Research on Predicting the Bending Strength of Ceramic Matrix Composites with Process of Incomplete Data

Gao Xiang, Li Guanghui, Tan Rong, and Yao Leijiang

Abstract—**With the rapid development of machine learning, it is possible to use neural networks to build models to predict performance of Ceramic Matrix Composites (CMCs) with raw materials and environments. In the traditional material science engineering, it always took a long time to develop a new CMC. Furthermore, there is still no theoretical basis providing references to design experiments to develop CMCs with ideal performances. This work proposed a model to predict the bending strength of CMCs with a Convolution Neural Network (CNN) using 8 factors considered to affect the bending strength of CMCs mainly. For the data were all collected from papers published on journals and conferences, and there is no standard to describe an experiment, the incompleteness of data influences the performance of our model seriously. Then we tried several methods to fill the data, finally the regression imputation with a dual-hidden-layer neural network performed a significant improvement of the CNN bending strength prediction model.**

Index Terms—**Bending strength, Ceramic Matrix Composites (CMC), Convolution Neural Network (CNN), imputation.**

I. Introduction

Ceramic matrix composites (CMCs) are a subgroup of composite materials, which consist of various fibers embedded in a ceramic matrix. They not only possess the exceptional properties of ceramic such as the resistance to high temperature, the high strength and rigidity, the low density and the resistance of corrosion, but also improve the disadvantages like poor toughness and reliability. CMCs have great potential and have been applied in the areas of aeronautics, astronautics, automotive industry, *et al*.

In the traditional material science engineering, how to develop a new CMC was always a challenge to scientists, the experience and intuition of whom played an important role in making decisions on choosing the raw materials and environments of experiments. It is because of the lack of theoretical basis, and the development of new CMCs progresses slowly all the time.

Manuscript received July 24, 2019; revised September 11, 2020. This work was supported in part by the National Key R&D Program of China (2016YFB0700500) and the Science and Technology Project of Shaanxi Province (2018GY-048).

Gao Xiang and Li Guanghui are with the School of Computer Science, Northwestern Polytechnical University, Xi'an, Shaanxi 710129 China (e-mail: gaoxg@nwpu.edu.cn, sxlllslgh@mail.nwpu.edu.cn).

Tan Rong is with the School of Software, Northwestern Polytechnical University, Xi'an, Shaanxi 710129 China (e-mail: 2660364434@qq.com).

Yao Leijiang is with the School of Laboratory of Science and Technology onUAV, Northwestern Polytechnical University, Xi'an 710072 China (e-mail: yaolj@nwpu.edu.cn).

In June, 2011, the United States government published Advanced Manufacturing Partnership (AMP) [1], a crucial part of which is Materials Genome Initiative (MGI) [2]. MGI is composed of computational tools, experimental tools and the digital data. The digital data contains all kinds of data and information from posted experiments, which is used to provide the basis of design for high throughput experiments.

In recent years, plenty of machine learning models appeared, which performed much better than traditional models on function fitting. Inspired by this, we tried four common methods to fit the experiment data collected from documents published to predict the bending strength of CMCs, which are support vector machine (SVM) [3], eXtreme Gradient Boosting (XGBoost) [4], deep neural network (DNN) and convolution neural network (CNN) [5].

Owing to the variety of the quality of documents, the lack of experiment factors exists in many documents, which results in the difficulty to train and predict using existing common models. Hence it is requisite to process the missing data. In this work, we tried three different methods to process the incomplete data, which are mean imputation, K-Means clustering imputation and DNN-based regression imputation [6]-[8].

II. The Bending Strength Prediction Model

A. Data Preprocessing

As the basis, the interface type, the reinforcement fiber type and the perform type are all text data, they could not be processed with mathematic models. So, they were encoded into One-Hot codes [9].

As for the other factors, which are the reinforcement fiber type, the porosity, the interface thickness and the density, and the bending strength that is to predict are all digital data. They were scaled between 0 and 1 aiming to improve the performance of prediction models [10].

B. Prediction Models

We introduced four methods to build the bending strength prediction models.

SVM is a classical supervised learning model and usually used on small dataset. When SVM is used to fit a function, it is generally called support vector regression (SVR) [11]. XGBoost is an optimized distributed gradient boosting library implementing machine learning algorithms under the Gradient Boosting framework.

DNN, which is also as known as the multilayer perceptron (MLP), is a kind of function approximation model inspired by the neuroscience [12]. And is the most classical neural network model. A deep feedforward network consists of several layers, and each layer is composed of several units operating in parallel [13]. CNN is to replace the matrix multiplication of at least one layer in a DNN into the convolution operation. It is of sparse interaction, which reduces computation and improves the computation efficiency that could describe the complex interaction among multiple variables [14].

When comparing the four methods mentioned above, we used R^2 score to evaluate their performance. It is a scalar ranging from 0 to 1 [15], the name of which is the goodness of fit, also known as the coefficient of determination and denoted R^2. The more R^2 is closed to 1, the better the model fits the data. It is one of the most significant indexes in the regression fitting [16].

C. *Comparison of Models*

For all four models, 90 percent of dataset were set to the training dataset, the other 10 percent was as the test dataset. We also used 10-fold cross validation to prevent overfitting.

SVR is a distance-based model, so it usually is influenced by missing data obviously. The target of SVR is to find a hyperplane which could make the distances between points and itself small enough. The sketch is as Fig. 1 shows.

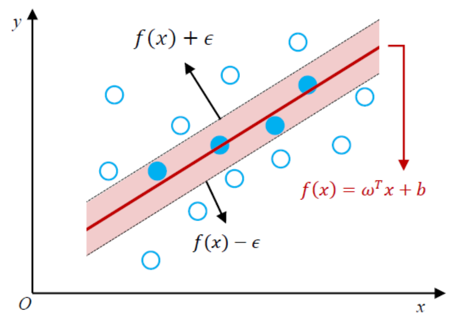

Fig. 1. Sketch of SVR model.

We replaced all missing values with zero to enable the model to work. The gamma and the penalty parameter C are two hyperparameters need to be tested by the grid search. When the C is 4.5 and the gamma is 0.1, we got the best R^2 score on the test dataset, which is 0.7795.

XGBoost is an open-source software library aiming to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library". XGBoost could process missing data well with its sparsity-aware split finding algorithm. This algorithm treats the non-presence as a missing value and learns the best direction to handle missing values. XGBoost is to build a certain number of trees, the sum of which is the target value to predict. While n presents the number of training samples, y presents the true value and the $\hat{y}$ presents the value predicted in last iteration, *loss* presents the loss function, *f(x)* presents a tree, Ω presents the regularization function, the objective function of XGBoost at iteration t is as equation (1) shows.

$$Obj^{(t)} = \sum_{i=1}^{n} loss\left(y_i, \hat{y}_i + f_t(x_i)\right) + \Omega(f_t) \qquad (1)$$

In this work, the XGBoost is to build trees as shows.

DNN and CNN could treat missing values correctly when they were replaced with zero [17]. The adaptive moment estimation (Adam) [18] is used as the optimization function, the rectified linear unit (ReLU) [19] is used to be the activation function, and the root-mean-square error (RMSE) is the loss function.

The aim of optimization function is to reduce the loss as much as possible, which is to make the neural network to fit data step by step. As for the activation function, a post-process step to drop noise after weighted calculation in neural cells, it is to make the neural network non-linear to possess stronger fitting capability.

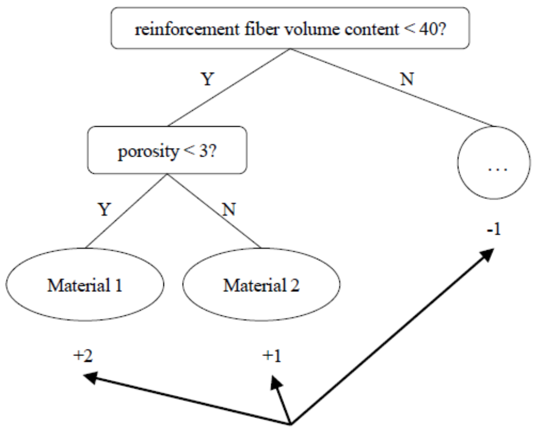

Fig. 2. Example of XGBoost tree in this work.

In fact, we also tried other optimization algorithms such as Stochastic Gradient Descent (SGD) and Adagrad, and other activation functions such as sigmoid and tanh function. The comparison result showed that the Adam and ReLU performed best.

For this order of magnitude of data, 500 epochs are enough to reach very stable and good metrics, so all results of neural networks below are gotten after 500 epochs.

We built a DNN model with two hidden layers, the architecture of which is as Fig. 3 shows.

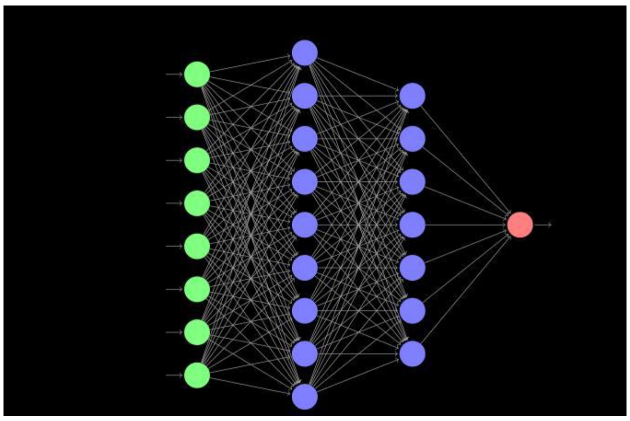

Fig. 3. DNN-based bending strength prediction model architecture.

TABLE I: R^2 SCORES ON THE TEST DATASET WITH DIFFERENT NUMBERS OF NEURONS OF HIDDEN LAYERS AFTER 500 EPOCHS

1st Layer / 2nd Layer	40	60	80	100	120	140	160	180
40	0.02	0.43	0.79	0.83	0.73	0.90	0.78	0.88
60	0.57	0.93	0.74	0.94	0.84	0.77	0.90	0.54
80	0.96	0.96	0.94	0.91	0.90	0.83	0.54	0.86
100	0.74	0.86	0.83	0.86	0.83	0.61	0.90	0.93
120	0.90	0.73	0.76	0.84	0.95	0.87	0.79	0.83
140	0.13	0.95	0.94	0.11	0.79	0.84	0.83	0.95
160	0.77	0.93	0.96	0.94	0.95	0.87	0.88	0.96
180	0.38	0.80	0.91	0.96	0.97	0.83	0.81	0.84

We tried different numbers of neurons of each layer from 40 to 180 by grid search to find the best pair that could get the highest R^2 score while overfitting phenomenon is not obvious, the result is as Table I shows.

As Table I shows, the best R^2 score of DNN is 0.97. There is not an evident trend with the numbers increasing.

Then we tried to build a CNN model, the architecture of which is as Fig. 4 shows.

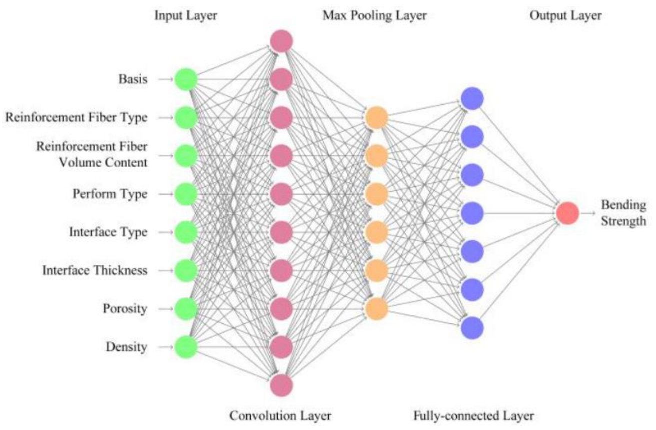

Fig. 4. CNN-based bending strength prediction model architecture.

The filter and the batch size of the convolution layer, and the number of neurons of the fully-connected layer are three hyperparameters need to be optimized by grid search. We searched the batch size from 1 to 9, the filter size from 20 to 180 and the number of neurons of the fully-connected layer from 20 to 180. The grid search result is as Fig. 5 shows.

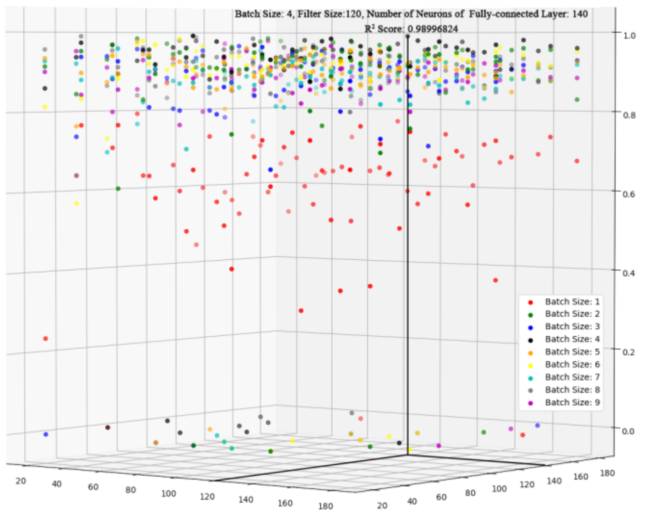

Fig. 5. Grid search result of hyperparameters of CNN after 500 epochs.

Finally, we found that when the batch size was 4, the filter size was 120 and the number of neurons of fully-connected layer was 140, the R^2 score on the test dataset reached the highest 0.9899 after 500 epochs.

The convolution layer executes a 1-D convolution function, which could extract information from multiple pieces of data by the convolution function more efficiently than fully-connected layer of DNN. The result also proved that CNN could get R^2 scores better and faster than DNN with less neurons.

The R^2 scores of four models mentioned above is as Table II shows.

TABLE II: R^2 SCORES COMPARISON OF FOUR MODELS

SVM	XGBoost	DNN	CNN
0.7795	0.9962	0.9731	0.9899

As Table II shows, SVM performed not well. XGBoost got the best R^2 score for its ability to treat missing data. DNN and CNN also got high scores, and CNN performed better.

III. PROCESS OF MISSING DATA

Then we turned to consider if the missing data could be processed so that the R^2 score could be improved further.

A. *Missing Data Statistic*

The factors influencing the bending strength of CMCs mainly consist of 8 parts, which are the basis, the reinforcement fiber type, the reinforcement fiber volume content, the perform type, the interface type, the interface thickness, the porosity and the density [20].

We collected 170 pieces of data, 22 of which are incomplete and the statistic is as Table III shows.

TABLE III: THE STATISTIC OF MISSING DATA

Reinforcement Fiber Volume Content	Porosity	Density
11	1	10

B. *Data Imputation*

```
Algorithm 1: K-Means Clustering Imputation
Input: data, data to imputation
Input: K
// Extract complete and incomplete data
$d_c \leftarrow$ Complete($d$)
$d_i \leftarrow$ Incomplete($d$)
Clusters $\leftarrow$ KMeans($d_c$, $K$)
Centers $\leftarrow$ Mean(Clusters)
// Find the cluster each $d_i$ denoted $d_i'$ belongs to
for $d_i'$ in $d_i$ do
    $m_d \leftarrow$ Distance($d_i'$, Clusters[0])
    $m_c \leftarrow 0$
    for $i = 1$ to Size(Clusters) - 1 do
        if Distance($d_i'$, Clusters[$i$]) < $m_d$ then
            $m_d$ = Distance($d_i'$, Clusters[$i$])
            $m_c = i$
        end if
    end for
    // Use the center of cluster $d_i'$ belongs to fill $d_i'$
    Fill($d_i'$, Centers[$m_c$])
end for
```

We tried three imputation methods, which are mean imputation, K-Means clustering imputation and DNN-based regression imputation.

Mean imputation is to replace any missing value with the mean of that variable for all other cases, which performs well when the values distribute around their mean values.

K-Means clustering imputation is a kind of unsupervised clustering algorithms. It needs to be assigned the number of clusters, denoted K. It is to divide all objects into K clusters, making every object be nearest to the center of the cluster it belongs to. K-Means imputation clusters all complete pieces of data firstly, then contrasts every incomplete piece of data with the centers of clusters by their distances. A piece of incomplete data would be filling with the center, which is generally the mean, of the cluster nearest to it. The algorithm is shown in Alg. 1.

For the K of K-Means clustering imputation needs to specified manually, we tried 18 numbers from 5 to 22. After finishing the imputation, the R^2 scores of the test dataset are as Table IV shows.

TABLE IV: R^2 SCORES OF AFTER K-MEANS CLUSTERING IMPUTATION

K \ Model	SVM	XGBoost	DNN	CNN
5	0.7207	0.9970	0.8536	0.9265
6	0.7206	0.9970	0.9199	0.9404
7	0.7207	0.9970	0.8925	0.9245
8	0.7207	0.9970	0.8909	0.8896
9	0.7206	0.9970	0.8002	0.8757
10	0.7207	0.9970	0.8383	0.8828
11	0.7206	0.9970	0.8595	0.9471
12	0.7207	0.9970	0.9408	0.8984
13	0.7207	0.9970	0.7914	0.9299
14	0.7206	0.9970	0.5263	0.8780
15	0.7207	0.9970	0.8914	0.8733
16	0.7207	0.9970	0.8304	0.8900
17	0.7207	0.9970	0.9414	0.9078
18	0.7207	0.9970	0.9775	0.8973
19	0.7207	0.9970	0.8034	0.9445
20	0.7207	0.9970	0.9698	0.9562
21	0.7207	0.9970	0.9197	0.9310
22	0.7203	0.9970	0.9298	0.9110

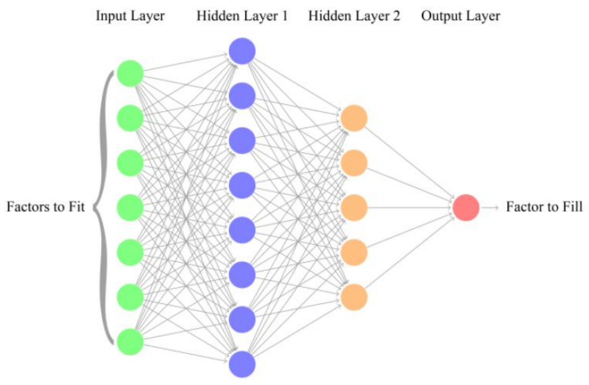

Fig. 6. DNN architecture to implement regression imputation.

As Table IV shows, after K-Means clustering imputation, SVM and XGBoost perform stably, while DNN and CNN are influenced by K obviously. R^2 score of SVM is lower than that of zero imputation. XGBoost performed a little better. DNN and CNN performed worse.

Regression imputation is a common imputation method in the traditional statistics that fits known data with a specified function to predict missing data. The fitting function is normally a linear function of a polynomial function, which performs not well when the data is extremely complex. Therefore, we used the regression imputation with neural networks as fitting functions to fill the incomplete data.

We build a DNN model with two hidden layers as the fitting function, the architecture of which is as Fig. 6 shows.

We also conducted grid search to get the best numbers of hidden layers of this DNN-based regression imputation model. Numbers of two hidden layers were tried from 20 to 160 with step 10. The imputation algorithm is as Alg. 2 shows.

After imputation methods above, R^2 scores of four models is as Table V shows. Hyperparameters were all optimized by grid search, and their scores in Table V are all the best.

As Table V shows, XGBoost performs very well and stably, which proves it is an ideal model to predict the bending strength. After DNN-based imputation, all models got a better R^2 score, in which CNN got the best 0.9998.

Algorithm 2: DNN-based Regression Imputation

```
Input: data, data to imputation
// Extract complete and incomplete data
$d_c \leftarrow$ Complete($d$)
$d_i \leftarrow$ Incomplete($d$)
$col_i \leftarrow$ IncompleteColumns($d$)
for $col$ in $col_i$ do
    $x \leftarrow d_c$.dropColumn($col$)
    $y \leftarrow d_c$.extractColumn($col$)
    $x_{train}, y_{train}, x_{test}, y_{test} \leftarrow$ SplitTrainTest($x$, $y$, 10% test)
    $score_{best} \leftarrow 0$
    $i_{best} \leftarrow 0$
    $j_{best} \leftarrow 0$
    for $i$ = 20 to 160 step 10 do
        for $j$ = 20 to 160 step 10 do
            $dnn \leftarrow$ DNN($i$, $j$)
            $dnn$.fit($x_{train}$, $y_{train}$, 10-fold cross validation)
            if $R^2$Score($dnn$.predict($x_{test}$), $y_{test}$) > $best_{score}$ then
                $i_{best} \leftarrow i$
                $j_{best} \leftarrow j$
                $score_{best} \leftarrow R^2$Score($dnn$.predict($x_{test}$), $y_{test}$)
            end if
        end for
    end for
    $dnn_{best} \leftarrow$ DNN($i_{best}$, $j_{best}$)
    $pred \leftarrow dnn_{best}$.predict($d_i$.dropColumn($col$))
    $d_i$.fillColumn($col$, $pred$)
end for
```

TABLE V: R^2 SCORES OF FOUR MODELS AFTER IMPUTATION

Model \ Method	Zero Imputation (No Process)	Mean Imputation	K-Means Clustering Imputation	DNN-based Regression Imputation
SVM	0.7795	0.7811	0.7207	0.8064
XGBoost	0.9962	0.9962	0.9970	0.9982
DNN	0.9731	0.7889	0.9698	0.9913
CNN	0.9899	0.9536	0.9562	0.9998

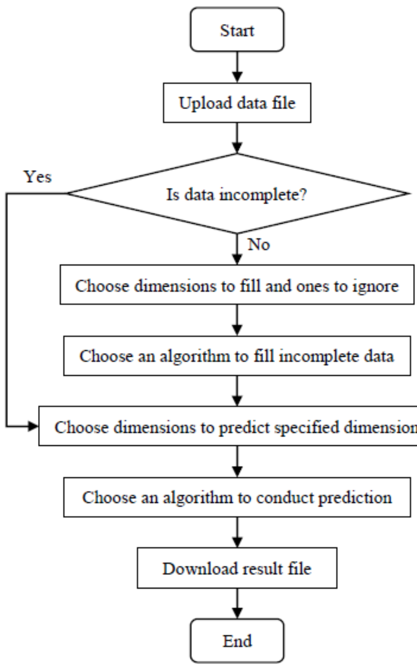
Start
Upload data file
Is data incomplete?
Yes
No
Choose dimensions to fill and ones to ignore
Choose an algorithm to fill incomplete data
Choose dimensions to predict specified dimension
Choose an algorithm to conduct prediction
Download result file
End

Fig. 7. Workflow of predicting CMCs bending strength with incomplete data.

IV. System Structure

The entire system workflow is as Fig. 7 shows.

To make the system easy to use for material researchers, we deployed it as a web application, the structure of which is as Fig. 8 shows.

The separation of front-end and backend is implemented by this web application. The front-end is based on Angular framework, and the backend is constructed with Django framework and TensorFlow, which are easy to scale out.

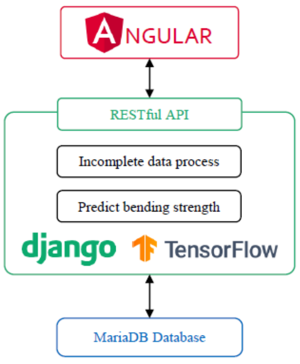

Fig. 8. Web application structure.

V. Conclusion

In this work, we collected experiments data of CMCs from documents, and tried to build models to predict the bending strength of CMCs with SVM, XGBoost, DNN and CNN. After grid search of hypermeters, we found XGBoost, DNN and CNN all performed well, and XGBoost got the best performance. Then we experimented three methods to process incomplete data to research how to improve the robustness of the prediction model. By comparison, we found the DNN-based regression imputation could improve the performance of the prediction models considerably. Finally, the CNN bending strength prediction model with DNN-based regression imputation got the best R^2 score 0.9998, and XGBoost with this imputation also got a high score 0.9982. This work provides a valuable design reference for researching new CMCs.

We did not research deeper neural networks to build prediction model on account of the small amount of data. Further, we did not research the similarity, specificity and 3-D physical structure of different CMCs, which were only distinguished by encoding. In our opinions, it is a direction worthy of further study and improvement to mine and analyze these hidden factors.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Gao Xiang and Li Guanghui conducted the research; Tan Rong and Yao Leijiang analyzed the data; Gao Xiang and Li Guanghui wrote the paper; all authors had approved the final version.

References

[1] S. Schmidt, S. Beyer, H. Knabe, H. Immich, R. Meistring, and A. J. A. A. Gessler, "Advanced ceramic matrix composite materials for current and future propulsion technology applications," *Acta Astronautica*, vol. 55, no. 3-9, pp. 409-420, Aug.-Nov. 2004.

[2] *Materials Genome Initiative for Global Competitiveness*, Executive Office of the President, National Science and Technology Council, 2011.

[3] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, article no. 27, 2011.

[4] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785-794, ACM.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. the IEEE*, Nov. 1998, vol. 86, no. 11, pp. 2278-2324.

[6] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967, vol. 1, no. 14, pp. 281-297, Oakland, CA, USA.

[7] O. Troyanskaya *et al.*, "Missing value estimation methods for DNA microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520-525, Jun. 2001.

[8] C. K. Enders, *Applied Missing Data Analysis*, Guilford press, 2010.

[9] J. E. Beck and B. P. Woolf, "High-level student modeling with machine learning," in *Proc. International Conference on Intelligent Tutoring Systems*, 2000, pp. 584-593, Springer.

[10] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. International Conference on Machine Learning*, 2015, pp. 448-456.

[11] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. Advances in neural information processing systems*, 1997, pp. 155-161.

[12] F. Rosenblatt, *Perceptions and the Theory of Brain Mechanisms*, Spartan Books, 1962.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems*, 2012, pp. 1097-1105.

[15] S. A. Glantz and B. K. Slinker, *Primer of Applied Regression and Analysis of Variance* (no. Sirsi) i9780070234079), 1990.

[16] N. R. Draper and H. Smith, *Applied Regression Analysis*. John Wiley & Sons, 1998.

[17] S. S. Haykin, S. S. Haykin, S. S. Haykin, K. Elektroingenieur, and S. S. Haykin, *Neural Networks and Learning Machines*, Pearson education Upper Saddle River, 2009.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," presented at the 3rd International Conference for Learning Representations, San Diego, May 7-9, 2015.

[19] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315-323.

[20] R. R. Naslain, "The design of the fibre-matrix interfacial zone in ceramic matrix composites," *Composites Part a-Applied Science and Manufacturing*, vol. 29, no. 9-10, pp. 1145-1155, 1998.

Gao Xiang received the Ph.D. degree in computer software and theory from Northwestern Polytechnical University, Xi'an, China, in 2004. Currently, he is an associate professor at the School of Computer Science, Northwestern Polytechnical University. His research interests include high creditability network, big data, network and information security and distributed computing.

Li Guanghui received the B.E. degree in software engineering from Northwestern Polytechnical University, Xi'an, China, in 2017. He is now a postgraduate student majoring in computer science and technology at the School of Computer Science, Northwestern Polytechnical University. His research interests include deep learning and data mining.

Tan Rong received the B.E. degree in software engineering from Shandong University, Wei'hai, China, in 2018. He is now a postgraduate student in software engineering. His main research direction is depth learning and image processing.

Yao Leijiang received the Ph.D. degree in flight vehicle design from Northwestern Polytechnical University, Xi'an, China, in 2000. Now he is an associate professor at the School of Laboratory of Science and Technology on UAV, Northwestern Polytechnical University. His research interests include structural integrity of new composites, fatigue and fracture of structural material and design and development of engineering material database.