ISSN 2616-5775 Vol. 7, Issue 6: 81-87, DOI: 10.25236/AJCIS.2024.070613

Time Series Prediction Model Based on Convolutional Neural Networks with Attention

Guanchen Du[1,*], Xinghe Chen[2]

[1]College of Mathematics and Computer Science, Shantou University, Shantou, 515000, China
[2]School of Computer Science and Engineering, North Minzu University, Yinchuan, 750021, China
[*]Corresponding author: 22gcdu@stu.edu.cn.

Abstract: *Time series prediction involves forecasting future values or events based on past data patterns and trends, providing a reliable basis for decision-making and planning, with a wide range of application prospects. Currently, many time series prediction techniques face issues such as low accuracy and high time costs, which do not meet societal needs. This paper aims to explore a new method for time series prediction, namely Convolutional Neural Networks (CNN) with an attention mechanism, to capture the key features of time series. Experiments and validations on multiple datasets have been conducted, and the results show that the proposed method significantly improves both accuracy and efficiency.*

Keywords: *Time Series Prediction, Convolutional Neural Network, attention mechanisms*

1. Introduction

A time series is a series of data points arranged in chronological order, such as stock prices, temperature, electricity consumption, etc. Time series prediction is a key issue in many fields, covering multiple application scenarios such as stock markets, weather forecasting, and traffic flow. However, according to the research of [1] and others, traditional time series prediction methods have certain limitations when dealing with nonlinear and complex time series data.

In the current research, many researchers are committed to improving the methods of time series prediction to enhance the accuracy and generalization ability of the model, such as literature 2 and literature 3 and others[2-3]. Traditional statistical models and machine learning methods perform well in some scenarios, but for time series with spatiotemporal correlation and nonlinear trends, traditional models may have certain limitations[4]. Therefore, it is necessary to explore new modeling techniques to better capture complex patterns in time series data. In the field of deep learning, some scholars have attempted to introduce deep learning models into the field of time series prediction, such as WJiang et al. [5] And research by Hajirahimi Z, et al.[6] Convolutional neural networks (CNN) have attracted widespread attention as a powerful feature extraction tool. The success of CNN in image processing has inspired researchers to apply it to time-series data, but research in this field is still in a relatively early stage. Further in-depth research is needed on how to effectively convert time series data into inputs suitable for CNN, as well as the selection of model parameters.

To address the aforementioned issues, this article will delve into how to use CNN with attention mechanism to predict time series. The motivation for this exploration is to fully leverage the attention mechanism's ability to focus on important information in the sequence, in order to improve the model's ability to capture key moments and patterns. In addition, we will also conduct comparative experiments using a Long Short Term Memory Network (LSTM) with attention mechanism on the same dataset to evaluate the performance of the two models in time series prediction tasks.

Through this study, we aim to provide more flexible and efficient modeling options for the field of time series prediction. A deeper understanding of the performance of CNN and LSTM in situations with attention mechanisms can help select the most suitable model for different application scenarios. Ultimately, we aim to improve the accuracy, robustness, and model training time of time series prediction, providing more reliable tools and methods for decision-making and planning in related fields.

ISSN 2616-5775 Vol. 7, Issue 6: 81-87, DOI: 10.25236/AJCIS.2024.070613

2. Construction of model

2.1 The Fully connected layer

Deep learning, as a branch of machine learning, aims to achieve highly abstract representation and understanding of data by simulating the structure and function of human brain neural networks. Deep learning models are typically presented in the form of deep neural networks, which contain multiple levels of neural networks that learn and extract features from different levels by passing data layer by layer. The hierarchical structure of these networks aims to simulate the human brain's ability to process and abstract complex information layer by layer.

Fully Connected Layer is a fundamental neural network hierarchy in deep learning neural networks. Under this operation, each neuron establishes a connection with all neurons in the previous layer, forming a fully connected network structure. This means that each neuron receives the output of all neurons in the previous layer as input, which is weighted by the weights associated with the connection, and each neuron also has an independent bias term to adjust the activation threshold of the neuron. The parameters of the fully connected layer include weights and bias terms, which are learned through backpropagation algorithms during the training process. The optimization algorithm adjusts these parameters by minimizing the loss function, enabling the model to better fit the training data.

The data input to the fully connected layer is usually the output of the previous layer, while the output of the fully connected layer is the weighted sum of each neuron, processed by the activation function. Activation functions (such as ReLU, Sigmoid, etc.) introduce nonlinear characteristics, enabling neural networks to learn more complex functions.

The fully connected layer helps to learn advanced feature representations of input data. Due to the fact that each neuron can be responsible for capturing different aspects of input, this layer has great flexibility in extracting abstract features. A fully connected layer typically appears at the end of a neural network to map advanced features to the final output category or regression value.

2.2 Convolution Neural network

Convolutional Neural Network (CNN) is a deep learning model originally designed to solve computer vision problems. The basic idea is to use convolution operations to locally perceive input data, extract abstract features at different levels, and thus enable the model to have efficient processing capabilities for grid structured data such as images.

In CNN, convolution operations slide on input data through convolution kernels (or filters) to locally perceive specific patterns of input, such as edges, textures, etc. The parameter sharing mechanism of convolutional layers enables the same convolutional kernel to share weights across the entire input, which not only reduces the number of model parameters but also improves the model's generalization ability. As shown in Figure 1.

The pooling layer is another important component of CNN, which downsamples the output of the convolutional layer to reduce the spatial size of the data, retain important information, and reduce computational complexity. Maximum pooling is a commonly used pooling operation that preserves the main features by selecting the maximum value in each region. As shown in Figure 2.

Activation functions (such as Corrected Linear Unit, ReLU) are introduced into the output of convolutional layers to introduce nonlinear characteristics, enabling the model to learn more complex feature maps. This non-linear transformation is crucial for improving the expressive power of the network.

CNN has achieved outstanding results in image processing and other grid structured data due to its hierarchical feature learning, parameter sharing, local perception, and spatial downsampling characteristics. Its design inspiration comes from inspiration for biological visual systems. By imitating the working principles of biological visual systems, CNN is widely used in the field of deep learning. It has not only achieved success in computer vision, but also demonstrated excellent performance in other fields such as natural language processing tasks.

ISSN 2616-5775 Vol. 7, Issue 6: 81-87, DOI: 10.25236/AJCIS.2024.070613

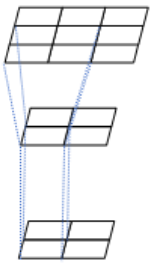

Figure 1: Convolution operation

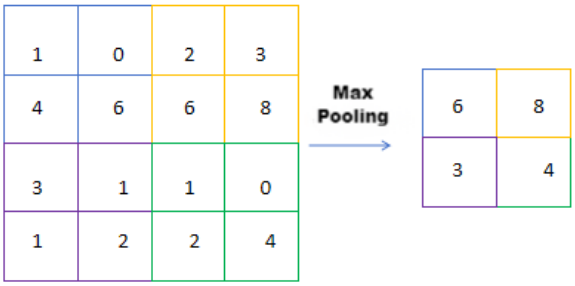

Figure 2: Pooling operation

2.3 Attention

Scaled Dot Product Attention is a form of attention mechanism widely used in deep learning, especially in natural language processing and sequence modeling tasks such as machine translation and language modeling. The design inspiration for this mechanism comes from the way humans pay attention to the relationships between different elements, allowing the model to selectively focus on information at different positions when processing sequence data, in order to better capture key features.

In the click attention mechanism, the focus is obtained by calculating the dot product between the query and key. The result of this dot product will be scaled to ensure a stable score range between query vectors in different dimensions. By scaling, the numerical value of attention score can be reduced, thereby improving the numerical stability of the model.

Specifically, consider the query matrix Q, key matrix K, and value matrix V of the input sequence. The calculation process of click attention mechanism is as follows:

$$Attentio\,n(Q,K,V) = softma\,x\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

Among them, represents the dimension of the query vector. In this formula, the dot product of the query and key is divided by this scaling factor, and then subjected to a Softmax operation to obtain the final attention weight. These weights are used to weight and sum the value matrix to obtain the final output.

Overall, click attention mechanism is a key mechanism used in deep learning to handle sequence relationships. Its scaling operation helps to improve the numerical stability of the model, making it more suitable for sequence modeling tasks in practical applications.

2.4 Overall architecture of the model

We use the structure shown in the following figure to model the time series. This model captures local features of the input sequence through convolutional layers, focuses on key time steps by introducing attention mechanisms, and finally maps these features into numerical predictions through fully connected layers. The design of this structure aims to better capture complex patterns in time series

ISSN 2616-5775 Vol. 7, Issue 6: 81-87, DOI: 10.25236/AJCIS.2024.070613

to improve predictive performance. Architecture of the model is shown as figure3.

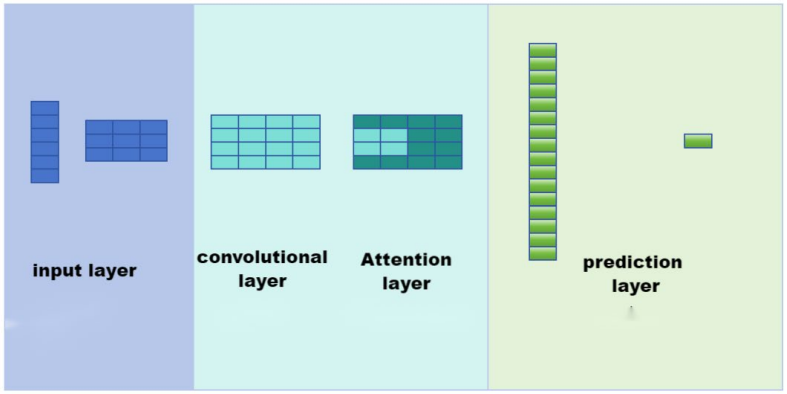

Figure 3: Architecture of the model

2.4.1 Input layer

This layer receives time series data as input, where the information for each time step is represented as a vector. These vectors are used as inputs to the model and passed on to subsequent convolutional layers.

Firstly, this thesis use the Pandas library to read the data and convert its data type to a floating-point number. This data is the training and testing set we used in the model. We divide the data into two parts, with the first m time steps used for model training as the training set and the last n time steps used as the validation set for the model.

Subsequently, we standardized the data from the training set. Perform sequence length segmentation on the standardized data table, where the parameter seq represents the sequence length and the parameter ws represents the window size, which is the number of historical time steps used for prediction. By using the sliding window method, we divide the time series data into input windows and corresponding target values (labels). The samples for each time step include data from the past 12 time steps as input, while the values for the next time step become labels. This processing flow aims to convert data into a format that is easy to process.

2.4.2 Convolutional layer

Next, this thesis use Convolutional Neural Networks (CNN) to extract features from images. Convolutional layers are used to capture local features and patterns in the input time series. By performing convolution operations on the input, the model can learn spatial relationships in the data. After the convolution operation, nonlinear transformations are introduced through activation functions.

In the CNN module, there are convolutional layers and T convolutional kernels. Each convolution kernel can extract a type of latent feature from the input matrix. In each convolutional layer, the input matrix of the layer can be represented as a matrix, where is the identifier of the input matrix layer. When=0, the input matrix is the training set matrix obtained from the original matrix in the first step, and the output of the layer is used as the input of the layer. Therefore, the features extracted by the Lc layer's th convolution kernel can be described using the following formula:

$$\mathrm{E}_{\mathrm{i}}^{\mathrm{l_c}} = \mathrm{relu}\left(\omega_{\mathrm{i}}^{\mathrm{l_c}} \otimes \mathbb{M}^{\mathrm{l_c}-1} + \mathrm{b}^{\mathrm{l_c}}\right) \tag{2}$$

Where $\omega_{\mathrm{i}}^{\mathrm{L_c}}$ is the L_c weight matrix of the i-th convolutional kernel in the layer; For its corresponding bias term b^{L_c}, $\otimes$ for convolution operations; The process of this convolution operation involves continuously iterating the L_c layer to gradually extract and learn abstract features from the input data.

In the 0th layer of our model, the dimension of the input matrix is: where 32 represents the small batch size we use in training the model; The next 1 represents the embedding dimension of time series prediction as 1; The last 12 represents our prediction for 12 consecutive values. Next, we perform two convolution operations using a convolution kernel of size, which changes the number of channels in the matrix. Subsequently, we adjusted its dimension to a maximum pooling operation, where 64 represents the number of channels output by the convolution. This processing step helps to extract key features from

ISSN 2616-5775 Vol. 7, Issue 6: 81-87, DOI: 10.25236/AJCIS.2024.070613

input data and preserve useful information while changing dimensions.

2.4.3 Attention mechanism layer

In order to enable the model to dynamically focus on different parts of the input sequence and more effectively capture important information in the sequence, we introduce a scaled dot product attention mechanism.

In the initialization method, we defined the basic parameters of the attention mechanism by setting the scaling factor and optional dropout ratio. In the forward propagation method, attention weights are generated by calculating the dot product between the query and key, applying a scaling factor, and using the softmax function to obtain attention scores. In our model, we ignored the mask as the entire input sequence is relevant to the regression task. Finally, by multiplying the attention weights with values, we obtain the final output and increase the robustness of the model through optional dropout operations. The main purpose of this module is to provide a reusable attention mechanism component that can be embedded into other neural network structures to capture important information in input sequences. It is worth noting that in this layer, the attention mechanism we used did not change the size of the input and output layers.

2.4.4 Prediction layer

The design of the prediction layer aims to transform the feature maps extracted by convolution and attention mechanisms into numerical predictions. We choose to use a fully connected layer to achieve this goal, and its expression is as follows:

$$\begin{aligned} \mathbf{X}_f &= f(\,output\,) \\ y_1 &= \text{relu}\left(\mathbf{W}_1\mathbf{X}_f + \mathbf{b}_1\right) \\ y_{L-1} &= \text{relu}(\mathbf{W}_{L-1}y_{L-2} + \mathbf{b}_{L-1}) \\ y_L &= \mathbf{W}_L y_{L-1} + \mathbf{b}_L \end{aligned} \tag{3}$$

Among them, $output$ represents the output results of convolution and attention mechanisms, where f represents the operation of expanding it into ne bit vector size, $hidden\_size$ represents the input dimension of this layer, and $W_i\&b_i$ represents the weight vector and bias vector of the i-th layer, respectively. These parameters are used for linear transformation of features. The activation function relu is used to introduce nonlinear transformations, making the results nonlinear and ultimately obtaining predictive results. The purpose of this structure is to transform the underlying feature maps into final numerical predictions by learning appropriate weights and biases.

3. Analysis of Experimental Findings

3.1 Datasets and Data Preprocessing

In this study, we used a publicly available dataset from "excluding commercial wholesalers, manufacturer sales branches, and offices: non durable goods: sales of beer, wine, and distilled alcoholic beverages." This dataset includes monthly sales of alcoholic beverages for the company from 1992 to 2015. Each data entry consists of two parts: time (e.g. June 1992) and the corresponding monthly sales revenue of the company (e.g. 199.3). Our goal is to predict the sales revenue of alcoholic beverages for the next month by using specific length of time series data.

In this experiment, we first use the read_csv function in the Pandas library to read the dataset. Then, we use the Torch library to convert the dataset into tensors. Next, we use the isnan method to find missing values and use the nanman method to average fill in the missing values. Finally, we normalize the data to ensure that all values are within the range of [0,1]. Then, we use the first 70% of the data as the training set and the last 30% as the testing set. We use a time window size of 12 to segment the training and testing sets. Each window contains 13 data points, with the first 12 used as inputs for the model. We use these 12 data points to predict the next value, and then optimize the predicted value to minimize the difference with the actual next value. This method can effectively train and test our model, capture time patterns using a time window, and make accurate predictions.

3.2 The experimental setup employed was as follows

The experimental setup used the Windows 11 Professional operating system. The CPU processor used is the 13th generation Intel (R) Core (TM) i7-12700H, and the GeForce RTX 3060 GPU has 6 GB

ISSN 2616-5775 Vol. 7, Issue 6: 81-87, DOI: 10.25236/AJCIS.2024.070613

of memory. The model training framework uses Python 3.8.5 with a torch architecture of '1.10.0+cu113'.

3.3 Evaluation metrics

In this experiment, we used Mean Squared Error (MSE) as the evaluation metric for our model, represented as follows

$$MSE = \frac{1}{\sum_{i=1}^{n}(predict_i - target_i)^2} \tag{4}$$

MSE is obtained by calculating the square of the difference between the predicted and actual values of each sample, and taking the average value. This square difference calculation method is more sensitive to larger errors, so MSE is a commonly used loss function in regression problems. During the training process, the optimization objective is to minimize MSE in order to make the model's predictions as close as possible to the actual target values.

3.4 Result and Analysis

Firstly, we visualize our data to be predicted, as shown in Figure 4

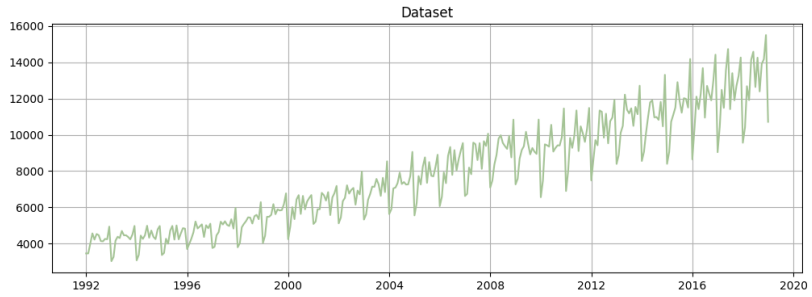

Figure 4: Dataset

We first used convolutional neural networks to train and test this dataset, and the test results obtained are shown in Figure 5, which are consistent with the actual results.

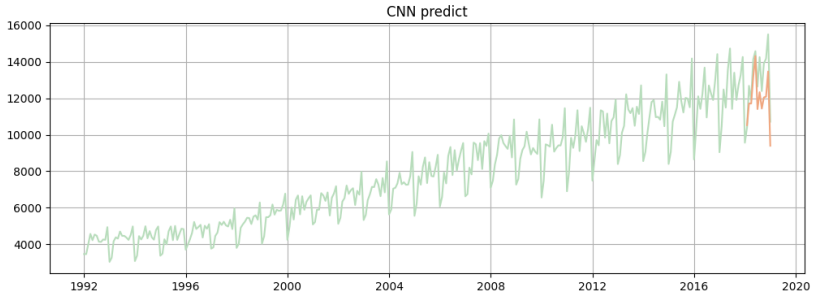

Figure 5: Prediction result

We will enlarge the rear part as shown in Figure 6.

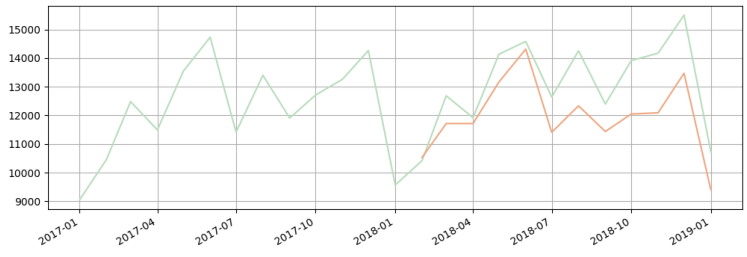

Figure 6: Enlarge the prediction results section

The above image clearly shows that our time series prediction model has a relatively high accuracy

ISSN 2616-5775 Vol. 7, Issue 6: 81-87, DOI: 10.25236/AJCIS.2024.070613

and can also capture the dynamic changes of data over time. This is mainly related to our design of networks with attention mechanisms; In CNN models with attention mechanisms, these models can more sensitively capture important patterns and trends in the input sequence by focusing on specific parts. By introducing attention mechanisms, we expect the model to automatically learn which time steps are more critical for prediction tasks, thereby improving its adaptability to complex time series.

We also compared our network with the LSTM architecture, and the comparison results are shown in Table 1

Table 1: Comparative experimental results

model name	best_train_loss	average_test_loss	time
our model	0.000150	0.0019469	88s
LSTM model	0.000600	0.0041972	122s

The above table shows that our model not only has lower loss values during training, but also has strong accuracy on the test set, and also has an advantage in time efficiency.

In summary, our study provides a new perspective and empirical results for model selection in time series prediction tasks by visualizing the predicted data and comparing them with convolutional neural networks with attention mechanisms and long short-term memory networks (LSTM). Firstly, we demonstrate the superiority of CNN models that introduce attention mechanisms in capturing dynamic changes in complex time series. Specifically, the model not only exhibits high accuracy on the test set, but also can more sensitively capture important patterns and trends in the sequence, thereby improving the model's adaptability and generalization ability. Secondly, we compared the differences between CNN models and traditional LSTM models in terms of training loss, testing accuracy, and time efficiency. The results indicate that our CNN model has achieved satisfactory results in all indicators and has a significant advantage in training time.

4. Conclusion

Through this study, we delved into the effectiveness of convolutional neural network models that introduce attention mechanisms in time series prediction tasks, and compared them with traditional long short-term memory networks. Our empirical results indicate that the introduction of attention mechanism in CNN models significantly improves the prediction accuracy and generalization ability of the model, and has significant advantages in training time.

However, we are also aware that there are still some limitations and room for future improvement in this study. For example, our model may still be affected by data noise, and further data preprocessing and model tuning may help improve the performance of the model. In addition, our research has not fully explored the applicability of attention mechanisms in other types of neural network models, which is also an important direction for future research.

References

[1] Wang X, Kang Y, Hyndman R J, et al. Distributed ARIMA models for ultra long time series [J]. International Journal of Forecasting, 2023, 39 (3): 1163-1184
[2] Masini R P, Medeiros M C, Mendes E F. Machine learning advantages for time series forecasting [J]. Journal of Economic Surveys, 2023, 37 (1): 76-111
[3] Ning Y , Wang X , Yu Q ,et al.Rapid Damage Prediction and Risk Assessment for Tropical Cyclones at a Fine Grid in Guangdong Province, South China[J].International Journal of Disaster Risk Science, 2023, 14(2):237-252.DOI:10.1007/s13753-023-00485-y.
[4] Meiting C , Yaoqiu K , Ningsheng H .Early-warning of Spatiotemporal Evolvement of Land Ecological Security in Guangdong Province Based on RBF[J].Research of Soil and Water Conservation, 2015.
[5] Jiang W, Ling L, Zhang D, et al. A time series forecasting model selection framework using CNN and data augmentation for small sample data [J]. Neural Processing Letters, 2023: 1-28
[6] Hajirahimi Z, Khashii M. Hybridization of hybrid structures for time series forecasting: A review [J]. Artistic Intelligence Review, 2023, 56 (2): 1201-1261