electronics

Article

A Multivariate Temporal Convolutional Attention Network for Time-Series Forecasting

Renzhuo Wan [1,2,†], Chengde Tian [1,†], Wei Zhang [1], Wendi Deng [1] and Fan Yang [3,*]

[1] School of Electronic and Electrical Engineering, Wuhan Textile University, Wuhan 430200, China; wanrz@wtu.edu.cn (R.W.); 2015363074@mail.wtu.edu.cn (C.T.); wzhang@wtu.edu.cn (W.Z.); wddeng@wtu.edu.cn (W.D.)
[2] Hubei Key Laboratory of Digital Textile Equipment, Wuhan Textile University, Wuhan 430200, China
[3] School of Mathematical and Physical Sciences, Wuhan Textile University, Wuhan 430200, China
* Correspondence: yangfan@wtu.edu.cn
† These authors contributed equally to this work.

Abstract: Multivariate time-series forecasting is one of the crucial and persistent challenges in time-series forecasting tasks. As a kind of data with multivariate correlation and volatility, multivariate time series impose highly nonlinear time characteristics on the forecasting model. In this paper, a new multivariate time-series forecasting model, multivariate temporal convolutional attention network (MTCAN), based on a self-attentive mechanism is proposed. MTCAN is based on the Convolution Neural Network (CNN) model, using 1D dilated convolution as the basic unit to construct asymmetric blocks, and then, the feature extraction is performed by the self-attention mechanism to finally obtain the prediction results. The input and output lengths of this network can be determined flexibly. The validation of the method is carried out with three different multivariate time-series datasets. The reliability and accuracy of the prediction results are compared with Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Convolutional Long Short-Term Memory (ConvLSTM), and Temporal Convolutional Network (TCN). The prediction results show that the model proposed in this paper has significantly improved prediction accuracy and generalization.

Keywords: multivariate time-series forecasting; self-attention mechanism; deep learning; neural network

Citation: Wan, R.; Tian, C.; Zhang, W.; Deng, W.; Yang, F. A Multivariate Temporal Convolutional Attention Network for Time-Series Forecasting. *Electronics* **2022**, *11*, 1516. https://doi.org/10.3390/electronics11101516

Academic Editors: Phivos Mylonas, Katia Lida Kermanidis and Manolis Maragoudakis

Received: 21 April 2022
Accepted: 29 April 2022
Published: 10 May 2022

1. Introduction

A multivariate time series is an important data object, which is a series of observations formed by multivariate variables recorded in chronological order. Multivariate time series are used in more and more fields, such as the environment [1,2], finance [3,4], transportation [5–7], healthcare [8], and energy [9,10]. In these fields, time-series prediction is used to monitor some critical data and avoid the occurrence of unforeseen situations that cause economic losses. For multivariate time-series prediction tasks, early solutions mainly choose recurrent networks, but recurrent networks suffer from gradient disappearance and gradient explosion problems, due to which the long-term dependence problem of RNNs [11,12] cannot be solved. The time-series structure on the one hand makes it difficult to have efficient parallel computing capability (the computation of the current state depends not only on the current input but also on the input of the previous state), and on the other hand makes the RNN model, including variants of LSTM [13], GRU [14], etc., more similar to a Markov decision process [15] in general and difficult to extract global information. In addition, CNN models [16] have started to be applied to sequence modeling. For multivariate time-series [17] problems, these models also have difficulty capturing the mapping relationships between multiple variables as well as adapting to complex data features.

Traditional CNNs are generally considered less suitable for modeling time-series problems, which is mainly due to the limitation of convolutional kernel size and thus

cannot capture long-time dependent information well. With the development of deep learning, some specially processed convolutional neural networks can also achieve good results for time-series modeling. The TCN model [18] based on CNN model uses causal and inflation convolution and residual modules [19] to make it suitable for temporal modeling tasks, and TCN can reach or even surpass the RNN model in many tasks. In contrast to the RNN model, the CNN model has no temporal structure and can perform parallel computations to maximize the use of computing power. Our goal is to explore a better architecture based on CNN with attention mechanism and feedforward neural networks to achieve an approximate replacement of recurrent networks and improve training efficiency while ensuring effectiveness for multivariate time-series problems.

Inspired by TCN and the attention mechanism [20], in this paper, we propose a MTCAN model for multivariate time-series prediction. In this MTCAN, we use a feedforward neural network as the base unit to construct residual blocks; then, we enhance the interpretation of features by an asymmetric residual block network [21] and finally perform feature extraction by a self-attentive mechanism. The paper is organized as follows: Section 2 reviews the background of the work. Section 3 describes the modeling approach. The experiments are analyzed and discussed in Section 4. Finally, conclusions and outlook are drawn in Section 5.

2. Background and Related Work

In the task of time-series prediction [22,23], researchers have proposed many solutions. Various models have been developed from the earliest classical statistical-based methods to the current deep learning algorithms. In 1927, the British statistician G.u. Yule proposed the AR (Auto Regressive) model [24,25]. In 1931, G.T. Walker proposed the MA (Moving Average) model and the ARMA model [26,27], which formed the basis of time-series analysis. Subsequently, Box and Jenkins discussed the ARIMA (an autoregressive integrated moving average) model [28]. All four models require the time series to be univariate, homoskedastic linear models. In recent years, techniques such as machine learning and neural networks have developed rapidly, and these new methods have been applied to time-series forecasting. In 1998, White applied the neural network approach to time-series forecasting. Vladimier N. Vapnik proposed the original support vector machine [29] and used it in capital cost estimation. In 2006, Geoffery Hinton and Ruslan Salakhutdinov proposed a solution to the gradient disappearance problem in deep network training, and deep neural networks came back into the limelight. In particular, CNNs and RNNs have received widespread attention. Convolutional neural is widely used in image recognition, and recurrent neural network is widely used in sequence modeling.

The main approach in deep learning to deal with prediction problems is recurrent neural networks. However, since recurrent neural networks suffer from gradient disappearance or explosion, they cannot solve the long-range dependence problem. For this problem, Hochreiter and Schmidhuber proposed the long short-term memory network [13]. For the gradient disappearance problem, a gate mechanism is used to solve it. For the problem of short-term memory overwriting long-term memory, LSTM adopts a cell state to preserve long-term memory and then cooperates with the gate mechanism to filter the information to achieve the control of long-term memory. The gated recurrent cell network was proposed by Cho et al. GRU can be regarded as a simplified version of LSTM. For LSTM and GRU, the iterative process can be greatly accelerated because GRU has fewer parameters and converges faster.

Although CNNs are generally tasked with image classification, with dedicated design, they have been confirmed to be significant tools for sequence modeling and prediction. Bai et al. proposed time-domain convolutional networks, which consist of dilated, causal 1D convolutional layers with the same input and output length. They were able to show that in many tasks, convolutional networks can achieve better performance than that of RNNs in avoiding common drawbacks of recursive models, such as the gradient explosion/disappearance problem or the lack of memory retention. For a multivariate time-series

task featuring the dataset of multivariate series at time step t, depending on the previous data point, any two data points may be correlated, and the data within the data points may be correlated. Therefore, a feasible multivariate time-series model should describe not only the correlation between the elapsed relationships and data points as in a univariate time series but also the correlation of the data within the data points.

Wan et al. proposed an M-TCN model [30] in solving multivariate time-series forecasting problems. It is proved to have better performance on multivariate time-series tasks, but M-TCN uses fully connected layers, which leads to a large number of parameters in the model and thus makes the model match more time-consumption.

The self-attention mechanism is a new attention mechanism proposed by Ashish Vaswani et al. The attention mechanism essentially assigns a weight factor to each element of the sequence, which can also be understood as soft addressing. If each element in the sequence is stored as (K, V), then the attention mechanism accomplishes addressing by computing the similarity between Q and K. The similarity computed between Q and K reflects the importance of the taken out V value. The difference between the self-attention mechanism and the attention mechanism is that the self-attention mechanism reduces the dependence on external information, and it is the elements in the sequence that find the similarity themselves. Such a mechanism enhances the capture of dependencies. We adopt the self-attention mechanism for feature extraction here to improve the effectiveness of the model and reduce the number of parameters.

In this context, our model uses a TCN-based design with a self-attention mechanism for feature extraction. It is tested under three datasets in the areas of PM2.5 prediction and electricity prediction as well as weather prediction.

3. Methodology

For the multivariate time-series forecasting task, we first describe the definition and construction of a sequence model. What we highlight is the idea and structure of the proposed model MTCAN by incorporating a self-attention mechanism.

3.1. Sequence Problem Statement

Deep learning is essentially the use of deep neural networks to fit the complex nonlinear relationships between data and labels. In order to obtain the desired nonlinear mapping during model learning, a large amount of data is needed for learning to extract the features of the data. A multivariate time-series forecast is actually a sequential prediction problem [31] as well. Suppose the input sequence is $x_{1:T} = x_1, x_2, \ldots, x_T$ with length T and the target sequence is $y_{1:H} = y_1, y_2, \ldots, y_H$ with length H. The goal of model is to construct a nonlinear mapping of the predicted time series from the current state:

$$y_{1:H} = SeqMod(x_{1:T}) \tag{1}$$

It is important to note a constraint that y_h should satisfy the causal constraint to prevent future information $x_{t>h}$ from leaking. The length of the input and output may not be the same. The *SeqMod* is essentially to find a neural network with the best prediction result.

In the traditional time-series modeling process, RNNs are generally chosen for sequence modeling because of the reliance on past information for sequence modeling. However, with the development of feedforward models, researchers have found that by applying special treatments to feedforward models, they can also be used for sequence modeling and can take advantage of parallel training.

3.2. Model Structure

For the time-series prediction problem, the memory of past information is required, and CNNs in general do not have the ability to remember. We generally believe that RNN models, such as LSTM, are the best standard approach to solve time-series prediction problems; however, CNN models can save a lot of time by being able to perform parallel operations compared to RNN models. Based on these considerations, we designed the

general framework inherited from CNNs. The goal is to obtain the best structures of convolutional network design as a flexible and stable framework for multivariate time-series forecasting. We denote the proposed network structure as the multivariate temporal convolutional attention network, MTCAN. The salient features of MTCAN are (1) a one-dimensional convolution is used instead of causal convolution, (2) a Residual Connect structure is used, (3) an attention mechanism is used to capture features, and (4) the input and output length of the model can be determined flexibly. The MTCAN network structure is in general a multi-headed network structure. In this work, we are using the asymmetric residual blocks and the attention mechanism to construct an effective network approach. The detailed structure of the model is as follows.

3.2.1. 1D Dilated Convolution

In causal convolution [32], the output at time t is only convolved with elements from the previous layer at time t and earlier. One major drawback of this network design is that in order to get a large enough perceptual field to obtain information over a long period of time, we need a very deep network or a very large filter, which would make the model too large. In causal convolution, it has a one-to-one causal relationship between input and output. For time-series problems, this design leads to no parallel computation during the operation, which makes feature learning inefficient. The one-dimensional convolutional network is used to avoid this situation and to improve the feature learning efficiency.

However, pooling is used in traditional 1D convolutional networks, which is designed to expand the perceptual field and reduce the size of the sequences. An obvious drawback of pooling is sequence feature loss in the information-merging process. The expanded convolution increases the range of the filter without increasing the number of weight parameters in it. Thus, the inflation convolution increases the perceptual field of the neural network without increasing the computational cost. Thus, for long information-dependent problems inside a time series, the dilation convolution can be well applied. It is because of these advantages of one-dimensional dilated convolution [33] that we adopt it as the basic unit of our model here. We can define the 1D dilated convolution as:

$$Out(k,q) = \sum_{c} \sum_{w+d*s=q} In(c,w) * Weight(k,c,s) \tag{2}$$

where $In(c,w)$ is the input vector, $Out(k,q)$ is the output vector, $Weight(k,c,s)$ is the filter size, and d is the expansion factor. Figure 1 shows the process of a dilated convolution.

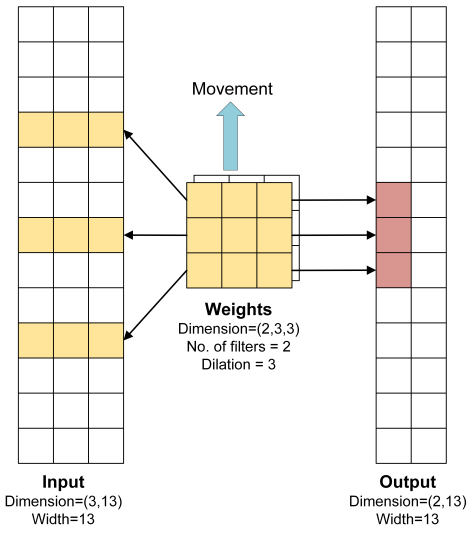

Figure 1. An example of the 1D dilated convolution layer with input channels $C = 5$, input width $W = 13$, number of filters $K = 2$, filter width $S = 3$, output with $Q = 13$, and dilation parameter $d = 3$.

3.2.2. Residual Connect

The use of convolutional neural networks for time-series prediction requires considering the problem of gradient disappearance due to the large number of layers in multilayer convolutional networks, while we use residual blocks [34] to build the model to avoid this problem and to improve the model by deepening the number of layers. A residual block contains a branch that leads to a series of transformations whose output adds the input of the residual block. We design a novel structure by multi-layer ordered residual networks and parallel residual networks. The core of the residual block is to create a shortcut between the front and back layers and does not introduce additional parameters or computational complexity. Whereas the hopping connections in ResNet lead to the fact that not all residual blocks are functional, we use direct connections to ensure that each residual block learns useful information.

To increase the effectiveness of the neural network, the size of the convolutional kernel can be increased or the depth of the network can be increased, but this would make the computation very large. We have taken the approach of invoking the asymmetric block structure [35], which will create an asymmetric factor in the structure of the whole network and have a positive effect on the whole model.

The structure of our residual block is shown in Figure 2. Our input goes into two channels; then, we start the expansion of the 1D convolution first, then correct it by correcting the linear unit, and finally sum the output. This process is repeated three times in residual block 1 and four times in residual block 2. In this way, residual block 1 and residual block 2 form an asymmetric structure. Since the dimensionality of the final feature mapping may be different, 1×1 convolution is introduced to adjust the dimensionality.

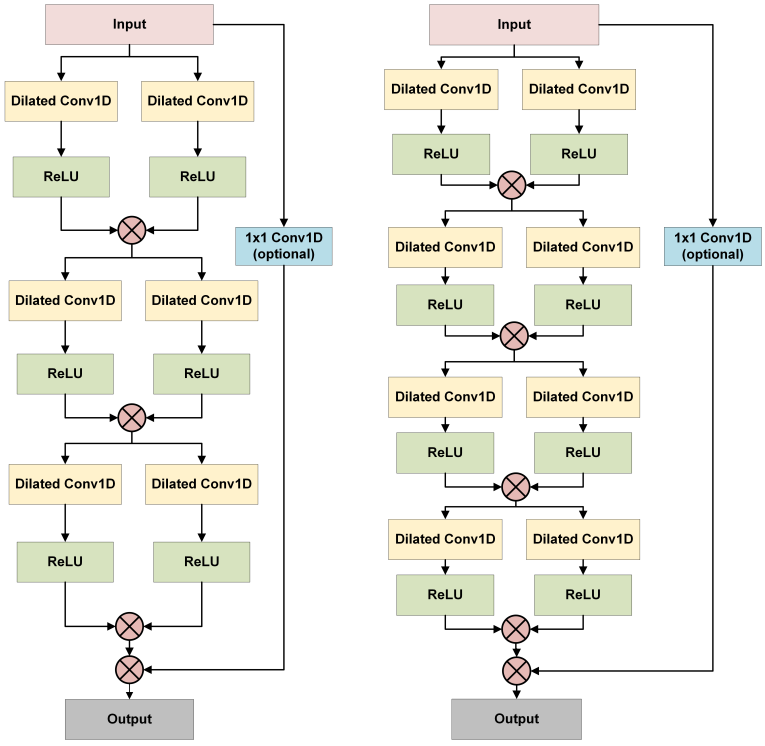

Figure 2. Residual Block 1 (**left**). Residual Block 2 (**right**).

The operation process in the repetition cell is as follows.

$$y_{1k} = ReLU(W_k * X + b_k) \tag{3}$$

$$y_{2k} = ReLU(W_k * X + b_k) \tag{4}$$

$$y_k = y_{1k} + y_{2k} \tag{5}$$

The last step is to add the input to the output of the block.

$$Output = (Input + f(Input)) \tag{6}$$

3.2.3. Self-Attention

The attention function can be described as mapping a query and a set of key–value pairs to an output, where the query, key, value and output are all vectors. The keys and queries are dotted multiplied to obtain the corresponding attention weights, and finally, the obtained weights and values are dotted to obtain the final output. For self-attention, the three matrices Q (Query), K (Key), and V (Value) are all from the same input. We first compute the dot product between Q and K and then divide the result by a scale $\sqrt{d_k}$ to prevent the result from being too large, where d_k is the dimensionality of a query and key vector. The result is then normalized using the Softmax operation and then multiplied by the matrix V to obtain the representation of the weight summation. This computational procedure can be expressed as follows.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{7}$$

Figure 3 shows the process of calculating attention weights. The self-attention mechanism is a variation of the attention mechanism, which reduces the reliance on external information and is better at capturing the internal relevance of data or features. Compared with RNN and its variant models, it is able to compute in parallel and can make better use of computing power. We use the self-attention mechanism in MTCAN to capture the internal correlation of long time series, which can better describe the internal correlation of time series and improve the performance of the whole model.

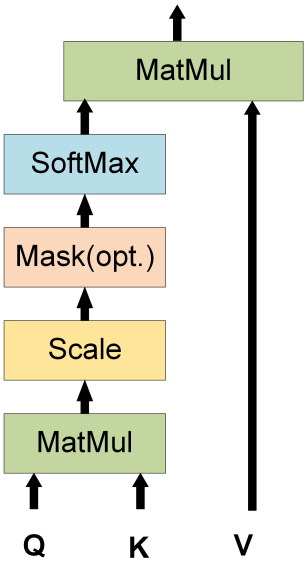

Figure 3. Scaled Dot-Product Attention.

3.2.4. MTCAN Model

Multivariate time series are actually multiple unitary time series, but there is a certain mapping relationship between these multiple time series. It is due to this structural feature that we adopt a structure similar to that of a multi-headed attention network. The difference with the multi-headed attention network is that we split the multivariate time series into multiple univariate time series and use each univariate time series as the input of each repeat block. This way, instead of using an identical sequence as the input of each repeat block, the mapping relationship between the multivariate time series can be captured more effectively.

We use a convolutional neural network as the basis of the model, and we need as deep a network as possible to solve the long time information dependence. However, deeper convolutional neural networks may experience gradient disappearance or add multiple layers without improving the model effect. To solve this problem, a residual block is used, which ensures that the depth of the convolutional neural network is deep enough to remember the information for a long time.

The information learned in each repetition block has different features for each individual variable. Moreover, in the repeat module, we first interpret by residual blocks; then, we extract features from the interpretation by self noticing, and finally, we merge the output vectors of each repetition block. The finally obtained variables are interpreted by the fully connected layer, and then, the prediction results are obtained. The structure of the multivariate temporal convolutional attention base network is shown in Figure 4.

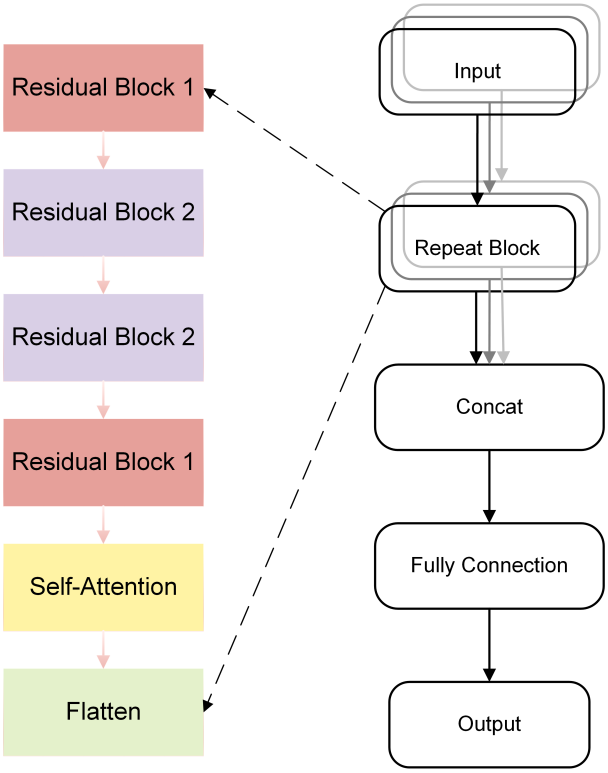

Figure 4. Overall structure of MTCAN model.

4. Experiments

To evaluate the effectiveness of MTCAN, we perform experiments on a multivariate time-series forecasting task. The three datasets are firstly described for the empirical study. All data can be downloaded online. Then, the parameters turning as well as the evaluation criteria are described. Finally, the proposed MTCAN model is compared with different models.

4.1. Data Preprocessing

Three datasets were used in this study, namely the ISO-NE dataset (ISO-NE Dataset available online: https://www.iso-ne.com/isoexpress/web/reports/load-and-demand, accessed on 1 June 2021), Beijing PM2.5 dataset (Beijing PM2.5 Dataset available online: https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data, accessed on 12 June 2021), and Jane Climate dataset (Jena Climate Dataset available online: https://storage.googleapis.com/tensorflow/tf-keras-datasets/jena_climate_2009_2016.csv.zip, accessed on 22 June 2021).

The ISO-NE dataset includes hourly demand, price, temperature and other features. The time frame of this dataset is from 2003 to 2014. We use two variables from this dataset, namely, hourly electricity demand and dry bulb temperature. For this dataset, hourly electricity demand is used as the forecast value.

The Beijing PM2.5 dataset has eight features that include dew point, temperature, atmospheric pressure, etc. The time range of this dataset is from 2010 to 2014. For this dataset, temperature is used as the predicted value.

The Jena Climate dataset contains 14 different features such as temperature, atmospheric pressure, and humidity, among others. We used only the data collected between 2009 and 2016, and these data were collected every 10 min. We used the first 10 variables of this dataset and extracted the data so that the time interval was 1 h. For this dataset, hourly temperatures were used as predicted values.

Table 1 shows the length of the time series, the number of variables, and the sampling interval for the three datasets. There are cases where the values in the dataset are null, so preprocessing is required. For some "NA" values in the dataset, we use 0 instead.

Table 1. Dataset statistics.

Dataset	Length of Time Series	Total Number of Variables	Sample Rate
ISO-NE	103,776	2	1 h
Beijing PM2.5	43,824	8	1 h
Jane Climate	70,080	10	1 h

4.2. Experimental Details

Algorithm 1 is an algorithm for learning rate iteration, where the learning rate decreases for every eight periods of no improvement in the validation score before the minimum learning rate is reached. This algorithm is used for all model learning rate iteration processes. For the multivariate time-series forecasting task, most models are chosen from {24, 72, 144} in length with a batch size of 100. MSE is used as a default loss function for the forecasting task. The optimization strategy uses Adam, and the initial learning rate is set to 0.001.

An LSTM model with a hidden layer of {50, 100, 200} cells is defined. The number of cells in the hidden layer is independent of the number of time steps in the input and output sequences. The final output represents the prediction results for the next 24 h. The optimizer is SGD [36]. The learning rate is set to 0.05, and the reduction rate is 0.3.

For the GRU model, we used a hidden layer of {64, 128, 256} units. The final output is a vector with 24 elements, which is the predicted result for the next 24 h. Adam [37] is used as the optimizer. The learning rate and reduction rate are the same as those of the LSTM model.

In the ConvLSTM encoder–decoder model, the shape of the input data is [*timestep*, *row*, *column*, *channel*]. *Timestep* is selected from {1, 3, 7}. *Row* is set to 1. *Column* is selected from {24, 72, 144}. *Channel* is selected from {2, 8, 10}. The optimization algorithm uses SGD. The learning rate is also set to 0.05. The size of all input-to-state and state-to-state kernels is 1×3.

For the TCN network, a hidden layer of {30, 50, 100} units is defined. The size of the convolution kernel is 1×3.

The MTCAN model uses Adam as the optimization strategy, and the initial learning rate is set to 0.001.

The MTCAN implementation is built on the Keras (Chollet, F. (2015). Keras. GitHub Repository: https://github.com/fchollet/keras, accessed on 10 March 2021) library and the Python-based TensorFlow backend. We conducted experiments on a machine with an NVIDIA 1080GPU (Santa Clara, CA, USA).

Algorithm 1 Algorithm of learning rate

Input: $min\_lr, init\_lr$
Output: new_lr
$factor = 0.7; epoch = 200; new\_lr = init\_lr; n$
if $n < epoch$ **then**
  $n+ = 1; wait+ = 1$
  if $best\_socre > cur\_score$ **then**
    $best\_lr = cur\_score; wait = 0$
    save model
  else
    if $wait == 8 \&\& new\_lr > min\_lr$ **then**
      $new\_lr = new\_lr * factor$
      $new\_lr = max(new\_lr, min\_lr)$
      $wait = 0$
    end if
  end if
end if
return new_lr

We use three evaluation metrics here, root-mean-square error (RMSE), relative root error (RRSE), and empirical correlation coefficient (CORR) of multivariate prediction to assess the performance of the model. The formula for calculating the three indicators is as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(P_i - T_i)^2} \tag{8}$$

$$RRSE = \sqrt{\frac{\sum_{i=1}^{n}(P_i - T_i)^2}{\sum_{i=1}^{n}(P_i - \bar{T})^2}} \tag{9}$$

$$CORR = \frac{1}{n}\sum_{i=1}^{n}\frac{\sum_t (P_{it} - \bar{P}_i)(T_{it} - \bar{T}_i)}{\sqrt{\sum_t (P_{it} - \bar{P}_i)^2 (T_{it} - \bar{T}_i)^2}} \tag{10}$$

In the formula, P_{it} is the predicted value and T_{it} is the actual value, $\bar{P}_i$ is the mean value of the predicted value and $\bar{T}_i$ is the mean value of the actual value. The smaller the value of RMSE and RRSE, the more accurate the prediction result of the model. The larger the value of CORR, the stronger the correlation between the predicted result and the actual value of the model. From the values of these three evaluation parameters, we can also evaluate the validity and accuracy of a model more clearly.

The training process of the model is shown in the Figure 5.

Step 1: Preprocess the time-series dataset. The original time-series dataset is divided into a training set and a test set, while the inputs and outputs are extracted from them separately.

Step 2: Initialize the parameters and hyperparameters of the proposed model.

Step 3: Train the MTCAN time-series prediction model. The Adam optimization algorithm and the mean squared loss function are used. The mean squared loss function is obtained from the predicted and actual values. The RRSE is calculated to save the model with the smallest RRSE, and the learning rate is changed if the loss value has been unchanged.

Step 4: Terminate the model training or loop the third step. If the maximum number of training times is reached, the training process of MTCAN is ended, and the optimized weights and biases of the model are obtained. If not, repeat step 3 until the termination condition is met.

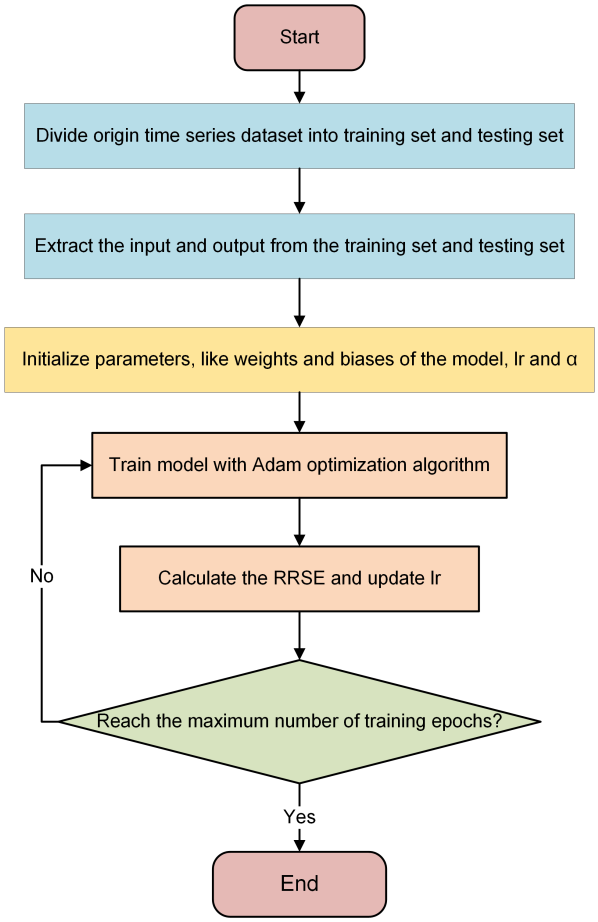

Figure 5. The flowchart presenting the training procedures.

4.3. Experimental Results

In this section, we evaluate the performance of the prediction model over different time horizons. The ISO-NE dataset, Beijing PM2.5 dataset, and Jana Climate dataset are used for prediction. We use three datasets with different numbers of parameters, different domains, and different amounts of data, so that we can evaluate a model more comprehensively. The time prediction range is between 1 and 24 h. This section compares the performance of our proposed MTCAN method with LSTM, TCN, ConvLSTM, and GRU.

Table 2 shows the results of the overall RRSE, CORR, and RMSE metrics for the multivariate test set from 1 to 24 h. The output sequence length is set to 24, representing the prediction time period from 1 to 24 h. In multivariate time-series prediction tasks, the longer the prediction time, the more difficult the prediction is. Therefore, our experiments were performed to analyze the results in detail within this time frame. The best results in each dataset are highlighted in bold. For RMSE and RRSE, lower values are better, while for CORR, higher values are better.

The effectiveness and generalization of the MTCAN model are stronger than the comparison models. The GRU model works well on the Beijing PM2.5 dataset but less well on the ISO-NE dataset and the Jena climate dataset. The LSTM is more balanced on the three datasets, but it is not as good as the MTCAN model. The ConvLSTM model works well on the ISO-NE dataset but not well enough on the other two datasets. The TCN model, on the other hand, has even worse generalization and has better results only on the ISO-NE dataset and the results on the other two datasets are very poor and not meaningful for comparison.

Table 2. Results summary (in RMSE, RRSE, and CORR) of all methods with three datasets: each row has the results of a specific method in a particular metric.

Methods	Metrics	ISO-NE Dataset	Beijing PM2.5 Dataset	Jane Climate Dataset
		Length = 24	Length = 24	Length = 24
GRU	RMSE	844.65	67.96	3.13
	RRSE	0.3149	0.7263	0.9199
	CORR	0.9497	0.6885	0.9199
LSTM	RMSE	784.50	68.37	2.93
	RRSE	0.2925	0.7281	0.3695
	CORR	0.9570	0.6873	0.9298
ConvLSTM	RMSE	688.05	82.50	4.88
	RRSE	0.2564	0.8803	0.6160
	CORR	0.9670	0.4870	0.7914
TCN	RMSE	726.07	113.84	8.45
	RRSE	0.2693	1.1460	1.0087
	CORR	0.9603	0.0074	0.1766
MTCAN	RMSE	645.60	65.02	2.72
	RRSE	0.2341	0.6949	0.3428
	CORR	0.9773	0.7210	0.9401

Figure 6 shows in detail the specific performance of the four models on the three datasets. From this figure, it can be concluded that the MTCAN model has smoother fluctuations in its curves due to several other models both in terms of generalizability and accuracy. We do not depict the experimental data curves of the TCN model because of its too poor generalizability.

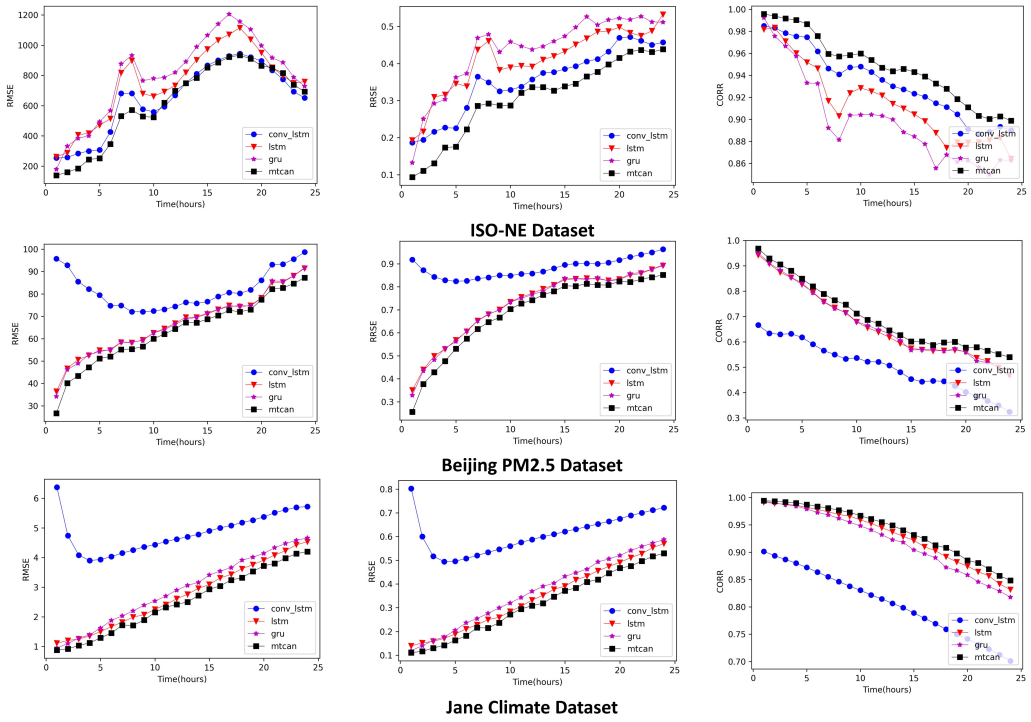

Figure 6. The RMSE, RRSE, and CORR for each lead time from 1 to 24 h vs. different algorithms over the ISO-NE Dataset, Peijing PM2.5 Dataset, and Jane Climate Dataset.

4.4. Ablation Experiments

For complex model structures, we do not know whether the added structure has a beneficial effect on the performance of the model. To better explain the validity of our proposed model, we conducted a careful further study. An ablation experiment was designed to demonstrate the effectiveness of the self-attention layer.

Figure 7 shows the repeat block structure of this model in detail. In this repeat block, we remove the self-attentive network layer. In this test, there is no change in the network except for this point.

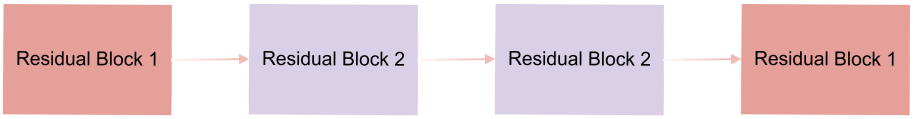

Figure 7. Repeat block structure of model.

Through Table 3, we can conclude that our addition of the self-attentive layer helps the network achieve more accuracy and better generalization, and it reduces the size of the number of model parameters. The number of parameters of the MTCAN model is about 25% that of the model, which greatly improves the model training efficiency. All components of the MTCAN model ensure the effectiveness and generalization of the model as well as reduce the size of the model and improve the learning efficiency.

Table 3. Results summary (in RMSE, RSE, and CORR) of all methods with three datasets: each row has the results of a specific method in a particular metric.

Methods	Metrics	ISO-NE Dataset	Beijing PM2.5 Dataset	Jane Climate Dataset
		Length = 24	Length = 24	Length = 24
Model	RMSE	649.50	65.53	2.76
	RRSE	0.2421	0.6989	0.3484
	CORR	0.9705	0.7159	0.9387
	Params	3,822,700	15,253,228	19,063,404
MTCAN	RMSE	645.60	65.02	2.72
	RRSE	0.2341	0.6949	0.3428
	CORR	0.9773	0.7210	0.9401
	Params	977,004	3,870,444	4,834,924

5. Conclusions

The MTCAN model is proposed and the network structure design is highlighted, which is designed by incorporating a one-dimensional dilated convolutional network as the basic unit, asymmetric residual blocks, and a self-attentive network at the end to enhance the capture of the mapping relationships within the time series to improve the performance of multivariate time-series prediction. The model is validated by three datasets—PM2.5, ISO-NE, and Jane Climate that have been compared with existing models LSTM, GRU, ConvLSTM, and TCN. The prediction results show that the model proposed in this paper has significantly improved calculation efficiency, prediction accuracy, and generalization. This improvement is attributed to the explainability of the self-attention mechanism, which shall be further investigated.

Author Contributions: Conceptualization, R.W. and C.T.; methodology, C.T.; software, R.W.; validation, W.Z. and W.D.; formal analysis, C.T.; investigation, F.Y.; resources, R.W.; data curation, W.Z. and W.D.; writing—original draft preparation, C.T.; writing—review and editing, R.W., W.Z. and W.D.; visualization, C.T.; supervision, R.W. and F.Y.; project administration, F.Y.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.K.; Woo, W.C. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, USA, 7–12 December 2015; Volume 28.
2. Zhang, P.; Zhang, L.; Leung, H.; Wang, J. A deep-learning based precipitation forecasting approach using multiple environmental factors. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017.
3. Celik, A.E.; Karatepe, Y. Evaluating and forecasting banking crises through neural network models: An application for Turkish banking sector. *Expert Syst. Appl.* **2007**, *33*, 809–815. [CrossRef]
4. Heaton, J.B.; Polson, N.G.; Witte, J.H. Deep learning for finance: Deep portfolios. *Appl. Stoch. Model. Bus. Ind.* **2017**, *33*, 3–12. [CrossRef]
5. Yu, R.; Li, Y.; Shahabi, C.; Demiryurek, U.; Liu, Y. Deep learning: A generic approach for extreme condition traffic forecasting. In Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, TX, USA, 27–29 April 2017; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2017.
6. Du, S.; Li, T.; Gong, X.; Yang, Y.; Horng, S.J. Traffic flow forecasting based on hybrid deep learning framework. In Proceedings of the 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), Nanjing, China, 24–26 November 2017.
7. Kruger, A.; Krajewski, W.F.; Niemeier, J.J.; Ceynar, D.L.; Goska, R. Bridge-mounted river stage sensors (BMRSS). *IEEE Access* **2016**, *4*, 8948–8966. [CrossRef]
8. Viton, F.; Elbattah, M.; Guérin, J.L.; Dequen, G. Heatmaps for visual explainability of cnn-based predictions for multivariate time series with application to healthcare. In Proceedings of the 2020 IEEE International Conference on Healthcare Informatics (ICHI), Oldenburg, Germany, 30 November–3 December 2020.
9. Kavaklioglu, K.; Ceylan, H.; Ozturk, H.K.; Canyurt, O.E. Modeling and prediction of Turkey's electricity consumption using artificial neural networks. *Energy Convers. Manag.* **2009**, *50*, 2719–2727. [CrossRef]
10. Gan, Z.; Li, C.; Zhou, J.; Tang, G. Temporal convolutional networks interval prediction model for wind speed forecasting. *Electr. Power Syst. Res.* **2021**, *191*, 106865. [CrossRef]
11. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1724–1734.
12. Jin, X.B.; Gong, W.T.; Kong, J.L.; Bai, Y.T.; Su, T.L. A variational Bayesian deep network with data self-screening layer for massive time-series data forecasting. *Entropy* **2022**, *24*, 335. [CrossRef]
13. Graves, A. Long short-term memory. In *Supervised Sequence Labelling with Recurrent Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 37–45.
14. Zhang, B.; Xiong, D.; Su, J. A GRU-Gated Attention Model for Neural Machine Translation. *arXiv* **2017**, arXiv:1704.08430.
15. Rust, J. Structural estimation of Markov decision processes. *Handb. Econom.* **1994**, *4*, 3081–3143.
16. Lipton, Z.C.; Kale, D.C.; Wetzel, R. Directly Modeling Missing Data in Sequences with RNNs: Improved Classification of Clinical Time Series. *Mach. Learn. Healthc.* **2016**, *56*, 253–270.
17. Mathonsi, T.; van Zyl, T.L. A statistics and deep learning hybrid method for multivariate time series forecasting and mortality modeling. *Forecasting* **2021**, *4*, 1–25. [CrossRef]
18. Bai, S.; Kolter, J.Z.; Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv* **2018**, arXiv:1803.01271.
19. Yu, F.; Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv* **2015**, arXiv:1511.07122.
20. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
21. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
22. Carta, S.; Podda, A.S.; Recupero, D.R.; Saia, R. A local feature engineering strategy to improve network anomaly detection. *Future Internet* **2020**, *12*, 177. [CrossRef]
23. Huang, B.; Zheng, H.; Guo, X.; Yang, Y.; Liu, X. A Novel Model Based on DA-RNN Network and Skip Gated Recurrent Neural Network for Periodic Time Series Forecasting. *Sustainability* **2021**, *14*, 326. [CrossRef]

24. Yule, G.U. VII. On a method of investigating periodicities disturbed series, with special reference to Wolfer's sunspot numbers. *Philos. Trans. R. Soc. Lond. Ser. A Contain. Pap. Math. Phys. Character* **1927**, *226*, 267–298.
25. Akaike, H. Fitting autoregressive models for prediction. *Ann. Inst. Stat. Math.* **1969**, *21*, 243–247. [CrossRef]
26. Walker, G.T. On periodicity in series of related terms. *Proc. R. Soc. Lond. Ser. A Contain. Pap. Math. Phys. Character* **1931**, *131*, 518–532. [CrossRef]
27. Rojas, I.; Valenzuela, O.; Rojas, F.; Guillén, A.; Herrera, L.J.; Pomares, H.; Marquez, L.; Pasadas, M. Soft-computing techniques and ARMA model for time series prediction. *Neurocomputing* **2008**, *71*, 519–537. [CrossRef]
28. Ediger, V.Ş.; Akar, S. ARIMA forecasting of primary energy demand by fuel in Turkey. *Energy Policy* **2007**, *35*, 1701–1708. [CrossRef]
29. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
30. Wan, R.; Mei, S.; Wang, J.; Liu, M.; Yang, F. Multivariate temporal convolutional network: A deep neural network approach for multivariate time series forecasting. *Electronics* **2019**, *8*, 876. [CrossRef]
31. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional sequence to sequence learning. In Proceedings of the International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017.
32. Oord, A.V.D.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; Kavukcuoglu, K. WaveNet: A Generative Model for Raw Audio. *SSW* **2016**, *125*, 2.
33. Chaudhary, N.; Misra, S.; Kalamkar, D.; Heinecke, A.; Georganas, E.; Ziv, B.; Adelman, M.; Kaul, B. Efficient and Generic 1D Dilated Convolution Layer for Deep Learning. *arXiv* **2021**, arXiv:2104.08002.
34. Yu, F.; Koltun, V.; Funkhouser, T. Dilated residual networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
35. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27–28 October 2019.
36. Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G. On the importance of initialization and momentum in deep learning. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013.
37. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.