**
stock	BLS	3.75±0.16	4.42±0.18	4.54±0.25	5.11±0.18
	BLS-L1	3.67±0.16	3.79±0.15	4.06±0.15	4.32±0.17
	BSCN	3.89±0.18	4.31±0.24	4.73±0.29	4.99±0.25
	L1RBLS	3.81±0.16	3.94±0.19	4.24±0.15	4.27±0.19
	L2RBLS	3.72±0.17	3.78±0.13	4.09±0.16	4.21±0.19
	RTLS-BLS	**3.44±0.14**	**3.55±0.20**	**3.89±0.14**	**4.13±0.16**
laser	BLS	**2.09±0.17**	2.09±0.17	2.62±0.22	3.53±0.19
	BLS-L1	2.61±0.23	2.02±0.16	2.56±0.25	2.49±0.18
	BSCN	2.43±0.22	3.06±0.49	2.97±0.33	3.61±0.29
	L1RBLS	2.93±0.28	2.12±0.21	2.61±0.35	3.19±0.40
	L2RBLS	3.00±0.24	2.17±0.27	2.52±0.27	2.48±0.17
	RTLS-BLS	2.16±0.17	**1.99±0.13**	**2.36±0.16**	**2.37±0.10**
autoMPG8	BLS	5.91±0.35	6.00±0.39	7.00±0.37	7.65±0.41
	BLS-L1	5.46±0.16	5.72±0.26	6.22±0.18	5.93±0.22
	BSCN	6.10±0.32	6.76±0.41	7.63±0.48	9.24±0.80
	L1RBLS	5.47±0.09	5.62±0.20	**6.12±0.11**	5.92±0.20
	L2RBLS	5.46±0.17	5.42±0.14	6.22±0.15	6.03±0.12
	RTLS-BLS	**5.22±0.18**	**5.33±0.17**	6.24±0.26	**5.96±0.32**
wankara	BLS	1.96±0.08	2.63±0.23	3.03±0.23	4.02±0.20
	BLS-L1	1.85±0.07	1.96±0.12	2.34±0.12	3.73±0.14
	BSCN	2.12±0.14	2.96±0.12	3.76±0.19	4.34±0.24
	L1RBLS	1.82±0.06	1.91±0.11	2.37±0.15	3.56±0.16
	L2RBLS	1.85±0.06	1.92±0.06	**2.30±0.06**	3.63±0.10
	RTLS-BLS	**1.72±0.06**	**1.86±0.08**	**2.26±0.10**	**3.32±0.11**

Table 7. Test RMSE values of all algorithms on different datasets

Datasets	Algorithms	Contamination rates (RMSE±STD ×10^{-2})			
		10%	20%	30%	40%
concrete	BLS	9.55±0.52	10.65±0.99	11.75±0.74	12.42±0.66
	BLS-L1	9.18±0.38	9.75±0.34	10.89±0.53	12.44±0.52
	BSCN	9.36±0.49	10.20±0.50	11.27±0.86	11.88±0.53
	L1RBLS	9.41±0.45	9.83±0.28	10.41±0.32	11.67±0.35
	L2RBLS	9.36±0.44	9.78±0.27	10.71±0.49	11.58±0.32
	RTLS-BLS	**8.56±0.53**	**9.30±0.33**	**10.12±0.49**	**11.05±0.33**
ele-2	BLS	2.11±0.17	3.80±0.20	3.36±0.14	4.85±0.21
	BLS-L1	1.86±0.08	3.65±0.15	3.48±0.13	4.34±0.17
	BSCN	2.66±0.32	4.82±0.18	4.48±0.45	5.75±0.54
	L1RBLS	1.81±0.06	3.83±0.38	3.46±0.07	4.32±0.48
	L2RBLS	1.80±0.05	3.71±0.24	3.69±0.16	4.27±0.22
	RTLS-BLS	**1.76±0.09**	**3.33±0.14**	**3.09±0.13**	**4.18±0.21**
friedman	BLS	5.77±0.21	6.86±0.42	7.15±0.32	8.30±0.73
	BLS-L1	5.76±0.28	6.18±0.29	6.97±0.33	7.08±0.23
	BSCN	5.73±0.22	6.49±0.29	6.91±0.29	8.14±0.36
	L1RBLS	6.05±0.31	6.40±0.34	7.19±0.35	7.26±0.39
	L2RBLS	5.73±0.24	6.17±0.27	7.05±0.25	7.09±0.22
	RTLS-BLS	**5.45±0.21**	**5.88±0.26**	**6.65±0.13**	**6.91±0.26**
plastic	BLS	15.07±0.09	17.10±0.68	17.96±0.41	20.14±0.47
	BLS-L1	15.11±0.12	17.41±0.55	19.64±1.17	22.02±1.39
	BSCN	15.19±0.12	16.53±0.13	17.40±0.26	19.97±0.39
	L1RBLS	15.09±0.13	16.96±0.48	17.94±1.00	21.50±1.84
	L2RBLS	15.22±0.25	16.90±0.44	18.32±0.98	20.53±1.45
	RTLS-BLS	**14.87±0.04**	**16.21±0.07**	**17.09±0.28**	**19.01±0.32**
stock	BLS	4.86±0.22	5.75±0.24	5.93±0.41	6.86±0.35
	BLS-L1	4.76±0.19	4.87±0.18	5.35±0.19	5.47±0.18
	BSCN	5.03±0.24	5.53±0.30	6.30±0.57	6.55±0.33
	L1RBLS	4.88±0.19	5.00±0.24	5.48±0.19	5.39±0.21
	L2RBLS	4.78±0.22	4.82±0.15	5.34±0.21	5.35±0.20
	RTLS-BLS	**4.43±0.19**	**4.55±0.22**	**5.05±0.20**	**5.22±0.20**
laser	BLS	3.50±0.36	3.45±0.71	4.57±0.43	4.53±0.39
	BLS-L1	3.98±0.34	3.04±0.36	4.19±0.40	4.01±0.21
	BSCN	4.18±0.58	5.95±1.81	5.29±0.96	6.06±1.07
	L1RBLS	4.35±0.46	3.21±0.46	4.32±0.54	5.22±0.62
	L2RBLS	4.35±0.30	3.32±0.51	4.16±0.41	4.06±0.30
	RTLS-BLS	**3.31±0.18**	**2.89±0.15**	**3.77±0.27**	**3.90±0.16**
autoMPG8	BLS	8.15±0.66	7.99±0.47	9.51±0.51	10.94±0.62
	BLS-L1	7.33±0.23	7.44±0.33	8.52±0.21	8.60±0.25
	BSCN	8.11±0.49	9.05±0.33	10.30±0.91	12.90±1.32
	L1RBLS	7.23±0.12	7.32±0.23	**8.40±0.15**	**8.56±0.24**
	L2RBLS	7.35±0.23	7.29±0.18	8.49±0.17	8.71±0.18
	RTLS-BLS	**6.95±0.18**	**7.00±0.26**	**8.40±0.23**	**8.56±0.34**
wankara	BLS	2.64±0.15	3.56±0.39	4.20±0.46	5.10±0.25
	BLS-L1	2.44±0.12	2.57±0.15	3.07±0.16	4.60±0.19
	BSCN	3.22±0.83	4.33±0.79	5.46±0.62	6.60±1.02
	L1RBLS	2.39±0.07	2.45±0.12	3.08±0.21	4.47±0.19
	L2RBLS	2.43±0.08	2.45±0.07	**2.98±0.09**	4.45±0.11
	RTLS-BLS	**2.36±0.09**	**2.41±0.09**	**2.98±0.13**	**4.12±0.09**

Table 8. Test MSE values of all algorithms on different datasets

Datasets	Algorithms	Contamination rates (MSE±STD ×10^{-3})			
		10%	20%	30%	40%
concrete	BLS	9.14±1.00	11.44±2.41	13.87±1.78	15.48±1.65
	BLS-L1	8.45±0.69	9.51±0.66	11.92±1.20	15.50±1.31
	BSCN	8.81±0.93	10.44±1.02	12.83±2.01	14.10±1.22
	L1RBLS	8.94±0.86	9.71±0.55	10.82±0.67	13.60±0.81
	L2RBLS	8.79±0.83	9.63±0.54	11.51±1.11	13.40±0.74
	RTLS-BLS	**7.33±0.89**	**8.72±0.61**	**10.32±0.99**	**12.19±0.72**
ele-2	BLS	0.45±0.08	1.44±0.15	1.13±0.09	2.36±0.21
	BLS-L1	0.35±0.03	1.33±0.11	1.21±0.09	1.88±0.15
	BSCN	0.72±0.18	2.37±0.70	2.02±0.42	3.32±0.65
	L1RBLS	0.33±0.02	1.48±0.30	1.20±0.05	1.89±0.47
	L2RBLS	0.33±0.02	1.38±0.18	1.36±0.11	1.83±0.20
	RTLS-BLS	**0.31±0.03**	**1.11±0.09**	**0.96±0.08**	**1.74±0.18**
friedman	BLS	3.33±0.24	4.71±0.58	5.10±0.46	6.93±1.31
	BLS-L1	3.32±0.33	3.81±0.36	4.92±.46	5.01±0.32
	BSCN	3.33±0.26	4.21±0.38	4.83±0.40	6.65±0.58
	L1RBLS	3.73±0.38	4.12±0.44	5.16±0.51	5.30±0.57
	L2RBLS	3.33±0.28	3.80±0.33	5.00±0.36	5.04±0.57
	RTLS-BLS	**3.01±0.23**	**3.46±0.31**	**4.40±0.18**	**4.83±0.36**
plastic	BLS	22.71±0.28	29.30±2.42	32.32±1.53	40.59±1.94
	BLS-L1	22.80±0.26	30.31±1.90	38.74±4.63	48.72±6.11
	BSCN	23.11±0.35	27.33±0.44	30.30±0.89	39.94±1.61
	L1RBLS	22.81±0.38	28.82±1.73	32.33±3.74	46.60±7.92
	L2RBLS	23.21±0.76	28.61±1.52	33.71±3.73	42.40±6.02
	RTLS-BLS	**22.11±0.11**	**26.60±0.23**	**29.24±0.96**	**36.19±1.21**
stock	BLS	2.42±0.21	3.31±0.28	3.53±0.52	4.69±0.49
	BLS-L1	2.30±0.19	2.40±0.18	2.91±0.21	3.00±0.20
	BSCN	2.51±0.24	3.14±0.34	4.03±0.81	4.34±0.44
	L1RBLS	2.43±0.19	2.48±0.24	3.01±0.21	2.94±0.23
	L2RBLS	2.33±0.21	2.32±0.15	2.91±0.23	2.86±0.21
	RTLS-BLS	**1.97±0.17**	**2.14±0.20**	**2.63±0.20**	**2.71±0.21**
laser	BLS	1.18±0.26	1.24±0.57	2.12±0.40	2.07±0.36
	BLS-L1	1.60±0.27	0.94±0.23	1.82±0.34	1.60±0.17
	BSCN	1.80±0.51	4.54±1.31	2.91±1.11	3.83±1.61
	L1RBLS	1.96±0.41	1.00±0.32	1.92±0.49	2.80±0.68
	L2RBLS	1.92±0.26	1.10±0.36	1.74±0.36	1.70±0.25
	RTLS-BLS	**1.10±0.12**	**0.84±0.09**	**1.41±0.20**	**1.52±0.13**
autoMPG8	BLS	6.69±1.14	6.41±0.76	9.07±0.98	12.01±1.35
	BLS-L1	5.38±0.35	5.54±0.49	7.27±0.36	7.39±0.43
	BSCN	6.60±0.80	8.23±1.24	10.69±2.18	16.80±3.47
	L1RBLS	5.22±0.17	5.36±0.33	**7.07±0.26**	**7.33±0.41**
	L2RBLS	5.40±0.34	5.31±0.27	7.21±0.28	7.59±0.31
	RTLS-BLS	**4.84±0.25**	**4.90±0.36**	**7.06±0.39**	**7.33±0.59**
wankara	BLS	0.70±0.08	1.28±0.28	1.79±0.40	2.61±0.26
	BLS-L1	0.60±0.06	0.66±0.08	0.95±0.10	2.12±0.18
	BSCN	1.11±0.79	1.93±1.03	3.02±0.78	4.46±7.73
	L1RBLS	**0.57±0.03**	0.60±0.06	0.95±0.13	2.00±0.17
	L2RBLS	0.59±0.04	0.60±0.03	**0.89±0.05**	1.98±0.10
	RTLS-BLS	**0.56±0.04**	**0.58±0.04**	**0.88±0.07**	**1.70±0.08**

Table 9. Test R^2 values of all algorithms on different datasets

Datasets	Algorithms	Contamination rates (R^2±STD $\times 10^{-2}$)			
		10%	20%	30%	40%
concrete	BLS	80.09±2.17	74.97±5.27	68.28±4.06	66.91±3.53
	BLS-L1	81.60±1.50	79.20±1.44	72.83±2.65	66.86±2.80
	BSCN	80.85±2.03	77.18±2.23	70.78±4.66	69.75±2.67
	L1RBLS	80.69±1.87	78.86±1.21	75.20±1.53	70.87±1.73
	L2RBLS	80.87±1.82	79.05±1.17	73.71±2.44	71.31±1.58
	RTLS-BLS	**84.00±1.94**	**81.07±1.34**	**76.50±2.26**	**73.87±1.53**
ele-2	BLS	99.01±0.17	97.01±0.31	97.44±0.21	94.42±0.50
	BLS-L1	99.24±0.07	97.25±0.23	97.25±0.21	95.55±0.36
	BSCN	98.42±0.40	95.12±1.44	95.41±0.95	92.16±1.54
	L1RBLS	99.28±0.05	96.95±0.61	97.28±0.11	95.53±1.12
	L2RBLS	99.28±0.04	97.15±0.38	96.91±0.24	95.68±0.47
	RTLS-BLS	**99.32±0.07**	**97.70±0.19**	**97.88±0.18**	**95.87±0.42**
friedman	BLS	89.43±0.77	85.22±1.82	85.96±1.26	80.28±3.60
	BLS-L1	89.42±1.04	88.04±1.11	86.64±1.27	85.73±0.92
	BSCN	89.56±0.82	86.80±1.19	86.87±1.10	81.11±1.66
	L1RBLS	88.34±1.21	87.15±1.37	85.77±1.39	84.98±1.63
	L2RBLS	89.54±0.88	88.06±1.03	86.34±0.98	85.69±0.88
	RTLS-BLS	**90.55±0.73**	**89.16±0.97**	**87.87±0.50**	**86.40±1.02**
plastic	BLS	79.75±0.25	74.64±2.06	73.45±1.23	66.63±1.56
	BLS-L1	79.63±0.32	73.74±1.66	68.16±3.78	59.96±5.04
	BSCN	79.42±0.32	76.33±0.38	75.09±0.74	67.20±1.28
	L1RBLS	79.68±0.34	75.07±1.43	73.45±3.02	61.71±6.49
	L2RBLS	79.33±0.68	75.26±1.28	72.31±3.01	65.17±4.97
	RTLS-BLS	**80.27±0.10**	**77.24±0.20**	**75.98±0.79**	**70.27±1.02**
stock	BLS	95.53±0.40	93.87±0.52	93.55±0.94	91.57±0.87
	BLS-L1	95.71±0.35	95.61±0.33	94.77±0.38	94.65±0.35
	BSCN	95.21±0.45	94.34±0.63	92.71±1.48	92.30±0.79
	L1RBLS	95.49±0.36	95.37±0.45	94.52±0.38	94.80±0.41
	L2RBLS	95.67±0.40	95.71±0.27	94.79±0.42	94.88±0.38
	RTLS-BLS	**96.29±0.32**	**96.16±0.36**	**95.34±0.36**	**95.12±0.38**
laser	BLS	96.46±0.75	96.05±1.81	92.71±1.39	94.19±1.02
	BLS-L1	95.45±0.78	97.01±0.73	93.87±1.18	95.48±0.47
	BSCN	94.91±1.47	85.49±2.03	89.98±3.82	89.36±4.51
	L1RBLS	94.55±1.18	96.65±1.03	93.43±1.69	92.23±1.91
	L2RBLS	94.58±0.75	96.40±1.14	93.95±1.26	95.34±0.69
	RTLS-BLS	**96.87±0.33**	**97.33±0.28**	**95.06±0.70**	**95.72±0.35**
autoMPG8	BLS	82.96±2.92	84.38±1.85	77.23±2.47	71.91±3.15
	BLS-L1	85.30±0.85	86.49±1.19	81.75±0.89	82.71±1.02
	BSCN	83.19±2.05	79.93±3.02	73.17±5.48	60.71±8.11
	L1RBLS	86.70±0.43	86.93±0.81	**82.27±0.65**	**82.85±0.97**
	L2RBLS	86.23±0.87	87.05±0.65	81.92±0.71	82.26±0.72
	RTLS-BLS	**87.68±0.64**	**88.05±0.88**	**82.29±0.97**	82.84±1.38
wankara	BLS	98.43±0.19	97.12±0.64	96.04±0.88	93.81±0.62
	BLS-L1	98.65±0.14	98.51±0.17	97.90±0.22	94.98±0.42
	BSCN	97.51±1.77	95.65±2.31	93.31±1.74	89.44±4.09
	L1RBLS	98.71±0.08	98.65±0.14	97.89±0.29	95.25±0.39
	L2RBLS	98.67±0.09	98.66±0.07	**98.02±0.11**	95.30±0.23
	RTLS-BLS	**98.74±0.09**	**98.69±0.09**	**98.03±0.17**	**95.97±0.18**

Table 10. Efficiency comparison on regression databases (Contamination rates=10%)

Datasets	Algorithms	ε	Training time(s)	Enhancement nodes±STD
concrete	BLS	0.115	**0.0163**	22.65±2.395
	BLS-L1		0.1078	18.28±3.725
	BSCN		0.1051	35.60±4.580
	L1RBLS		0.0846	18.58±3.501
	L2RBLS		0.0493	18.66±3.931
	RTLS-BLS		0.0935	**12.96±2.025**
ele-2	BLS	0.02	**0.1168**	53.80±5.852
	BLS-L1		0.4595	55.64±6.620
	BSCN		0.2261	51.70±7.953
	L1RBLS		0.1505	54.62±5.615
	L2RBLS		0.1673	54.30±5.435
	RTLS-BLS		0.3240	**46.10±2.514**
friedman	BLS	0.07	**0.0297**	29.60±9.891
	BLS-L1		0.1855	23.40±5.602
	BSCN		0.1205	25.50±5.312
	L1RBLS		0.0541	27.60±9.294
	L2RBLS		0.0469	26.80±8.522
	RTLS-BLS		0.0773	**16.20±4.045**
plastic	BLS	0.155	0.4989	25.60±7.947
	BLS-L1		1.1045	84.60±5.911
	BSCN		**0.2979**	18.30±6.567
	L1RBLS		0.9081	46.30±2.662
	L2RBLS		0.9237	53.50±3.227
	RTLS-BLS		0.4540	**17.40±1.506**
stock	BLS	0.055	**0.0667**	38.30±6.485
	BLS-L1		0.2314	21.50±9.585
	BSCN		0.2531	55.10±3.479
	L1RBLS		0.0746	31.60±9.178
	L2RBLS		0.0673	27.10±8.945
	RTLS-BLS		0.2483	**14.30±4.322**
laser	BLS	0.045	**0.0421**	39.60±1.368
	BLS-L1		0.2411	36.00±4.422
	BSCN		0.2669	59.40±4.169
	L1RBLS		0.0686	45.70±5.016
	L2RBLS		0.0522	36.10±2.741
	RTLS-BLS		0.0983	**17.40±4.766**
autoMPG8	BLS	0.082	**0.0086**	4.60±2.914
	BLS-L1		0.0712	2.90±2.079
	BSCN		0.1273	44.50±3.866
	L1RBLS		0.0619	6.00±2.000
	L2RBLS		0.0222	6.00±4.615
	RTLS-BLS		0.0644	2.70±0.675
wankara	BLS	0.025	**0.0383**	139±14.153
	BLS-L1		0.5290	36.40±8.637
	BSCN		1.4914	174±16.812
	L1RBLS		0.0925	55.50±4.260
	L2RBLS		0.0497	45.30±3.393
	RTLS-BLS		0.0953	**34.50±5.049**

Table 11. Results of ablation studies

Datasets	Algorithms	Contamination rates (RMSE±STD $\times 10^{-3}$)			
		10%	20%	30%	40%
concrete	Baseline(BLS)	9.55±0.52	10.65±0.99	11.75±0.74	12.42±0.66
	BLS+KSP-BLS-AE	9.03±0.48	10.08±0.45	10.84±0.65	11.79±0.57
	BLS+R-TLS	8.89±0.40	9.85±0.25	10.33±0.27	11.66±0.46
	RTLS-BLS	**8.56±0.53**	**9.30±0.33**	**10.12±0.49**	**11.05±0.33**
friedman	Baseline(BLS)	5.77±0.21	6.86±0.42	7.15±0.32	8.30±0.73
	BLS+KSP-BLS-AE	5.55±0.22	6.55±0.33	6.91±0.25	7.58±0.37
	BLS+R-TLS	5.51±0.29	6.16±0.26	6.80±0.22	7.35±0.27
	RTLS-BLS	**5.45±0.21**	**5.88±0.26**	**6.65±0.13**	**6.91±0.26**
autoMPG8	Baseline(BLS)	8.15±0.66	7.99±0.47	9.51±0.51	10.94±0.62
	BLS+KSP-BLS-AE	7.60±0.20	7.53±0.27	8.92±0.31	9.05±0.53
	BLS+R-TLS	7.42±0.16	7.37±0.36	8.71±0.23	8.77±0.43
	RTLS-BLS	**6.95±0.18**	**7.00±0.26**	**8.40±0.23**	**8.56±0.34**

Table 12. Performance comparison of different models on a core network traffic dataset in European cities

Algorithms	MAE±STD	RMSE±STD	MSE±STD	R^2±STD
BLS	0.0180±0.0004	0.0248±0.0006	0.0006±0	0.9928±0.0006
BLS-L1	0.0172±0.0008	0.0235±0.0009	0.0006±0	0.9936±0.0009
BSCN	0.0190±0.0007	0.0268±0.0010	0.0007±0.0001	0.9916±0.0007
L1RBLS	0.0187±0.0016	0.0242±0.0016	0.0006±0.0001	0.9932±0.0009
L2RBLS	0.0179±0.0012	0.0243±0.0015	0.0006±0.0001	0.9931±0.0008
RTLS-BLS	**0.0160±0.0007**	**0.0221±0.0009**	**0.0005±0**	**0.9943±0.0004**

Table 13. Performance comparison of different models on UK academic backbone network traffic dataset

Algorithms	MAE±STD	RMSE±STD	MSE±STD	R^2±STD
BLS	0.0169±0.0008	0.0232±0.0012	0.0005±0.0001	0.9929±0.0007
BLS-L1	0.0168±0.0008	0.0219±0.0009	0.0005±0	0.9937±0.0004
BSCN	0.0183±0.0018	0.0252±0.0035	0.0006±0.0002	0.9915±0.0026
L1RBLS	0.0166±0.0011	0.0219±0.0013	0.0005±0.0001	0.9937±0.0008
L2RBLS	0.0164±0.0010	0.0215±0.0010	0.0005±0	0.9939±0.0006
RTLS-BLS	**0.0158±0.0004**	**0.0206±0.0004**	**0.0004±0**	**0.9944±0.0002**

In future research, several aspects warrant consideration. First, classification tasks are of significant importance in the field of machine learning, and the RTLS-BLS method could be extended to address classification tasks involving noisy input and output values. Second, the determination of parameters for RTLS-BLS currently relies on grid search, a process that can be time-consuming. The identification of an efficient and effective approach to swiftly obtain optimal parameters represents another challenge to be addressed in future studies.

6 Acknowledgement

This work is supported in part by the National Natural Science Foundation of China (Nos. 62373116 and 62163007), Guizhou Provincial Science and Technology Projects (QKHZC[2023]118, PTRC[2020]6007-2), Guizhou Provincial Basic Research Program (Natural Science) (ZK[2024]-035), Natural Science Research Foundation of Guizhou University ([2022]26).

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of the ACM 60(6)(2017) 84–90.

[2] D.W. Otter, J.R. Medina, J.K. Kalita, A survey of the usages of deep learning for natural language processing, IEEE transactions on neural networks and learning systems 32(2)(2021) 604–624.
[3] L. He, B. Ding, H. Wang, T. Zhang, An optimal 3d convolutional neural network based lipreading method, IET Image Processing 16(1)(2022) 113–122.
[4] P. Covington, J. Adams, E. Sargin, Deep neural networks for youtube recommendations, in: Proc. the 10th ACM conference on recommender systems, 2016.
[5] S. Grigorescu, B. Trasnea, T. Cocias, G. Macesanu, A survey of deep learning techniques for autonomous driving, Journal of field robotics 37(3)(2020) 362–386.
[6] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521(7553)(2015) 436–444.
[7] C.P. Chen, Z. Liu, Broad learning system: An effective and efficient incremental learning system without the need for deep architecture, IEEE transactions on neural networks and learning systems 29(1)(2018) 10–24.
[8] C.P. Chen, Z. Liu, S. Feng, Universal approximation capability of broad learning system and its structural variations, IEEE transactions on neural networks and learning systems 30(4)(2019) 1191–1204.
[9] F. Yang, A CNN-based broad learning system, in: Proc. 2018 IEEE 4th International Conference on Computer and Communications, 2018.
[10] T. Li , B. Fang, J. Qian, X. Wu, CNN-based broad learning system, in: Proc. 2019 IEEE 4th International Conference on Signal and Image Processing, 2019.
[11] S. Feng, C.P. Chen, Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification, IEEE Transactions on Cybernetics 50(2)(2020) 414–424.
[12] S. Huang, Z. Liu, W. Jin, Y. Mu, Broad learning system with manifold regularized sparse features for semi-supervised classification, Neurocomputing 463(2021) 133–143.
[13] W. Ding, Y. Tian, S. Han, H. Yuan, Greedy broad learning system, IEEE Access 9(2021) 79307–79315.
[14] L. Zhang, J. Li, G. Lu, P. Shen, M. Bennamoun, S.A.A. Shah, Analysis and variants of broad learning system, IEEE Transactions on Systems, Man, and Cybernetics: Systems 52(1)(2022) 334–344.
[15] H. Zhao, J. Zheng, W. Deng, Y. Song, Semi-supervised broad learning system based on manifold regularization and broad network, IEEE Transactions on Circuits and Systems I: Regular Papers 67(3)(2020) 983–994.
[16] W. Fan, Y. Si, W. Yang, M. Sun, Class-specific weighted broad learning system for imbalanced heartbeat classification, Information Sciences 610(2022) 525–548.
[17] Y. Zheng, B. Chen, S. Wang, W. Wang, Broad learning system based on maximum correntropy criterion, IEEE Transactions on Neural Networks and Learning Systems 32(7)(2021) 3083–3097.
[18] P. Huang, B. Chen, Bidirectional broad learning system, in: Proc. 2020 IEEE 7th International Conference on Industrial Engineering and Applications, 2020.
[19] X.-N. Fan, S.-W. Zhang, Lpi-bls: Predicting lncrna–protein interactions with a broad learning system-based stacked ensemble classifier, Neurocomputing 370(2019) 88–93.
[20] M. Wang, Q. Ge, H. Jiang, G. Yao, Wear fault diagnosis of aeroengines based on broad learning system and ensemble learning, Energies 12(24)(2019) 4750.
[21] L. Zhu, C. Lian, Z. Zeng, Y. Su, A broad learning system with ensemble and classification methods for multi-step-ahead wind speed prediction, Cognitive Computation 12(2020) 654–666.
[22] S. Issa, Q. Peng, X. You, Emotion classification using eeg brain signals and the broad learning system, IEEE Transactions on Systems, Man, and Cybernetics: Systems 51(12)(2021) 7382–7391.
[23] D. Liu, S. Baldi, W. Yu, J. Cao, W. Huang, On training traffic predictors via broad learning structures: A benchmark study, IEEE Transactions on Systems, Man, and Cybernetics: Systems 52(2)(2022) 749–758.
[24] J. Jin, Z. Liu, C.P. Chen, Discriminative graph regularized broad learning system for image recognition, Science China Information Sciences 61(2018) 1–14.
[25] S. Feng, W. Ren, M. Han, Y. W. Chen, Robust manifold broad learning system for large-scale noisy chaotic time series prediction: A perturbation perspective, Neural Networks 117(2019) 179–190.
[26] J.-W. Jin, C.P. Chen, Regularized robust broad learning system for uncertain data modeling, Neurocomputing 322(2018) 58–69.
[27] W. Guo, T. Xu, M-estimator-based robust broad learning system, Control and Decision, 38(4)(2023) 1039–1046.
[28] M. Gan, H.-T. Zhu, G.-Y. Chen, C.L.P. Chen, Weighted generalized cross-validation-based regularization for broad learning system, IEEE Transactions on Cybernetics 52(5)(2022) 4064–4072.
[29] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313(5786)(2006) 504–507.
[30] K. Yang, Y. Liu, Z. Yu, C.L.P. Chen, Extracting and composing robust features with broad learning system, IEEE Transactions on Knowledge and Data Engineering 35(4)(2023) 3885–3896.
[31] H. Lai, Y. Pan, C. Lu, Y. Tang, S. Yan, Efficient k-support matrix pursuit, in: Proc. European Conference on Computer Vision, 2014.
[32] A. Argyriou, R. Foygel, N. Srebro, Sparse prediction with the k-support norm, in: Proc. Advances in Neural Information Processing Systems 25, 2012.
[33] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends® in Machine learning, 2011 (1–122).

[34] I. Markovsky, S. Van Huffel, Overview of total least-squares methods, Signal processing 87(10)(2007) 2283–2302.
[35] G.H. Golub, P.C. Hansen, D.P. O'Leary, Tikhonov regularization and total least squares, SIAM journal on matrix analysis and applications 21(1)(1999) 185–194.
[36] C. Zhang, S. Ding, W. Du, Broad stochastic configuration network for regression, Knowledge-Based Systems 243(2022) 108403.