Using Deep Neural Networks to Predict the Tensile Property of Ceramic Matrix Composites Based on Incomplete Small Dataset

GAO Xiang¹, LI Guanghui¹, TAN Rong² and YAO Leijiang³

¹ School of Computer Science, Northwestern Polytechnical University, Xi’an 710129, China
² School of Software, Northwestern Polytechnical University, Xi’an 710072, China
³ School of Laboratory of Science and Technology onUAV, Northwestern Polytechnical University, Xi’an 710072, China

sxlllsgh@mail.nwpu.edu.cn

Abstract. It has been a hot spot in the area of material science that how to design an experiment to produce a kind of ceramic matrix composites (CMCs) which possesses ideal performances. An approach is to build models with data from previous experiments recorded in published papers and predict the experiment parameters needed by the ideal CMCs. According to the database of CMCs funded by the National Material Genome Engineering, 8 factors were considered to affect the tensile property of CMCs, which were the basis, the reinforcement fiber type, the reinforcement fiber volume content, the perform type, the porosity, the interface type, the interface thickness and the density. Among the data we collected from papers, however, only few of pieces contained all the 8 factors, most were incomplete, some of them even lacked multiple factors. This paper’s work mainly researched how to take advantage of the incomplete data to build an effective model to predict the tensile property of CMCs. We proposed a model to predict the tensile property of CMCs based on a 1-D convolution neural network (CNN), the training data of which were all from papers. To decrease the influences of the incompleteness of data, we tried several methods to process the missing data, such as the mean imputation, the K-Means clustering imputation, the Hot-Deck Imputation and the regression imputation. The results showed that the regression imputation with a dual-hidden-layer feedforward network performed better and improved the performance of the CNN tensile property prediction model.

1. Introduction

In the traditional material science engineering, it used to take decades for experimenting to develop a new material. Further, the conditions like raw materials and environment of experiments depend on the experience and intuition of scientists deeply. As a consequence, the development of new materials progress as a slow pace. When it comes to the 21st century, all walks of life have been developing more and more rapidly, and the demand of various materials which could adapt to different special environments is becoming more and more urgent, so the slow traditional development progress of materials becomes a weak link of the upgrading of industry. Therefore, it becomes a problem to be solved urgently that finding an approach to predict the manufacturing process of materials of which performance meets requirements according to existing ones.
Ceramic matrix composites (CMCs) are a subgroup of composite materials, which consist of various fibres 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.[1]

In June, 2011, the United States government published Advanced Manufacturing Partnership (AMP), 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.

Due to the development process of materials is extremely sophisticated, time-consuming and costly, it is an effective way to extract the information from published documents for collecting research data as much as possible, which could be used to model to provide clues for designing experiments. Owing to the variety of the quality of documents, the lack of experiment factors exists in a certain number of documents, which results in the impossibility to train and predict using existing common models. Hence it is requisite to process the missing data.

In the traditional data process, dropping missing values, zero imputation, mean imputation and Hot Deck Imputation are usually used to process missing data[3]. Furthermore, K-Means clustering imputation and regression imputation based on the machine learning are often used[4-6]. This work compared and tested incomplete data processing methods above, researches the factors influencing the tensile property of CMCs, and builds a model to predict the tensile property of CMCs.

2. The tensile property prediction model

2.1. Data collection and preprocessing

The factors influencing the tensile property of CMCs mainly consist of 8 parts, which are the basis, the reinforcement fibre type, the reinforcement fibre volume content, the perform type, the porosity, the interface type, the interface thickness and the density[7], and we used the tensile strength to measure the tensile property of a material.

We collected 342 pieces of data totally, 67 pieces of which are complete, while there are different degrees of deficiencies in the other 275 ones, the statistics of which is as table 1.

Table 1. The statistic of missing data.

| Missing Item | Reinforcement Volume Content | Interface Type | Interface Thickness | Porosity | Density |
|--------------|------------------------------|----------------|---------------------|----------|---------|
| Count        | 76                           | 48             | 106                 | 250      | 227     |

As table 1 shows, there is a serious lack of data. In the 275 pieces of data, there are 21 pieces lacking of all the 5 factors mentioned above at the same time.

As the basis, the reinforcement fibre type, the perform type and the interface type are all text data, they could not be processed with mathematic models. So, they were encoded by One-Hot Encoding[8].

As for the other factors, which are the reinforcement fibre type, the porosity, the interface thickness and the density, and the tensile property that is to predict are all digital data. They were scaled between 0 and 1 aiming to improve the performance using neural networks[9].

2.2. Data mining model

This work tried to build the prediction model with a support vector machine (SVM)[10], a deep feedforward network and a 1-D convolution neural network and compared them. The deep feedforward network (DNN), which is also as known as the multilayer perceptron (MLP), is a kind of function approximation model inspired by the neuroscience[11]. and is the most classical neural
network model[12]. The SVM model usually performs well on the small dataset, so we used its result to assess the performance of neural networks.

A deep feedforward network consists of several layers, and each layer is composed of several units operating in parallel. Every unit represents a function from a vector to a scalar, which is called a neuron as figure 1 shows.

\[
\text{Input} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \xrightarrow{\omega_1, \omega_2, \omega_3, \ldots, \omega_n} f \xrightarrow{\text{Output}}
\]

**Figure 1.** A neuron.

The expression of the function is

\[
f(x; W, b) = g(W^T x + b)
\]  

where \( x \) means the vector inputted into the layer, which is

\[
x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}
\]

\( W \) means the weight matrix, which is

\[
W = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \vdots \\ \omega_n \end{pmatrix}
\]

\( b \) means the bias matrix. In another word, each input \( \omega_i x_i \) will be added with a bias \( b_i \), and the output will be calculated with the activation function \( g(x) \).

The neural network composed of such neurons is called forward, for there is no feedback connection between the output of the model and the model itself, which is called the feedforward neural network. The vector composed of input values in Figure 1 is generally called the input layer, and the vector composed of neurons in the last layer is usually called the output layer. Other layers between them are often habitually called the hidden layers. If the count of hidden layers is greater than or equal to two, the neural network is often called the deep feedforward network.

A convolution neural network (CNN) is to replace the matrix multiplication of at least one layer in a DNN into the convolution operation. CNN is of sparse interaction, which reduces computation and improves the computation efficiency that could describe the complex interaction among multiple variables[13].

We inputted 8 factors above into the neural networks and let the tensile property be the output. The adaptive moment estimation (Adam)[14] is used as the optimization function, and the rectified linear unit (ReLU)[15] is used to be the activation function. While comparing the performance of DNN and CNN, we filled the missing values with zero, for the neural networks is not sensitive with zero inputs, which means the missing data was not processed[16]. The 90 percent of the dataset was set to the training set, the other 10 percent was as the test set. We used the two neural networks respectively to train models and compared the performance of them on the test set with the root-mean-square error (RMSE) and the coefficient of determination as the evaluation indexes. RMSE is also the loss function in the training of the neural networks. The coefficient of determination, denoted \( R^2 \) and pronounced “R
squared", is a scalar ranging from 0 to 1. The more $R^2$ is close to 1, the better the model fits the data. It is one of the most significant indexes in the regression fitting[17, 18].

2.2.1. Deep feedforward network. We used a dual-hidden-layer feedforward network to build the prediction model, the structure of which is as figure 2 shows.

![Figure 2. A deep feedforward network with two hidden layers. The number of the neurons in the input layer is 8 that is equal to the dimensions of the input data. The number of the neurons in the output layer is 1, which corresponds the tensile property to predict. The numbers of the neurons in the hidden layers could not be determined by a reliable theory at present, so they should be tested in the experiments below.](image)

2.2.2. 1-D convolution neural network. Given the excellent performance of CNNs while extracting local features, this work also implemented a 1-D convolution neural network, the structure of which is as figure 3 shows.

![Figure 3. A 1-D convolution neural network.](image)

2.3. Process of missing data
Both neural networks above ignored missing data. For better prediction performance, we tried different methods to process missing data.
In the traditional methods processing missing data, there are roughly two types, which are dropping and filling. In filling methods, there are mean imputation, forward imputation, back imputation, Hot-Deck Imputation and more. With the machine learning developing, there are imputation methods based on K-Means clustering and neural networks appeared. In this work, we tried dropping rows with missing values, mean imputation, Hot-Deck Imputation, K-Means clustering imputation and regression imputation based on the neural network to process missing data, and compared the performance of them.

It is the easiest operation to drop rows with missing values, which could be free from dealing with missing data, but lose information in the part of rows not lost. This method is often used in the situation that there are quite limited influences after parts of data dropped. The 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. Hot-Deck Imputation is to fill the missing dimensions of a piece of data with values from the most similar but complete piece with it.

K-Means clustering 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 centre 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 centres of clusters by their distances. A piece of incomplete data would be filling with the centre, which is generally the mean, of the cluster nearest to it.

As for the 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, and then use the neural networks above to predict the tensile property.

3. Result and discussion

3.1. Comparison of the models

The SVM model, usually called the support vector regression (SVR)[19] model, possesses a large number of hyperparameters, two of which often need to be tested by the grid search, the gamma and the penalty parameter C. We also used 10-fold cross validation to prevent overfitting. When the C is 5.0 and the gamma is 0.1, we got the best R^2 score on the test dataset, which is 0.8935.

Aiming to find the best numbers of neurons of hidden layers in the feedforward neural network, we tested different numbers of neurons by the grid search method. The R^2 scores of the models on the test set after 500 epochs are as table 2 shows.

**Table 2.** The R^2 scores on the test dataset with different numbers of neurons of hidden layer after 500 epochs.

| 2nd Layer | 1st Layer | 20   | 40   | 60   | 80   | 100  | 120  | 140  | 160  |
|-----------|-----------|------|------|------|------|------|------|------|------|
| 20        | 0.9532    | 0.9764| 0.9665| Nan  | 0.9906| Nan  | 0.9715| 0.9892|      |
| 40        | 0.9876    | 0.9781| Nan  | 0.9925| Nan  | 0.9682| 0.9942|      |      |
| 60        | 0.9914    | 0.9821| Nan  | 0.9819| 0.9517| 0.9831| 0.9951| 0.9779|      |
| 80        | 0.9569    | Nan  | Nan  | 0.9914| 0.9950| 0.9810| 0.9920|      |      |
| 100       | 0.8950    | 0.9613| 0.9915| 0.9061| 0.9941| 0.9636| 0.9764|      |      |
| 120       | 0.8962    | 0.9832| 0.9363| 0.9720| 0.9939| 0.9870| 0.9804| 0.9809|      |
| 140       | 0.9650    | 0.9711| 0.9602| 0.9738| 0.9784| Nan  | 0.9918| 0.9626|      |
| 160       | 0.9840    | Nan  | 0.9542| 0.9719| 0.9898| 0.9867| 0.9861| 0.9870|      |
According to table 2, the $R^2$ scores after 500 epochs could reach a range from 0.8950 to 0.9951. Some $R^2$ scores are Nan, for their models does not converge. The variance of $R^2$ scores in table 2 is 0.000527.

For finding the best numbers of the neurons of the convolution layer (also as known as the convolution kernels or the filters) and the fully-connected layer, we set the size of the convolution kernels to 5, and tested different numbers of the filters and the neurons of the fully-connected layer by the grid method. The $R^2$ scores of the models on the test dataset after 500 epochs are as table 3 shows.

Table 3. The $R^2$ scores with different numbers of filters and neurons of the fully-connected layer after 500 epochs.

| Filters | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 |
|---------|----|----|----|----|-----|-----|-----|-----|-----|
| Fully Connected Layer | 20 | 0.9975 | Nan | 0.9915 | 0.9966 | Nan | 0.9950 | 0.9971 | Nan | 0.9900 |
|          | 40 | 0.9961 | 0.9800 | 0.9954 | Nan | 0.9900 | Nan | 0.9961 | 0.9783 | Nan |
|          | 60 | 0.9914 | 0.9897 | 0.9812 | 0.9898 | 0.9454 | 0.9952 | 0.9954 | 0.9837 | 0.9534 |
|          | 80 | Nan | 0.9764 | 0.9636 | 0.9787 | 0.9961 | 0.9953 | 0.9941 | 0.9989 | 0.9884 |
|          | 100 | 0.9907 | 0.9871 | 0.9920 | 0.9877 | Nan | Nan | Nan | 0.9970 | 0.9934 |
|          | 120 | Nan | 0.9952 | 0.9707 | 0.9950 | Nan | Nan | Nan | 0.9934 | 0.9883 | 0.9639 |
|          | 140 | 0.9931 | 0.9952 | 0.9948 | 0.9930 | 0.9969 | 0.9956 | 0.9912 | 0.9662 | 0.9810 |

As is shown in table 3, the $R^2$ scores could reach a range from 0.9454 to 0.9989 after 500 epochs, which means the CNN performs better than the tow-layer feedforward neural network. Furthermore, the variance of $R^2$ scores of CNN is 0.000141, which is nearly one fourth of that of DNN.

Therefore, we used the CNN as the model to train and predict the tensile property of CMCs. The size of the filters was set to 5, and the number of which was set to 160. The number of the neurons of the fully-connected layer was set to 80. Eventually, the $R^2$ score could reach 0.9989.

3.2. Comparison of different imputation methods
In section 3.1, we set all missing values to zero, which means we let the neural networks ignore the missing data. In order to achieve a better performance to predict the tensile property, we tried several different methods to process the missing data. The model to predict the tensile property is the CNN model above.

3.2.1. Dropping the rows with missing values. All the 275 pieces of data with missing values was dropped. In the 67 pieces of data left, 90% of them was as the training set, and 10% of them was as the test set. After 500 epochs, the $R^2$ score was only 0.9880.

3.2.2. Mean imputation. The $R^2$ score was 0.9971.

3.2.3. Hot-Deck Imputation. There is no strict definition for the similarity among the pieces of data, we used the Euclidean distance to measure it. After the Hot-Deck Imputation, the $R^2$ score was 0.9993, which showed a significant improvement.

3.2.4. K-Means clustering imputation. As the K, which presents the number of clusters, needs to specified manually, we tried 22 numbers from 5 to 26. After finishing the imputation, the $R^2$ scores of the test set are showed as table 4.
Table 4. The $R^2$ scores of the test set with different numbers of the K-Means clusters.

| Clustering Number | $R^2$ Score | Clustering Number | $R^2$ Score |
|-------------------|-------------|-------------------|-------------|
| 5                 | 0.9966      | 15                | 0.9975      |
| 6                 | 0.9973      | 16                | Nan         |
| 7                 | 0.9966      | 17                | 0.9967      |
| 8                 | 0.9972      | 18                | 0.9970      |
| 9                 | Nan         | 19                | 0.9986      |
| 10                | 0.9974      | 20                | 0.9988      |
| 11                | 0.9976      | 21                | 0.9970      |
| 12                | Nan         | 22                | Nan         |
| 13                | 0.9986      | 23                | 0.9990      |
| 14                | Nan         | 24                | 0.9970      |

The $R^2$ scores fell to the range from 0.9966 to 0.9990, which means the K-Means clustering imputation worked not better.

3.2.5. Regression imputation based on the neural networks. Due to the small amount of data and the difficulty training a neural network with multiple outputs, we built a dual-hidden-layer feedforward network with single output. There are 100 neurons of each hidden layer, and the input layer possesses 7 neurons corresponding the factors except the factor to fill. 90% of the data was as training set, and the other 10% was as the test set. We got an ideal filling effect after 300 epochs as table 5 shows.

Table 5. The statistic of the regression imputation with the dual-hidden-layer feedforward network.

| Item                  | Missing Item | Reinforcement Volume Content | Interface Type | Interface Thickness | Porosity | Density |
|-----------------------|--------------|-------------------------------|----------------|---------------------|----------|---------|
| Training Set Count    | 266          | 294                           | 236            | 92                  | 115      |         |
| Prediction Set Count  | 76           | 48                            | 106            | 250                 | 227      |         |

After the regression imputation with the dual-hidden-layer feedforward network above, we trained the prediction model with the CNN model above.

Finally, the $R^2$ score could be optimized to 0.9997. This result shows that the tensile property prediction model could get a significant improvement by the regression imputation based on a neural network.

As for the K-Means clustering imputation, we let the clustering number as 23, which led to the best $R^2$ score according to table 4, and compared the $R^2$ scores of different methods. The result is as table 6 shows.

Table 6. The comparison results of $R^2$ scores after different missing data process methods.

|                  | Zero Imputation (Ignore Missing Data) | Dropping Row | Mean Imputation | Hot Deck Imputation | K-Means clustering Imputation | Regression Imputation |
|------------------|---------------------------------------|--------------|----------------|--------------------|-------------------------------|----------------------|
|                  | 0.9989                                | 0.9880       | 0.9971         | 0.9993             | 0.9990                        | 0.9997               |

Table 6 shows that the Hot-Deck Imputation and the regression imputation could get a better prediction model than ignoring the missing data.

4. Conclusion
This work collected the CMC experiments data from documents, built a CMC tensile property prediction model with missing data by building a dual-hidden-layer feedforward network and a 1-D convolution neural network. Then we experimented multiple methods to process incomplete data to research how to improve the robustness of the prediction model. By comparison, we found the Hot-Deck Imputation and the regression imputation based on a dual-hidden-layer feedforward network could improve the performance of the prediction model considerably. 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 analyse these hidden factors.

Acknowledgments
Authors wishing to acknowledge assistance or encouragement from colleagues, special work by technical staff or financial support by the National Key R&D Program of China (2016YFB0700500) and the Science and Technology Project of Shaanxi Province (2018GY-048).

References
[1] Schmidt, S., et al., Advanced ceramic matrix composite materials for current and future propulsion technology applications. 2004. 55(3-9): p. 409-420.
[2] Science, N. and T. Council, Materials genome initiative for global competitiveness. 2011: Executive Office of the President, National Science and Technology Council.
[3] Kim, J.K., W.A. Fuller, and W.R.J.T.A.o.A.S. Bell, Variance estimation for nearest neighbor imputation for US census long form data. 2011. 5(2A): p. 824-842.
[4] MacQueen, J. Some methods for classification and analysis of multivariate observations. in Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. 1967. Oakland, CA, USA.
[5] Troyanskaya, O., et al., Missing value estimation methods for DNA microarrays. 2001. 17(6): p. 520-525.
[6] Enders, C.K., Applied missing data analysis. 2010: Guilford press.
[7] Naslain, R.R.J.C.P.A.A.S. and Manufacturing, The design of the fibre-matrix interfacial zone in ceramic matrix composites. 1998. 29(9-10): p. 1145-1155.
[8] Beck, J.E. and B.P. Woolf. High-level student modeling with machine learning. in International Conference on Intelligent Tutoring Systems. 2000. Springer.
[9] Ioffe, S. and C.J.a.p.a. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.
[10] Suykens, J.A. and J.J.N.p.l. Vandewalle, Least squares support vector machine classifiers. 1999. 9(3): p. 293-300.
[11] Rosenblatt, F., Perceptions and the theory of brain mechanisms. 1962: Spartan books.
[12] Goodfellow, I., Y. Bengio, and A. Courville, Deep learning. 2016: MIT press.
[13] Krizhevsky, A., I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. in Advances in neural information processing systems. 2012.
[14] Kingma, D.P. and J.J.a.p.a. Ba, Adam: A method for stochastic optimization. 2014.
[15] Glorot, X., A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. in Proceedings of the fourteenth international conference on artificial intelligence and statistics. 2011.
[16] Haykin, S.S., et al., Neural networks and learning machines. Vol. 3. 2009: Pearson education Upper Saddle River.
[17] Glantz, S.A. and B.K. Slinker, Primer of applied regression and analysis of variance. 1990.
[18] Draper, N.R. and H. Smith, Applied regression analysis. Vol. 326. 1998: John Wiley & Sons.
[19] Drucker, H., et al. Support vector regression machines. in Advances in neural information processing systems. 1997.