Construction Cost Index Forecasting: A Multi-feature Fusion Approach

Tianxiang Zhan\textsuperscript{a,b}, Yuanpeng He\textsuperscript{a,b}, Fuyuan Xiao\textsuperscript{b,c,*}

\textsuperscript{a}College of Computer and Information Science, Southwest University, Chongqing, 400715, China
\textsuperscript{b}School of Big Data and Software Engineering, Chongqing University, Chongqing, 401331, China
\textsuperscript{c}National Engineering Laboratory for Integrated Aero-Space-Ground-Ocean Big Data Application Technology, China

Abstract

The construction cost index is an important indicator of the construction industry. Predicting CCI has important practical significance. This paper combines information fusion with machine learning, and proposes a multi-feature fusion (MFF) module for time series forecasting. Compared with the convolution module, the MFF module is a module that extracts certain features. Experiments have proved that the combination of MFF module and multi-layer perceptron has a relatively good prediction effect. The MFF neural network model has high prediction accuracy and efficient prediction efficiency. At the same time, MFF continues to improve the potential of prediction accuracy, which is a study of continuous attention.

Keywords: Information Fusion, Construction Cost Index, Time Series Forecasting, Machine Learning

1. Introduction

The construction cost index (CCI) is an indicator that reflects the construction cost, and it is a research hotspot in the fields of construction and finance. The prediction of CCI is meaningful and necessary. Effectively improving the prediction level of CCI is one of the research goals. CCI data is a time series, and there are many forecasting methods for time series. Time series forecast-
ing methods include statistical methods, fuzzy forecasting methods [1, 2, 3, 4], complex methods [5, 6], evidence theory methods [7], machine learning methods [8, 9, 10], deep learning methods and so on [11, 12, 13].

In order to improve the prediction effect of CCI, this paper combines the ideas of information fusion and machine learning. Information fusion is a technology to fuse information from different sources to synthesize target data [14, 15, 16]. It is often used for intelligent decision-making, time series analysis and so on. This paper proposes a Multi-feature Fusion (MFF) neural network to predict CCI.

MFF uses the idea of pattern recognition to process time series in different feature. And MFF module generates a CCI feature sequence through the proposed sliding window and function sequence. The feature sequence saves the feature information of the CCI slices, and fuses the feature information into the required prediction data. Multi-layer perceptron here replaces the traditional information fusion method, which further improves the prediction effect. MFF neural network is composed of MFF module and Multi-layer perceptron. Experiments have proved that MFF has predictive accuracy in predicting CCI data. At the same time, MFF has the potential to further improve the accuracy of forecasting, and the proposal of MFF has made a contribution to time series forecasting.

The structure of this paper is as follows: the second section introduces some basic theories of MFF, the third section is the definition of MFF, the fourth section shows the effect of predicting CCI and the analysis of CCI prediction, and the fifth section summarizes the paper.

2. Preliminaries

This section includes the basic theory of MFF. It supposes that the time series $T$ is as follows.

$$T = \{(t_1, v_1), (t_2, v_2), (t_3, v_3), (t_4, v_4),..., (t_n, v_n)\}$$  \hspace{1cm} (1)

where $t_i$ is the point in time, and $v_i$ is the value at point $t_i$ in the time series.

The time series are treated as raw data as shown in Fig.1. The length of the
time series $T$ is $n$.

Raw Data

![Figure 1: The example of the raw data](image)

### 2.1. Sliding window and time slice set

Sliding window is a method in machine learning. By setting a fixed window size, data can be sliced by sliding. Assuming that the window size $Ws$ is a fixed integer ($Ws \leq n$, here $Ws = 3$ is taken as an example), the process of sliding the window is shown in Fig. 2.

![Figure 2: The process of sliding window](image)

**Definition 1.** The definition of the Sliding Window is as follows:

\[
\text{SlidingWindow}(T, Ws) = \{(t_i, \text{TimeSlice}_i)|1 \leq i \leq (n - Ws + 1)\}
\]  \quad (2)
where time slice means a continuous subsequence of the original time series and the definition of the Time Slice is as follows:

\[
\text{TimeSlice}_i = \{v_i, v_{i+1}, ..., v_{i+W_s-1}\}
\]  

(3)

\[(t_i, v_i) \subseteq T\]

(4)

**Definition 2.** The time slice set is the union of time slices generated by the time series through the sliding window as shown in Fig.3.

\[
\text{TimeSliceSet} = \text{SlidingWindow}(T, W_s)
\]

(5)

2.2. Multilayer perceptron

The multi-layer perceptron (MLP) is promoted from the perceptron learning algorithm (PLA) [17]. Multilayer perceptron can effectively enhance the robustness of machine learning and the problem of overfitting. The structure of MLP is shown in Fig.4 below. Each node in the MLP sums the input according to the weight and bias, and the weight and bias will change during the
optimization process.

Figure 4: The structure of MLP

2.3. Mean squared error loss function

Mean squared error (MSE) loss function is a loss function in machine learning [18]. The mean squared error is defined as follows.

\[
\text{MSE}_{\text{Loss}} = \text{mean}(L) \tag{6}
\]

\[
L = l(x, y) = \{ l_1, l_2, l_3, ..., l_n \} \tag{7}
\]

\[
l_i = (x_i - y_i)^2 \tag{8}
\]

where \( x \) is the input, \( y \) is the target, and the shapes of \( x \) and \( y \) are the same.

2.4. Adam method

Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments [19, 20]. Adam is simple to implement, has high computational efficiency, low memory requirements, and reduces the angle of the angle line, making it ideal
for data and parameter problems \cite{19,20}. The pseudo code of Adam is as follows \cite{19,20}. And the good default parameters of Adam are shown in Table 1 \cite{19,20}.

### Algorithm 1 Adam method

**Require:**
- $\alpha$: Stepsize
- $\beta_1, \beta_2 \in [0,1)$: Exponential decay rates for the moment estimates
- $f(\theta)$: Stochastic objective function with parameters $\theta$

**Require:**
- $\theta_0$: Initial parameter vector
- $m_0 \leftarrow 0$
- $v_0 \leftarrow 0$
- $t \leftarrow 0$

**while** $\theta_t$ not converged **do**

- $t \leftarrow t + 1$
- $g_t \leftarrow \nabla f_t(\theta_t)$
- $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
- $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
- $\hat{m}_t \leftarrow m_t / (1 - \beta_1)$
- $\hat{v}_t \leftarrow v_t / (1 - \beta_2)$
- $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$

**return** $\theta_t$

| Parameter | Meaning | Good default settings |
|-----------|---------|-----------------------|
| $\alpha$ | Step Size | 0.001 |
| $(\beta_1, \beta_2)$ | Exponential decay rates for the moment estimates | $(0.9, 0.999)$ |
| $\epsilon$ | Term added to the denominator | $10^{-8}$ |
| $f(\theta)$ | Stochastic objective function with parameters $\theta$ | \( \backslash \) |

**Table 1:** The meaning and good default settings of Adam parameter

2.5. Cyclical learning rates

Cyclical learning rates (CLR) is a method of dynamically adjusting the learning rate in machine learning. CLR eliminates the need for experiments to find the best value and timetable for the global learning rate. CLR does not reduce the learning rate in a monotonous manner, but rather makes the learning rate fluctuate between reasonable boundary values on a regular basis. The parameters and schematic diagram of CLR are shown in Table 2 and Fig. 5.
| Parameter           | Meaning                                                                 |
|---------------------|-------------------------------------------------------------------------|
| Base learning rate  | Lower learning rate boundaries in the cycle for each parameter group    |
| Max learning rate   | Upper learning rate boundaries in the cycle for each parameter group    |
| Step size up        | Number of training iterations in the increasing half of a cycle         |
| Step size down      | Number of training iterations in the decreasing half of a cycle         |

Table 2: The meaning of CLR

![Schematic diagram of CLR](image)

Figure 5: Schematic diagram of CLR

3. Multi-feature Fusion

3.1. Step 1: Input time series

The input of MFF is the time series $T$. The time series $T$ is as follows:

$$T = \{(t_1, v_1), (t_2, v_2), (t_3, v_3), (t_4, v_4), \ldots, (t_n, v_n)\}$$  \hspace{1cm} (9)

where $t_i$ is used as an index and does not exist in the form of $(t_i, v_i)$ tuples.

3.2. Step 2: Slice time series

When generating a time slice set, MFF needs to determine the size of a sliding window $Ws$. The calculation process of Time slice set $S_T$ is as follows:

$$S_T = \text{SlidingWindow}(T, Ws)$$  \hspace{1cm} (10)

When generating a time slice set, the setting of $Ws$ needs to be considered. The number of time slices is $(n - Ws)$. Excessive $Ws$ results in fewer slices and
fewer learning samples. If \( W_s \) is too small, each sample can only reflect short time series characteristics. \( W_s \approx \frac{1}{2}n \) is default parameters. The shape of \( S_T \) is \(((n - W_s + 1), W_s)\).

3.3. Step 3: Input function sequence

In step 3, MFF needs to complete the preprocessing of the time slice set \( S_T \) and convert the time slice into a feature sequence. The function sequence is a converter that converts the time slice into a feature sequence as shown in Fig. 6.

![Figure 6: Example of feature conversion (Window size=8, there are four functions in the function sequence. \( v_i \) is the value corresponding to the time node \( i \) in the time series.)](image)

**Definition 3.** Function sequence is a set of functions, defined as follows:

\[
Fs(x) = \{F_1(x), F_2(x), ..., F_m(x)\} \tag{11}
\]

\[
F_i(x) = f_{e_i} \tag{12}
\]

where \( m \) is the number of functions in the function sequence \( Fs \), \( x \) is a time slice and \( F_i(x) \) transfers \( x \) which is in the shape \((1 \times W_s)\) to feature \( f_{e_i} \) which is in the shape of \((1 \times 1)\).

After the function sequence is input, the time slice set \( S_T \) is converted to the feature sequence set \( S_F \) as follows:

\[
S_F = Fs(S_T) = \{Fs(S_T_1), Fs(S_T_2), ..., Fs(S_T_{n-W_s+1})\} \tag{13}
\]

\[
Fs(S_T_i) = \{f_{e(i,1)}, f_{e(i,2)}, ..., f_{e(i,m)}\} \tag{14}
\]
\( f_{e(i,j)} \) represents the feature value generated by the function \( F_j(x) \) in the time slice \( S_T \). For different training targets and training data, the feature function \( F(x) \) selected in MFF for prediction is different. The shape of feature sequence set \( S_F \) is \(((n - Ws + 1), m)\). The MFF module is composed of sliding window and function sequence processing.

### 3.4. Step 4: Multilayer perceptron: forward propagation

In MFF, MLP has four layers: input layer, hidden layer 1, hidden layer 2 and output layer. The nodes in the three layers are \( m, n_1, n_2 \) and 1 as shown in Fig.7. Each feature sequence will be input into MLP, and then a result will be input. Whenever the result corresponding to the feature sequence is generated, it will do back propagate and optimize the parameters. A forward propagation and back propagation are called an epoch. Each epoch will update the result of the result as follows:

\[
\text{result} \leftarrow \text{MLP}(m, n_1, n_2)
\]  

![Figure 7: The structure of MLP in the MFF](image_url)
3.5. Step 5: Multilayer perceptron: back propagation and parameter optimization

In MFF, each epoch needs back propagation and parameter optimization. The loss function of MFF is MSE and the target is next time node’s value of the current time slice. After calculating the loss in each epoch, the parameters of MFF are back-propagated and optimized by Adam algorithm and CLR. When initializing MFF, it is necessary to input the upper and lower limits of the learning rate, which are dynamically adjusted by the CLR algorithm during training. MFF does not use the traditional gradient descent method of MLP, but uses the Adam algorithm for gradient descent, which accelerates machine learning and strengthens the effect of machine learning.

The loss and model parameters calculated in each epoch will be saved in a set. In MFF, the number of epochs $N$ is a variable set in advance. After the back propagation and parameter optimization of each epoch updated by the Adam algorithm and CLR, MFF returns to Step 4 for the next epoch training. When the last epoch is completed, a set of training parameters with the smallest loss will be selected for prediction. The process of MFF is shown in Fig.8 and the pseudo code of MFF is as follows.

3.6. Step 6: Predict

In MFF, the model parameter with the smallest loss is applied to the MLP and then the time series that needs to be predicted are input into the MFF to complete the prediction.

4. Experiment

4.1. Data set description

Engineering News Record (ENR) is a monthly publication that publishes the CCI \cite{21,22}. CCI has been studied by many civil engineers and cost analysts because it contains vital building industry price information. The CCI data set includes a total of 295 data values of construction costs from January 1990 to July 2014.
Figure 8: The process of MFF

Figure 9: The flow chart of MFF
Algorithm 2 MFF($T, Ws, F_s, N, \text{Shape}$)

**Require:** Time series $T$
**Require:** Sliding window size $Ws$
**Require:** Function Sequence $F_s$
**Require:** Number of epoch $N$
**Require:** Shape of MLP

1. Slice time series by sliding window algorithm
2. Generate feature sequence $S_F$ through time slice set $S_T$ and function sequence $F_s$
3. Train: Set $F_s$ as training set
4. for Epoch = 1 to $N$ do
5. MLP: forward propagation
6. MLP: back propagation and parameter optimization by Adam and CLR algorithm
7. Save model parameter and loss in model set $S_M$
8. Predict: Apply the model with minimum loss in the MLP
9. Input last feature and output the result $\hat{y}_{n+1}$
10. return $\hat{y}_{n+1}$

4.2. Experiment preprocessing

For CCI, MFF needs to determine the size of a window, $Ws = 180$ in the experiment as an example. At the same time, the last data is used as the target of the penultimate time point, without sliding window. A total of 116 time slices were generated, and there were 116 corresponding feature sequences. In this experiment, 116 pieces of data are divided into experimental set and test set according to the ratio of 8:2.

The choice of function is variable. In this experiment, the MFF function sequence is composed of 6 functions. The function names and definitions are shown in Tab.3. Also, the number of nodes $(m, n)$ of MLP is set to $(8, 5)$ and the max epoch is 10000 in this experiment. In the CLR algorithm, the base learning rate is $10^{-12}$ and the max learning rate is $10^{-4}$. In the MFF training process, the gradient descent uses the Adam algorithm, which can improve the training efficiency of the model.

4.3. Experimental results

To evaluate the prediction of each method, there are five measures of error: mean absolute difference (MAD) [25], mean absolute percentage error (MAPE) [26], symmetric mean absolute percentage error (SMAPE) [27], root mean
Function Definition

| Index            | The order of time nodes in the current slice |
|------------------|---------------------------------------------|
| Mean             | Average of the time series                  |
| Standard deviation | Standard deviation of the time series        |
| Distance         | Time series maximum minus minimum           |
| ApEn             | Approximate entropy of time series [23]     |
| Degree           | The sum of the degrees of the visibility graph [24] |

Table 3: Function sequence in the experiment \((F(x) \text{ selected in this experiment})\)

Square error (RMSE) [28], and normalized root mean squared error (NRMSE) [29]:

\[
\text{MAD} = \frac{1}{N} \sum_{t=1}^{N} |\hat{y}(t) - y(t)|
\]  \hspace{1cm} (16)

\[
\text{MAPE} = \frac{1}{N} \sum_{t=1}^{N} \frac{|\hat{y}(t) - y(t)|}{y(t)}
\]  \hspace{1cm} (17)

\[
\text{SMAPE} = \frac{2}{N} \sum_{t=1}^{N} \frac{|\hat{y}(t) - y(t)|}{\hat{y}(t) + y(t)}
\]  \hspace{1cm} (18)

\[
\text{RMSE} = \sqrt{\frac{1}{N} \sum_{t=1}^{N} (\hat{y}(t) - y(t))^2}
\]  \hspace{1cm} (19)

\[
\text{NRMSE} = \sqrt{\frac{\sum_{t=1}^{N} (\hat{y}(t) - y(t))^2}{y_{\text{max}} - y_{\text{min}}}}
\]  \hspace{1cm} (20)

where \(\hat{y}(t)\) is the predicted value, \(y(t)\) is the true value and \(N\) is the total number of \(\hat{y}(t)\).

Fig. 10 shows the prediction of MFF(8,5). The predicted value of MFF is close to the actual value, and the prediction effect is good.

### 4.4 Comparative Experiment

In order to verify the prediction effect of MFF, the prediction results of MFF will be compared with three different types of prediction methods statistical prediction methods, machine learning regression methods and hybrid prediction methods. At the same time, in order to distinguish it from the existing deep learning methods, MFF and MLP, Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM) prediction methods have been ablated experiments.
4.4.1. Comparison between MFF and statistical prediction methods

In this section, MFF will be compared with statistical prediction methods. Among the statistical comparison methods, Simple Moving Average (SMA) (K=1) [30], Autoregressive Integrated Moving Average model (ARIMA) [31], Seasonal Autoregressive Integrated Moving Average model (Seasonal ARIMA) [32] and Exponential Smoothing (ETS) [33] are commonly used methods for prediction. Random walk is also a commonly used prediction method in statistics. Mao and Xiao proposed a random walk prediction method based on complex networks, which has good prediction performance and will also be used as a comparison method [34].

Tab.4 and Fig.11 are the comparison of the experimental effects of MFF and statistical prediction methods. According to the experimental results, MFF performs better than the statistical methods mentioned above. In Fig.11, the statistical method predicts the result is relatively stable, the trend is similar to the true value, but compared with MFF, MFF is more stable, and MFF is closer to the actual value.

Figure 10: Prediction of MFF(8,5)
| Method                        | MAD    | MAPE   | SMAPE  | RMSE   | NRMSE  |
|-------------------------------|--------|--------|--------|--------|--------|
| ETS [33]                      | 55.6591| 0.5838 | 0.5816 | 64.4560| 300.9969|
| Seasonal ARIMA [32]           | 45.3349| 0.4769 | 0.4753 | 54.8709| 240.0670|
| SMA(K=1) [30]                 | 43.7391| 0.4582 | 0.4566 | 55.8180| 256.9233|
| ARIMA [31]                    | 38.6931| 0.4055 | 0.4044 | 47.7177| 214.7822|
| Mao and Xiao’s Method [34]    | 37.7301| 0.3940 | 0.3928 | 49.7481| 226.0986|
| MFF(8,5)                      | 22.2877| 0.2318 | 0.2316 | 29.2458| 131.5833|

Table 4: Forecasting error of MFF and statistical prediction methods

![Comparison between MFF and statistical prediction methods](image)

Figure 11: Comparison between MFF(8,5) and statistical prediction methods
4.4.2. Comparison between MFF and machine learning regression methods

In this section, MFF will be compared with machine learning regression methods. Among the machine learning comparison methods, Decision Tree Regression (DTR) [35], Ordinary least squares Linear Regression (Linear) [36], Lasso model fit with Least Angle Regression (Lasso) [37], Support Vector Machines Regression (SVM) [38], Bayesian Ridge Regression (Bayesian) [39] and Logistic Regression (Logistic) [40] [41] are commonly used methods for prediction.

Tab.5 and Fig.12 are the comparison of the experimental effects of MFF and machine learning regression methods. According to the experimental results, MFF performs better than the machine learning methods mentioned above. The trend of machine learning regression methods is stable and more accurate than statistical methods. There is a gap between the predicted value and MFF, and MFF is closer to the true value. At the same time, the jitter of MFF is small, and the trend is close to the real trend of CCI.

![Comparison between MFF and machine learning regression methods](image)

Figure 12: Comparison between MFF(8, 5) and other methods

4.4.3. Comparison between MFF and hybrid prediction methods

In this section, MFF will be compared with hybrid prediction methods. Hybrid model is a method for time series forecasting to improve forecast accuracy.
| Method     | MAD   | MAPE  | SMAPE | RMSE  | NRMSE |
|------------|-------|-------|-------|-------|-------|
| DTR        | 58.39 | 0.61  | 0.61  | 71.71 | 368.87|
| Linear     | 47.87 | 0.50  | 0.49  | 60.67 | 279.28|
| SVM        | 45.65 | 0.48  | 0.48  | 52.74 | 245.01|
| Lasso      | 30.97 | 0.32  | 0.32  | 40.16 | 189.95|
| Bayesian   | 30.82 | 0.32  | 0.32  | 39.88 | 188.15|
| Logistic   | 30.39 | 0.32  | 0.32  | 39.62 | 187.27|
| MFF(8,5)   | 22.29 | 0.23  | 0.23  | 29.25 | 131.58|

Table 5: Forecasting error of MFF and machine learning regression methods

The use of hybrid models can combine the linear characteristics of statistical methods and the characteristics of machine learning nonlinear prediction Artificial Neural Network (ANN) to further increase the accuracy of prediction. Common hybrid models are ARIMA-ANN \[42\] and ETS-ANN \[43\].

This experiment takes a parallel approach in the experiment of the hybrid model. In order to ensure the rigor of the experiment, the ANN hybrid model and MFF in the hybrid model use the same training parameters include Adam algorithm and CLR.

Tab.6 and Fig.13 are the comparison of the experimental effects of MFF and hybrid prediction methods. According to the experimental results, MFF performs better than the hybrid prediction methods mentioned above. The hybrid model further improves the accuracy of prediction. ARIMA and ETS as statistical models have good prediction performance. ANN as a non-linear model for processing ARIMA and ETS further improves the accuracy and reduces errors. But compared with MFF, the hybrid model has a greater degree of jitter.

| Method         | MAD    | MAPE | SMAPE | RMSE  | NRMSE  |
|----------------|--------|------|-------|-------|--------|
| ARIMA-ANN \[42\] | 45.29 | 0.47 | 0.47 | 53.62 | 240.71 |
| ETS-ANN \[43\]  | 24.47 | 0.25 | 0.25 | 32.03 | 147.48 |
| MFF(8,5)        | 22.29 | 0.23 | 0.23 | 29.25 | 131.58 |

Table 6: Forecasting error of MFF and hybrid prediction methods

4.4.4. Ablation experiment

In order to explore the relationship between MFF's ability to improve the prediction effect and MLP, a combination of MFF and LSTM and CNN are used for ablation experiments. LSTM and CNN are common deep learning neural network models, which have applications in time series prediction \[44, 45\]. By
| Structure                        | Training Iteration | Learning Rate | Loss Function | Optimizer Function |
|---------------------------------|--------------------|---------------|---------------|--------------------|
| (Hidden Layer/                 |                    |               |               |                    |
| Fully connected Layer/         |                    |               |               |                    |
| Convolution Layer/             |                    |               |               |                    |
| Pooling Layer/                 |                    |               |               |                    |
| Dense Layer)                   |                    |               |               |                    |
| MLP Hidden Layer=100           | 10000              | 0.01          | MSE            | Adam               |
| Activation=Relu                |                    |               |               |                    |
| Output Layer=1                 |                    |               |               |                    |
| LSTM Hidden Layer=50           | 10000              | 0.01          | MSE            | Adam               |
| Activation=Relu                |                    |               |               |                    |
| Input Timestep=3               |                    |               |               |                    |
| Output Timestep=1              |                    |               |               |                    |
| Dense Layer=1                  |                    |               |               |                    |
| CNN Convolution Layer:         |                    |               |               |                    |
| Filters=64                     |                    |               |               |                    |
| Kernel Size=2                  |                    |               |               |                    |
| Activation=Relu                |                    |               |               |                    |
| Pooling Layer:                 |                    |               |               |                    |
| Pool Size=2                    |                    |               |               |                    |
| Dense Layer=100                |                    |               |               |                    |
| Dense Layer=1                  |                    |               |               |                    |
| MFF+LSTM Hidden Layer=50       | 10000              | 0.01          | MSE            | Adam               |
| Activation=Relu                |                    |               |               |                    |
| Input Timestep=3               |                    |               |               |                    |
| Output Timestep=1              |                    |               |               |                    |
| Dense Layer=1                  |                    |               |               |                    |
| MFF Layer                      |                    |               |               |                    |
| CNN MFF Layer                  |                    |               |               |                    |
| Filters=64                     |                    |               |               |                    |
| Kernel Size=2                  |                    |               |               |                    |
| Activation=Relu                |                    |               |               |                    |
| Pooling Layer:                 |                    |               |               |                    |
| Pool Size=2                    |                    |               |               |                    |
| Dense Layer=100                |                    |               |               |                    |
| Dense Layer=1                  |                    |               |               |                    |
| MFF+CNN                         |                    |               |               |                    |
| MFF Layer                      |                    |               |               |                    |
| Convolution Layer:             |                    |               |               |                    |
| Filters=64                     |                    |               |               |                    |
| Kernel Size=2                  |                    |               |               |                    |
| Activation=Relu                |                    |               |               |                    |
| Pooling Layer:                 |                    |               |               |                    |
| Pool Size=2                    |                    |               |               |                    |
| Dense Layer=100                |                    |               |               |                    |
| Dense Layer=1                  |                    |               |               |                    |

Table 7: Comparison of method parameters for ablation experiments
replacing the MLP inside MFF with CNN, LSTM, the model parameters of the
comparison method in the ablation experiment are shown in the Tab.7.

The MFF module is essentially a feature extraction, and the effect is similar
to the convolutional layer, but MFF is a directional feature extraction. CNN can
effectively extract features through the convolutional layer and the pooling layer,
but this feature is not a definite feature, it will change with the change of data
and training parameters, and it is not robust in prediction. LSTM solves the
long-term dependence on information, but when the data passes through the
MFF, the index feature eliminates the context of the time series, and the learning
effect will not change due to the learning order. The combination of CNN and
MFF will lose information, while LSTM will lose its long-term dependence on
information. Single MLP is a non-linear fitting of the original time series data,
which has no advantages compared with CNN and LSTM [46]. Under this
premise, combining the MFF module with MLP would be a relatively good
choice.

Tab.8 and Fig.14 are the comparison of the ablation experiment. According
to the experimental results, MFF performs better than single deep learning
prediction method MLP, CNN and LSTM and MFF combined with CNN and
4.5. Additional experiment

In order to show the prediction effect of MFF\((M, N)\), the experimental effect of different MLP parameters \((M, N)\) will be tested here. Both M and N were tested from 1 to 20, and a total of 400 models were tested. At the same time, the top 10 models and errors of the prediction effect are shown in Tab.9.

Experimental results show that the prediction performance of MFF can be further improved by trying different MLP models in the MFF. MFF has the potential for further improvement.

|        | MAD     | MAPE    | SMAPE   | RMSE   | NRMSE   |
|--------|---------|---------|---------|--------|---------|
| MFF+CNN| 42.9095 | 0.4474  | 0.4463  | 55.9896| 262.7687|
| MFF+LSTM| 41.3319 | 0.4332  | 0.4325  | 49.7223| 219.0653|
| CNN    | 38.4514 | 0.4005  | 0.4000  | 46.3639| 199.8156|
| MLP    | 38.2716 | 0.3991  | 0.3982  | 51.7000| 227.9186|
| LSTM   | 27.0059 | 0.2805  | 0.2801  | 34.8895| 165.9856|
| MFF(8,5)| 22.2877 | 0.2318  | 0.2316  | 29.2458| 131.5833|

Table 8: Forecasting error in the ablation experiment
### 4.6. Experiment conclusion

In MFF, the function sequence contains the generation methods for the characteristics of multiple directions of the time sequence. Through the method of information fusion, MFF fuses different features into prediction targets. Feature fusion uses machine learning which more flexibly fuse target data based on existing data. The Adam and CLR algorithms are used for back propagation and parameter optimization of MLP, which improves the training effect while increasing the robustness and efficiency of MLP.

The prediction effect of MFF is better than common statistics, machine learning and hybrid methods. Compared with deep learning CNN and LSTM neural network predictions, the directional feature learning of the MFF module determines that MLP is a relatively suitable model. After ablation experiments, it is proved that the combination of MFF module and MLP has a better prediction effect. In additional experiments, by adjusting the parameters of MFF, the prediction accuracy of MFF is further improved, and MFF has the potential to continue to improve the prediction effect. Among the five statistical error indicators in the experiment, MFF has the smallest error, indicating that MFF has high prediction accuracy.

In terms of time complexity, the time complexity of the MFF module depends on the function sequence selected in MFF. In this experiment, the time complexity of the MFF module is $O(n^2)$ and the time complexity of the convolutional layer in CNN is the same. In terms of actual test time, MFF has good predictive performance.

In a conclusion, MFF has a more accurate prediction effect and efficient
prediction performance. At the same time, MFF has the potential to further improve forecast accuracy.

5. Conclusion

The paper proposed the MFF method to predict CCI. By combining information fusion and machine learning, the prediction effect of MFF has been improved compared with commonly used prediction methods. The proposal of MFF has contributed to CCI and time series forecasting. In the future, MFF will continue to improve and explore time series forecasting methods based on information fusion and machine learning.

Acknowledgment

This research is supported by the National Natural Science Foundation of China (No. 62003280) and Chongqing Talents: Exceptional Young Talents Project (CQYC202105031).

Conflict of Interests

The authors declare that there are no conflict of interests.

References

[1] A. Parida, R. Bisoi, P. Dash, S. Mishra, Times series forecasting using chebyshev functions based locally recurrent neuro-fuzzy information system, International Journal of Computational Intelligence Systems 10 (1) (2017) 375–393.

[2] F. Xiao, On the maximum entropy negation of a complex-valued distribution, IEEE Transactions on Fuzzy Systems 29 (11) (2021) 3259–3269.

[3] F. Xiao, CaFtR: A fuzzy complex event processing method, International Journal of Fuzzy Systems (2021) DOI: 10.1007/s40815–021–01118–6.

[4] D. Xie, F. Xiao, W. Pedrycz, Information quality for intuitionistic fuzzy values with its application in decision making, Engineering Applications of Artificial Intelligence (2021) Accepted.
[5] L. Chen, Y. Deng, K. H. Cheong, Probability transformation of mass function: A weighted network method based on the ordered visibility graph, Engineering Applications of Artificial Intelligence 105 (2021) 104438. doi: https://doi.org/10.1016/j.engappai.2021.104438

[6] F. Xiao, CEQD: A complex mass function to predict interference effects, IEEE Transactions on Cybernetics (2021) DOI: 10.1109/TCYB.2020.3040770.

[7] T. Zhan, F. Xiao, A fast evidential approach for stock forecasting, International Journal of Intelligent Systems (2021). doi:https://doi.org/10.1002/int.22598

[8] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of AAAI, 2021.

[9] Y. Huang, Y. Gao, Y. Gan, M. Ye, A new financial data forecasting model using genetic algorithm and long short-term memory network, Neurocomputing 425 (2021) 207–218.

[10] Y. Huang, X. Mao, Y. Deng, Natural visibility encoding for time series and its application in stock trend prediction, Knowledge-Based Systems 232 (2021) 107478. doi:https://doi.org/10.1016/j.knosys.2021.107478

[11] A. F. W. Ho, B. Z. Y. S. To, J. M. Koh, K. H. Cheong, Forecasting hospital emergency department patient volume using internet search data, IEEE Access 7 (2019) 93387–93395.

[12] C. Cheng, F. Xiao, A distance for belief functions of orderable set, Pattern Recognition Letters 145 (2021) 165–170.

[13] Y. Huang, Y. Deng, A new crude oil price forecasting model based on variational mode decomposition, Knowledge-Based Systems 213 (2021) 106669. doi:https://doi.org/10.1016/j.knosys.2020.106669

[14] F. Xiao, Evidence combination based on prospect theory for multi-sensor data fusion, ISA Transactions 106 (2020) 253–261.
[15] F. Xiao, GIQ: A generalized intelligent quality-based approach for fusing multi-source information, IEEE Transactions on Fuzzy Systems 29 (7) (2021) 2018–2031.

[16] J. Deng, Y. Deng, K. H. Cheong, Combining conflicting evidence based on pearson correlation coefficient and weighted graph, International Journal of Intelligent Systems (2021) DOI: 10.1002/int.22593.

[17] J. Tang, C. Deng, G.-B. Huang, Extreme learning machine for multilayer perceptron, IEEE transactions on neural networks and learning systems 27 (4) (2015) 809–821.

[18] G. Pavlov, A. Maydeu-Olivares, D. Shi, Using the standardized root mean squared residual (srmr) to assess exact fit in structural equation models, Educational and Psychological Measurement 81 (1) (2021) 110–130.

[19] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2015).

[20] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, arXiv preprint arXiv:1711.05101 (2017).

[21] S. M. Shahandashti, B. Ashuri, Forecasting engineering news-record construction cost index using multivariate time series models, Journal of Construction Engineering and Management 139 (9) (2013) 1237–1243.

[22] S. Hwang, Time series models for forecasting construction costs using time series indexes, Journal of Construction Engineering and Management 137 (9) (2011) 656–662.

[23] L. Montesinos, R. Castaldo, L. Pecchia, On the use of approximate entropy and sample entropy with centre of pressure time-series, Journal of neuro-engineering and rehabilitation 15 (1) (2018) 1–15.

[24] L. Lacasa, B. Luque, F. Ballesteros, J. Luque, J. C. Nuno, From time series to complex networks: The visibility graph, Proceedings of the National Academy of Sciences 105 (13) (2008) 4972–4975.

[25] S. Yitzhaki, et al., Gini’s mean difference: A superior measure of variability for non-normal distributions, Metron 61 (2) (2003) 285–316.
[26] A. De Myttenaere, B. Golden, B. Le Grand, F. Rossi, Mean absolute percentage error for regression models, Neurocomputing 192 (2016) 38–48.

[27] C. Tofallis, A better measure of relative prediction accuracy for model selection and model estimation, Journal of the Operational Research Society 66 (8) (2015) 1352–1362.

[28] R. J. Hyndman, A. B. Koehler, Another look at measures of forecast accuracy, International journal of forecasting 22 (4) (2006) 679–688.

[29] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky, V. A. Kamaev, et al., A survey of forecast error measures, World Applied Sciences Journal 24 (24) (2013) 171–176.

[30] S. Guan, A. Zhao, A two-factor autoregressive moving average model based on fuzzy fluctuation logical relationships, Symmetry 9 (10) (2017) 207.

[31] F.-M. Tseng, H.-C. Yu, G.-H. Tzeng, Combining neural network model with seasonal time series arima model, Technological forecasting and social change 69 (1) (2002) 71–87.

[32] F.-M. Tseng, G.-H. Tzeng, et al., A fuzzy seasonal arima model for forecasting, Fuzzy Sets and Systems 126 (3) (2002) 367–376.

[33] B. Billah, M. L. King, R. D. Snyder, A. B. Koehler, Exponential smoothing model selection for forecasting, International journal of forecasting 22 (2) (2006) 239–247.

[34] S. Mao, F. Xiao, Time series forecasting based on complex network analysis, IEEE Access 7 (2019) 40220–40229.

[35] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learnin, Cited on (2009) 33.

[36] G. D. Hutcheson, Ordinary least-squares regression, L. Moutinho and GD Hutcheson, The SAGE dictionary of quantitative management research (2011) 224–228.
[37] J. Taylor, R. Lockhart, R. J. Tibshirani, R. Tibshirani, Post-selection adaptive inference for least angle regression and the lasso, arXiv preprint arXiv:1401.3889 354 (2014).

[38] H. Tong, D.-R. Chen, L. Peng, Analysis of support vector machines regression, Foundations of Computational Mathematics 9 (2) (2009) 243–257.

[39] W. Xu, X. Liu, F. Leng, W. Li, Blood-based multi-tissue gene expression inference with bayesian ridge regression, Bioinformatics 36 (12) (2020) 3788–3794.

[40] D. W. Hosmer Jr, S. Lemeshow, R. X. Sturdivant, Applied logistic regression, Vol. 398, John Wiley & Sons, 2013.

[41] A. Defazio, F. Bach, S. Lacoste-Julien, Saga: A fast incremental gradient method with support for non-strongly convex composite objectives, in: Advances in neural information processing systems, 2014, pp. 1646–1654.

[42] C. N. Babu, B. E. Reddy, A moving-average filter based hybrid arima–ann model for forecasting time series data, Applied Soft Computing 23 (2014) 27–38.

[43] S. Panigrahi, H. S. Behera, A hybrid ets–ann model for time series forecasting, Engineering applications of artificial intelligence 66 (2017) 49–59.

[44] Z. Zeng, H. Xiao, X. Zhang, Self cnn-based time series stream forecasting, Electronics Letters 52 (22) (2016) 1857–1858.

[45] J. Cao, Z. Li, J. Li, Financial time series forecasting model based on ceemdan and lstm, Physica A: Statistical Mechanics and its Applications 519 (2019) 127–139.

[46] P. H. Borghi, O. Zakordonets, J. P. Teixeira, A covid-19 time series forecasting model based on mlp ann, Procedia Computer Science 181 (2021) 940–947.