Journal of Electrical and Computer Engineering Innovations
(JECEI)
Journal homepage: http://www.jecei.sru.ac.ir

Research paper

A Transformer Self-Attention Model for Time Series Forecasting

R. Mohammdi Farsani, E. Pazouki*

Artificial Intelligence Department, Faculty of Computer Engineering, Shahid RajaeeTeacher Training University, Tehran, Iran.

Article Info

Article History:
Received 14 May 2020
Reviewed 07 July 2020
Revised 18 September 2020
Accepted 21 November 2020

Keywords:
Time Series Forecasting (TSF)
Self-attention model
Transformer neural network

*Corresponding Author's Email Address:
ehsan.pazouki@sru.ac.ir

Abstract

Background and Objectives: Many real-world problems are time series forecasting (TSF) problem. Therefore, providing more accurate and flexible forecasting methods have always been a matter of interest to researchers. An important issue in forecasting the time series is the predicated time interval.
Methods: In this paper, a new method is proposed for time series forecasting that can make more accurate predictions at larger intervals than other existing methods. Neural networks are an effective tool for estimating time series due to their nonlinearity and their ability to be used for different time series without specific information of those. A variety of neural networks have been introduced so far, some of which have been used in forecasting time series. Encoder decoder Networks are an example of networks that can be used in time series forcasting. an encoder network encodes the input data based on a particular pattern and then a decoder network decodes the output based on the encoded input to produce the desired output. Since these networks have a better understanding of the context, they provide a better performance. An example of this type of network is transformer. A transformer neural network based on the self-attention is presented that has special capability in forecasting time series problems.
Results: The proposed model has been evaluated through experimental results on two benchmark real-world TSF datasets from different domain. The experimental results states that, in terms of long-term estimation Up to eight times more resistant and in terms of estimation accuracy about 20 percent improvement, compare to other well-known methods, is obtained. Computational complexity has also been significantly reduced.
Conclusion: The proposed tool could perform better or compete with other introduced methods with less computational complexity and longer estimation intervals. It was also found that with better configuration of the network and better adjustment of attention, it is possible to obtain more desirable results in any specific problem.

Introduction

A time series is a sequence of data that is usually collected at consecutive times. The time series is defined by

$$x(t), t = 0,1,2,\ldots \quad (1)$$

where t denotes the elapsed time. Forecasting the time series means estimating the future of the series according to its past (see in [1]). In fact, time series forecasting means estimating the time series value in interval $t{:}T$ based on the past known interval $1{:}t-1$. Three common methods for forecasting time series are:Random models, Support Vector Machine (SVM)

models and Artificial Neural Network models [1]. Autoregressive Integrated Moving Average (ARIMA) model is one of the most popular and widely used random models which is used as a base model in most time series research (see in [2], [3] and [4]). This model assumes that the data are linear and follows a certain probability distribution such as normal. However, this model supports wide variety of time series problems. One of the important tools for solving forecasting time series problem is artificial neural network (ANN). Since ANN solves problem without any conditions or assumption, it is considered as an effective and widely used tool. Some samples of ANN that are used in this area are MultiLayer Perceptron (MLP) [5], Convolutional Neural Network (CNN) [6] Recurrent Neural Network (RNN) [7]. The SVM is another widely used tool to solve the time series forecasting problems [8]. Random modeling is one of the oldest forecasting tools in time series problem. Two common random model for forecasting time series are Autoregressive (AR) and Moving Averaging (MA) [2] and [3]. Also, two new models Autoregressive Moving Average (ARMA) and ARIMA are introduced by combining these two models (see in [2] and [3]). In AR(p) future point of a time series is calculated based on linear combination of p previous value of time series, a constant value and random noise that is shown in (2),

$$y_i = c + \sum_{i=1}^{p} \varphi_i y_{i-1} + \epsilon_i \quad (2)$$

where y_{i-1} are previous value of time series, ϵ_i is random noise, c is a constant value and φ_i are the linear parameters of the model. MA model is proposed with some change AR. In MA future point is calculated based on a linear combination of previous error that is shown in (3),

$$y_t = \mu + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t \quad (3)$$

where ϵ are the errors of time series with some statistical properties, μ is a mean value and φ_i are the linear parameters of the model. ARMA and ARIMA models are proposed by efficient combining of the AR and MA models. Method BJ has proposed an iterative approach for finding best model among ARIMA models without any assumption on time series problem [1].

Random models are used as benchmarks for comparisons in most studies because they are understandable. Random models perform well on linear data whereas most real time series data are nonlinear. Due to the limitations of random models in real data with special features such as missing values, multidimensional and nonlinear, they are less applicable.

Because of their desirable properties, neural networks are more suitable tools than random models. Neural networks are inherently data-driven and self-adaptive. Neural networks have the ability to learn a variety of linear and nonlinear data without the need to have expert knowledge and assumption of the data [9]. The neural network is a model of supervised learning so time series data must be transformed into this model. A sliding window is used to convert time series data into a supervised learning structure. A fixed length window slides over the time series data. The values in the window are considered as observations, which are represented by x. Values of the time series that are intended to be considered as neural network outputs represented by y. As a result, the $y = f(x)$ is modeled using a neural network. A variety of neural networks have been introduced so far, some of which have been used in forecasting time series. One of the examples of ANN which is used in forecasting time series is MLP. TLNN neural network is a MLP ANN which is used for estimating unseen value of time series [10] which is used for estimating air line time series data. Some of the prominent features of MLPs that make them effective for time series estimation are nonlinear modeling, noise resistance, flexibility in the number of inputs and outputs that give flexibility to the forecasting length and the possibility of its development in multidimensional applications. CNN are another set of neural networks that have a variety of applications in recent years. Numerous modeling using CNNs to estimate time series have also been proposed. In these models, CNNs filters learn periodic patterns and estimate time series with them. One-dimensional CNN is usually used in time series forecasting. The observed time series values are given as input to the network and by using a multilayer network unobserved values is estimated in the output [9]. Model WaveNet [11] is one of the CNN used for forecasting time series. In these model, casual convolution is used that means $t+1$ value of times series is uncorrelated to $x_{t+2}, \ldots, x_{t+T}$ values. In this model pooling layer is not used and so the dimensions of the input and the output of the network is equal. This model is so fast because the recurrent connections are not used. In this model, the acceptance domain is increased by increasing the size of the layers and filters. For solving this problem a dilated convolution neural network was proposed. The dilated convolution increases the input domain by ignoring some input steps. The UFCNN is an other sample of CNN which is used for forecasting time series data [12]. The UFCNN is a kind of FCN that is used for semantic image segmentation. The UFCNN use undecimated wavelet transform undecimated convolution for forecasting time series data. In [13] a variety of deep learning based models for time series forcasting in financial applications are presented and compared acoodring to their results. The Recurrent Neural Network (RNN) such as Long Short-

Term Memory (LSTM) is the other tools that is used for forecasting time series [14] and [15]. In this network, the history of the inputs is used by using a recurrent connection. The LSTM give accurate estimation of time series data by using the historical state of the inputs and current values of the inputs simultaneously. Many various of LSTM models were proposed for forecasting time series. In [9] a multilayer neural network is used at the end of the based model of LSTM that estimates the next value of time series data based on the LSTM outputs. Stacked LSTM is another type of LSTM that uses a stack of LSTM to estimate more complex patterns [16]. Bidirectional LSTM is another variation of LSTM that is made to distinct LSTM [17]. One LSTM is used for forward estimating and other is used for backward estimating. In time series forecasting the bidirectional LSTM is used as a transformer from input to coded output. Each input is coded based on the future and the final coded output is given to the common LSTM for estimating as an input [18]. CNN-LSTM is a mixture of CNN neural network and LSTM neural network that the spatial model of the input is extracted with the CNN and temporal model of the input is extracted with LSTM. Using a mix of CNN and LSTM in time series forecasting is varied. A one-dimensional CNN is usually used to obtain spatial information, followed by a RNN to obtain temporal information [19], [20], [15] and [21]. In [15] a complete review of methods based on RNN for time teries torecasting is provided. Due to the desirable performance of neural network-based models in the time series forcasting, in new and serious applications such as modeling severe epidemic and pandemic behaviour of Covid-19 are used [22] and [23]. In [22], the future conditions of novel Coronavirus is predicted by LSTM model. In [23] a LSTM model is used to predict the future mutation rate of Coronavirus virus which was able to provide acceptable results. In [24] a transformer neural network is used for time series forcasting where transformer-based model is used without change.

Encoder and Decoder networks are another approach of neural networks, an encoder network encodes the input data based on a particular pattern and then a decoder network decodes the output based on the encoded input to produce the desired output. Since these networks have a better understanding of the context, they provide a better performance. A wide variety of these types of networks are introduced in time series estimation.]One of the most successful of these models was the LSTM sequence-to-sequence model with attention [25], [26] and [27]. In attention networks, the impact of coded inputs on outputs is controlled and intelligently applied in terms of other knowledge of the problem. Neural networks are an effective tool for estimating time series due to their nonlinearity and their ability to be used for different time series without specific information of those. Of course, challenges still remain, for example to provide good performance over long sequences, especially in RNN, which require further research. In this paper, one of the objectives is improving the performance of neural network-based tools for estimating long sequences.

Proposed Model

Since neural network has suitable features for time series forecasting, in this study, a model based on one of the desirable neural network models for estimating time series is proposed. The base model which is used in this study is the transformer model. In following describes how to apply this model to time series forecasting.

A. Transformer Model

The transformer model [28] was initially proposed for machine translation, but due to its high performance it was quickly incorporated into other areas such as image production [29], audio [30], text summarization [31], and music [32]. The transformer does not use recurrent and convolution, but uses attention in modeling. The transformer uses an encoder-decoder approach. Initially the input is entered into the encoder, after encoding, the output is generated according to the encoded input and the previous outputs in the decoder. The transformer network is used in proposed model as a based model.

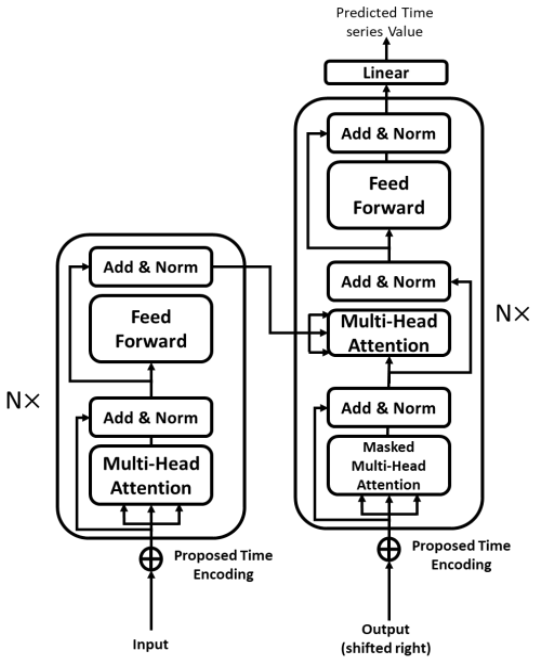

Fig. 1: Architecture of customized transformer neural network.

The encoder section contains a stack of encoders. The number of encoders in a stack is a free parameter, which is usually considered to be 6 layers [28]. In the decoder, a stack of decoders which is equal to the number of

layers in the encoder is used. Each encoding layer has its own parameters i.e. no weight is shared between these layers. Unlike recurrent networks, the transformer has no problem vanishing gradient and can access any point in the past regardless of the distance between words. This feature allows the transformer to find long-term dependencies. Also, unlike recurrent networks, the transformer lacks sequential computation and can run completely in parallel at high speeds. Because of the transformer design for machine translation, it cannot be directly used to forecasting time series. In following, changes made to the transformer to make it suitable for forecasting time series are given.

B. Transformer Model Customization

To adapt the transformer model with any custom time series some modifications to the base model are provided which is shown in Fig. 1. First the embedding layers of the model 's input related to NLP are omitted and the value of time series at any time from $z_t \in R^n$ is given to the model as input. The soft-max layer of the output that is used for classifying is also omitted and the mean square error (MSE) function which is related to regression is used as cost function according to (4).

$$e = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{4}$$

where y_i is the real output, $\hat{y}_i$ is the generated output by the model. The transformer model uses Sin-Cosine method to encode time, which significantly reduces model accuracy in forcasting continuous time series. So, in proposed model, the time is coded by embedding a unique vector to the input data for each time. Vector with length n is used for time intervals with length n, for the value of time i, all its components are zero, except for postion ith, whose value is one. The Fig. 2 illustrate the vectors added in this encoding. In this method each row is concatenated to the vector of the corresponding time.

1	0		0	0
0	1		0	0
.	.		.	.
.	.	...	.	.
.	.		.	.
0	0		1	0
0	0		0	1

Fig. 2: Spatial encoder vectors.

In fact using this method for encoding time, a trainable vector is added to v, q and k vectors, letting the model to determine the value of the vectors by itself in contrary to the cosine and sine encoding method. Equation 6 illustrate the changes of the new encoding model in compare to the cosine-sine encoding method (5).

$$\begin{aligned}
&k_i = x_i W^k, v_i = x_i W^v, q_i = x_i W^q \\
&x_i = x_i^d + pos_i \\
&q_i = (x_i^d + pos_i)W^q = x_i^d \times W^q + pos_i \times W^q \\
&v_i = (x_i^d + pos_i)W^v = x_i^d \times W^v + pos_i \times W^v \\
&k_i = (x_i^d + pos_i)W^k = x_i^d \times W^k + pos_i \times W^k \\
&e_{i,j} = \frac{q_i \times k_j^T}{\sqrt{d}} \\
&a_{i,j} = \frac{exp(e_{i,j})}{\sum_{k=1}^{n} exp(e_{i,k})} \\
&z_i = \sum_{j=1}^{n} a_{i,j} \times v_j
\end{aligned} \tag{5}$$

where x_i^d represents the embedding vector and pos_i represents the cosine-sine vector in i time. Equation (5) is rewritten using new proposed time 's encoding in the form of (6) (only the modified parts of which are re-presented),

$$\begin{aligned}
x_i &= concat(x_i^d, one - hot(i)) \\
q_i &= (x_i^d)W^q + a_i^q \\
k_i &= (x_i^d)W^k + a_i^k \\
v_i &= (x_i^d)W^v + a_i^v
\end{aligned} \tag{6}$$

where x_i^d represents the time series vector, $z_{t=i} \in R^n$ and $one - \mathrm{h}ot(i)$ represents the time encoding vector in time i. The a_i^q, a_i^k and a_i^v are traniable vectors. In other words in (5) x_i^d is a trainable vector resulting from embedding and $pos_i * W$ vector is the time coder. While in (6) x_i^d is the time series value in i which is determined but the a_i^q, a_i^k and a_i^v time coder vectors are trainable vectors.

One of the problems in time encoding using the proposed $one - hot$ method, is the input vector enlargement because of the increase the input steps. It happens in a way that even the size of time encoding vector becomes bigger than the input vector. To solve this problem, several different time encoding vectors are used for different time intervals. For example in a hourly data for one week instead of using a $168(24 \times 7)$ vector, a 24 vector was using to specify the hour and 7 vectors for days of the week. The two vectors are concatenated so the time is coded in a $31(24 + 7)$ vectors. In addition, instead of encoding the time absolutely from the beginning of the series, the typical date and time value is also used.

One of the reported research on transformer model, is replacing Cosine and Sine location encoding with relative vector [33] that increase the performance significantly. The presented solution of the reported research, using the relative vectors, is very similar to encoding time through concatenating a $one - hot$ vector to the input vector. In the relative solutio, time is only coded in the v and k vector and the q vector only depends on the input vector independent of time which no explanation has been provided in the research. The relative solutio is also studied in proposed model.

Equation (7) shows the difference in calculating the q vector in this case. The other calculations are the same.

$$k_i = x_i W^k, v_i = x^d W^v, q_i = x_i^d W^q$$
$$x_i = concat(x_i^d, one - hot(i)) \quad (7)$$
$$q_i = (x_i^d)W^q, v_i = (x_i^d)W^v + a_i^v, k_i = (x_i^d)W^k + a_i^k$$

In this research, in addition to the cost function MSE another cost function is used based on a probabilistic approach. Suppose that $z_t \in R^n$ is the value of z series in t which is a n dimensional vector. if i, is the ith dimension of the time series, the aim is to model the conditional probability distribution of the following equation, $P(Z_{i,t_0:T}|Z_{i,1:t_0-1})$, which is shown the probability of future value of the i dimension. Equation (8) shows the observed interval of the time series and (9) shows unknown interval of the time series.

$$Z_{i,t_0:T} := [Z_{i,t_0}, Z_{i,t_0+1}, \ldots, Z_{i,T}] \quad (8)$$

$$Z_{i,1:t_0-1} := [Z_{i,1}, Z_{i,2}, \ldots, Z_{i,t_0-1}] \quad (9)$$

Equation (8) is realized based on the autoregression which is proposed in RNN architecture [34]. it is assumed for the sake of simplicity in calculations, the probabilities of each time series value based on its previous values are independent of each other. Accordingly, the conditional probability distribution in the form of (10) can be calculated.

$$Q_\theta(Z_{i,t_0:T}|Z_{i,1:t_0}) = \Pi_{t=t_0}^T Q_\theta(Z_{i,t}|Z_{i,1:t-1}) = \Pi_{t=t0}^T l(Z_{i,t}|\theta(x_t)) \quad (10)$$

where x_t shows the output of decoder at time t. In (10) $l(Z_{i,t}|\theta(x_t))$ indicates the probability of occurrence of the time series value $Z_{i,t}$ with respect to the matched probability distribution parameters θ. The distribution function is assumed to be the Gaussian probability distribution. The parameters of the Gaussian probability distribution function are mean and standard deviation $\theta = (\mu, \sigma)$ and the equation of probability distribution function is presented in (11).

$$l_G(\mu, \sigma) = (2\pi\sigma^2)^{-\frac{1}{2}} exp(\frac{(z-\mu)^2}{2\sigma^2}) \quad (11)$$

$$\mu(x_{i,t}) = W_\mu^T x_{i,t} + b_\mu \quad (12)$$

$$\sigma(x_{i,t}) = log(1 + exp(W_\sigma^T x_{i,t} + b_\sigma)) \quad (13)$$

In order to estimate the probability distribution parameters based on the final output of the transformer neural network (12) and (13) are used. Probability distribution parameters proposed in (10) by maximizing probability logarithms is obtained as shown in (14). In other words, the probability distribution parameters are calculated to maximize the probability of joint distribution of the samples in the prediction interval.

$$L = \sum_{i=1}^N \sum_{t=t_0}^T log(l(Z_{i,t}|\theta(x_t))) \quad (14)$$

Therefore, based on equations (10), (11) and (14) final cost function is shown as (15),

$$L = \sum_{i=1}^N \sum_{t=t_0}^T -log(\sqrt{2\pi}) - log(\sigma_{i,t}) - \frac{(z_{i,t}-\mu_{i,t})^2}{2\sigma_{i,t}^2}. \quad (15)$$

Since $log(\sqrt{2\pi})$ value is constant, it can be ignored. Also, by considering the variance constant, the cost function will be converted to the MSE cost function, indicating that the MSE is in fact a special case of the maximum likelihood cost function, where the variance is constant.

In training phase, time series value is entered through the encoder-decoder attention layer from the observation section to the prediction section, and in the prediction or decoder interval, the network is trained by minimizing the $-L$ (see (15) function through the back propagation algorithm.

In testing phase, the network does not have access to the prediction interval. In this case, the network uses the output at any time as input at a later time. Figure 3 shows the model performance in the model training and evaluation process.

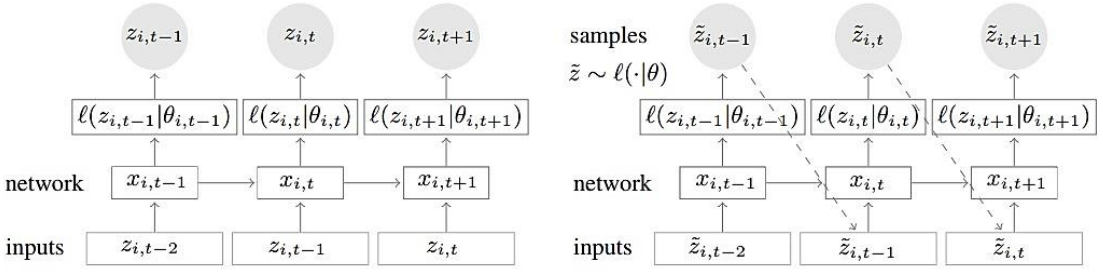

Fig. 3: Function of the model at training phase(left) and at testing phase(right) [35].

According to Fig. 3 the parameters of the distribution functions $l(Z_{i,t}|\theta(x_t))$ are estimated from the model output that is the estimated x_t. Then from the estimated distribution, the point with the highest probability is considered as the output. Because Gaussian distribution considers the probability distribution function, the point has the maximum probability of being the mean point. The model is trained

by maximizing the joint probability of the output, and the model output at each step is used as input in the next step because of the prediction part is unknown.

Results and Discussion

A series of experiments have been performed to validate the proposed model. First, two datasets in different context that are used in the experiments are presented and quantitative validation metrics are introduced. Then the results of a variety of experiments with different purposes are presented. The results are discussed and summarized.

A. Datasets

The electricity consumption dataset in a power grid and traffic data of its application are used to validate the proposed model. The electricity consumption dataset was collected from monitoring of the electricity consumption of 370 households every 15 minutes from 01-01-2011 to 01-01-2015. By averaging every 4 consecutive time steps, the time intervals for both samples is converted from 15 minutes to 1 hour. So there are 24 observations per day. Also the model is trained by data from 01-01-2014 to 01-09-2014 and is validated by data from 01-09-2014 to 01-01-2015 data. In the training 7 consecutive days is used as input and 1 day later as output. In fact, the 168 time steps are considered as inputs and the next 24 steps as the estimation intervals . The traffic dataset was collected from 963 traffic monitoring in the city of San Francisco USA every 10 minutes from 01-01-2008 to 30-03-2009. By averaging every 6 consecutive time steps, the dataset is converted from 10-minute data to hourly data. Similar to the electricity consumption dataset, there are 24 observations per day, the last 7 days are used to train the model as input interval and the next 1 day as output or estimation interval. In fact we have 168 input steps and 24 output steps. In this dataset all available data prior to 15-06-2008 is used to train the model and the rest of the data is used as the validation data.

B. Evaluation Metrics

In this study two different metrics are used to evaluate the proposed model. Normalized Mean Square Error($NMSE$) and Normalized Deviation(ND) are two metrics of network performance evaluation. The ND actually represents the standard $\rho - risk$ metric that is well-known for predicting time series with $\rho = .5$. The equation of $NMSE$ and the equation of ND are given in (16) and (17) respectively.

$$NRMSE = \frac{\sqrt{\frac{1}{N(T-t_0)}\sum_{i,t}(Z_{i,t}-\hat{Z}_{i,t})^2}}{\frac{1}{N(T-t_0)}\sum_{i,t}|Z_{i,t}|} \tag{16}$$

$$ND = \frac{\sum_{i,t}|Z_{i,t}-\hat{Z}_{i,t}|}{\sum_{i,t}|Z_{i,t}|} \tag{17}$$

where N is a dimension of time series for each step, $(T - t_0)$ is the estimation interval, $z_{i,t}$ is the actual value of time series data and $\hat{z}_{i,t}$ is the estimation value of the time series in time t and dimension i. Both metrics used are normalized, meaning that the effect of time series variations on them is diminished.

C. Results

Normalization is used in pre-processing the dataset. The values of the series are subtracted from the mean and divided by the standard deviation. The equation that is used for normalization is shown in (18),

$$\begin{aligned} y &= \frac{x-\mu}{\delta} \\ \mu &= \frac{1}{n}\sum_{i=1}^{N} x_i \\ \delta^2 &= \frac{1}{n-1}\sum_{i=1}^{N}(x-\mu)^2 \end{aligned} \tag{18}$$

where y is the normalized value and μ is the mean and δ^2 is the variance. Also, in some experiments, the data is used that is transferred to the interval between zero and one according to the (19) as preprocessing. In the sliding window method, a dataset with length $T - w + 1$ is created from time series data with length T and window size w. w is the sum of the input and output intervals for each step of estimation.

$$y = \frac{x-min}{max-min}. \tag{19}$$

After normalization, the estimation problem should be converted to the problem with the observer. The step of process is performed with a sliding window that is divided in two section, the first is a observation and known values and the second one is the prediction section and unknown. In training phase two section of sliding window are known and in test phase the prediction section is estimated with the trained models and the actual values of this section is used for calculating evaluation metrics. This window moves forward one step at a time and generates new data. The results are obtained on a transformer with 2 layers, 8 heads, feed-forward layer with 512 neurons and 64 batch sizes after 15,000 training steps. A number of influential parameters of the network are discussed below. The experiments were performed by two transformer modes with the MSE objective function and the probabilistic objective function that are introduced. To validate the results, all the results were obtained from three different experiments and the mean and standard deviation of the different experiments were reported as a result. For comparing the results with other proposed methods the results which are reported in [35] are used. To compare the results, a similar datasets, training and testing procedures which are used in [35] are used. The obtained results from our proposed model, TRMF [36] and DeepAR [35] are shown in Table 1 and Table 2.

Table 1: The result of electricity consumption dataset

Metrics	TRMF	Deep AR	Transformer with MSE objective function	Transformer with Probabilistic objective function
NRMSE↓	1.15	1	1.041±0.052	**0.889±0.029**
ND↓	0.16	**0.07**	0.137±0.006	0.122±0.003

Table 2: The result of traffic dataset

Metrics	TRMF	Deep AR	Transformer with Probabilistic objective function
NRMSE↓	0.43	**0.42**	0.521±0.047
ND↓	0.18	**0.17**	0.24±0.012

As shown in Table 1 and Table 2 the transformer with the probabilistic objective function yields better results than the transformer with MSE objective function on both evaluation metrics. Also, the transformer with probabilistic objective function performs better on $NMSE$ metric than on other models, but on ND metric it yields better than the TRMF model for electricity consumption dataset.

Each time step in the transformer attend to the other steps and each step is coded according to all other steps. By changing the mask vectors, it is possible to determine what other steps to pay attention to each time step. By changing the masks so that each time step is only causally attend on its preceding steps or the number of time steps ahead of itself and also the number of time steps r in its neighborhood, no change in network performance is observed. The results of the experiments shows that changes in the number of layers and attentions did not affect the network performance. One reason for this phenomenon is the difference between common time series with the concepts of natural languages time series. Generating attention vector(Mask vector), considering the context of each step, seems to be a good solution to this problem. In electricity consumption dataset based on the context behind the data, seasonal, weekly or daily steps can be defined. For example, by changing the definition of attention to how the encoder in each time step would only pay attention to the same day time steps and in decoder would only pay attention to same time step of previous day better results were obtained. Based on the results of this experiment that are shown in Table 3 and Table 4 performance has obviously increased with the context based change in attentions. Therefore, it can be concluded that a suitable change in attitudes can increase the efficiency of the transformer.

Another improvement which is introduced is encoding the time of values in time series with their actual values instead of relative values. For example, in the relative encoding modes for encoding time in a daily time sequence with 3 consecutive steps one-hot 1, 2 and 3 vectors were attached to each step as the first to third days.

Table 3: The result of electricity consumption dataset after change in attention

Metrics	TRMF	Deep AR	Transformer with Probabilistic objective function	Transformer with attention change
NRMSE↓	1.15	1	0.889±0.029	**0.734±0.012**
ND↓	0.16	**0.07**	0.122±0.003	0.101±0.0004

Table 4: The result of traffic dataset after change in attention

Metrics	TRMF	Deep AR	Transformer with Probabilistic objective function	Transformer with attention change
NRMSE↓	0.43	**0.42**	0.521±0.047	0.485±0.011
ND↓	0.18	**0.17**	0.24±0.012	0.189±0.0007

Whereas in encoding by the actual value method, if the three consecutive steps are related to Tuesday through Thursday one-hot 4, 5 and 6 vectors were attached to each step. Encoding the location with the actual value did not cause a significant change in the electricity consumption dataset as before, but in the traffic data set the changes are significant which is shown in Table 5.

Table 5: The result of traffic dataset with actual value location encoding

Metrics	TRMF	Deep AR	Transformer with relative time encoding	Transformer with actual time encoding
NRMSE↓	0.43	**0.42**	0.521±0.047	0.46±0.01
ND↓	0.18	0.17	0.24±0.012	**0.147±0.0005**

Another investigated result is the use of normalization preprocessing based on (19) for data transfer between zero and one and the use of sigmoid function on linear transformation based (12) and (13) for estimates of the distribution parameters. The result of the estimation is given in Table 6 and Table 7.

Table 6: The result of traffic dataset with zero-one normalization and Sigmoid Function

Metrics	TRMF	Deep AR	Based model of Transformer	Transformer with new configuration
NRMSE↓	0.43	**0.42**	0.521±0.047	0.433±0.009
ND↓	0.18	0.17	0.24±0.012	**0.134±0.0006**

Table 7: The result of electricity consumption dataset with zero-one normalization and Sigmoid Function

Metrics	TRMF	Deep AR	Based model of Transformer	Transformer with new configuration
NRMSE↓	1.15	1	0.889±0.029	**0.722±0.017**
ND↓	0.16	**0.07**	0.122±0.003	0.93±0.004

One of the important features in time series prediction models is the ability of the model to predict long time. The performance of many models, such as the Deep AR model, is reduced by more than four times as the number of predictive steps is increased [37]. Changing the efficiency of the proposed transformer model with the probabilistic objective function in the RMSE metric when the prediction interval is increment from 24 steps to 48, 96 and 168 in the electricity consumption dataset is shown in Fig. 4.

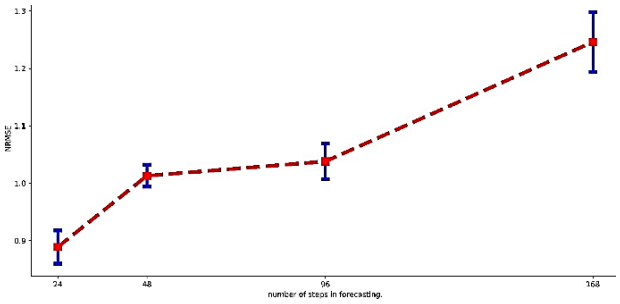

Fig. 4: The effect of increasing the estimation interval(Horizontal Axis) on NRMSE(Vertical Axis) in transformer.

As Fig. 4 shows, the error increases with the prediction interval increment, but the range of the error is from 0.9 to 1.3, while other models such as Deep AR is shown greater error by increasing prediction interval [37]. In other words, Fig. 8 shows that the proposed transformer is capable of long-term prediction.

Due to the lack of sequential computing, the transformer can be run completely in parallel. For this reason, the transformer has a higher speed than the recurrent neural network models. Also, self-attention has fewer computations than recurrent and convolution methods when the sequence length is less than the input dimension. On the other hand it lacks sequential computation. The restricted self-attention is a kind of attention that each word attend to only r neighbor words [28]. The implementation of the heads is done so that the calculations of several different heads are paralleled. Thus increasing the number of heads does not change the time complexity of the transformer. Also as shown in Table 9 as the input sequence increases, the computations increase quadratically. Increasing the input and output interval does not affect the number of network parameters. As the number of layers increases, the number of parameters change linearly. In order to display the increase rates of the number of network parameters in Fig. 10 the number of parameters for different number of layers are provided for the sequence dimensions 963, feed dimensions 512, and dimension of v, q and k vectors 20.

Table 8. Compare the computational complexity for self-attention, where **n** is the length of input, **d** is the dimension of input, **k** is the kernel size on convolution and **r** is the number of neighborhood for self-attention [28]

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2.d)$	$O(1)$	$O(1)$
Recurrent	$O(n.d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k.n.d^2)$	$O(1)$	$O(\log_k(n))$
Recurrent	$O(r.n.d)$	$O(1)$	$O(n/r)$

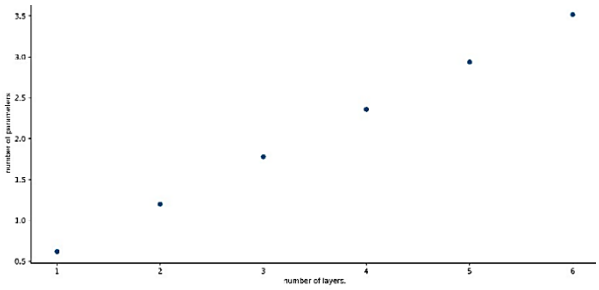

Fig. 5: Change the number of the network parameters (Vertical Axis) by changing the number of layers(Horizontal Axis).

According to Fig. 5, the network parameters vary linearly between 2 million to 12 million for 1 to 6 layers. The runtime of recurrent models, such as Deep AR, on the common dataset is approximately 7 hours [35], while with the transformer model achieving better results in $NMSE$ metric and competing results in ND metric in about 40 minutes.

The transformer model has the parameters of the number of layers, the dimensions of the feed-forward layer, the number of heads, and the dimensions of the vectors k, q and v. The dimensions of the feed-forward layer are typically much larger than the input dimension of the problem and are about twice that. In estimations As the number of layers increased, there was no change in the efficiency of the converter. The transformer uses a sine and cosine function to encode time. This method is less efficient in time series data, so this function was replaced by the time encoding method by adding a spatial vector. In training phase of the model Adam optimizer is used [38]. The rate of training on the network is very impressive and it is very difficult to determine. By examining different training rates and using different cut rates, finally the Noam scheduling training rate introduced in the transformer was used.

Also, to train the model, Google's research collaboration service with the K80 graphics computing unit was used.

Conclusion

Many issues in different domains can be modeled in the time series forecasting. Therefore, providing a desirable tool for estimating time series can have significant applications. In this paper, an effective tool for estimating time series is introduced. The objectives of this paper have been to provide a more accurate and long interval estimation tool. Neural networks are effective and useful tools for modeling various problems such as estimation time series problem. Introducing the transformer network as an effective tool in natural language processing has made a significant improvement in performance compared to previous methods in recent years. Since contextual data is a complex example of time series data, the transformer network seemed to be a useful tool in forecasting time series. In this paper, the modified transformer model use in a variety of time series with changes in the structure of this neural network. The natural language processing layers were removed, changes were made to the encoding temporal layer, and a new regression-based objective function was introduced. Experiments were performed using a modified transformer network on a variety of datasets and it was found that the tool could perform better or compete with other introduced methods with less computational complexity and longer estimation intervals. It was also found that with better configuration of the network and better adjustment of attention, it is possible to obtain more desirable results in any specific problem. Therefore, the proposed model has the potential to provide more desirable results with further study and it is a good base model for presenting a set of effective methods in time series forecasting.

Author Contributions

Reza Mohammadi Farsani designed and simulated and carried out the data analysis, and Ehsan Pazouki collected the data and interpreted the results and wrote the manuscript.

Acknowledgment

The authors gratefully thank the anonymous reviewers and the editor of JECEI for their useful comments and suggestions.

Conflict of Interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy have been completely witnessed by the authors.

References

[1] R. Adhikari, R.K. Agrawal, "An introductory study on time series modeling and forecasting," ArXiv Prepr. ArXiv13026613, 2013.

[2] G. E. Box, G. M. Jenkins, G. Reinsel, "Time series analysis: forecasting and control Holden-day San Francisco," BoxTime Ser. Anal. Forecast. Control Holden Day1970, 1970.

[3] K.W. Hipel, A.I. McLeod, Time series modelling of water resources and environmental systems, 45, Elsevier, 1994.

[4] G.P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," Neurocomputing, 50: 159–175, 2003.

[5] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[6] A. Krizhevsky, I. Sutskever, G.E. Hinton, "Imagenet classification with deep convolutional neural networks," Comm. ACM, 60(6): 1097–1105, 2012.

[7] F.A. Gers, J. Schmidhuber, F. Cummins, "Learning to forget: continual prediction with LSTM," Neural Comput., 12(10): 2451–2471, 2000.

[8] L.-J. Cao, F.E.H. Tay, "Support vector machine with adaptive parameters in financial time series forecasting," IEEE Trans. Neural Netw., 14: 1506–1518, 2003.

[9] J. Brownlee, Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery, 2018.

[10] J. Faraway, C. Chatfield, "Time series forecasting with neural networks: a comparative study using the airline data," J. R. Stat. Soc. Ser. C Appl. Stat., 47: 231–250, 1998.

[11] A. Borovykh, S. Bohte, C.W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," ArXiv Prepr. ArXiv170304691, 2017.

[12] R. Mittelman, "Time-series modeling with undecimated fully convolutional neural networks," ArXiv Prepr. ArXiv150800317, 2015.

[13] O.B. Sezer, M.U. Gudelek, A.M. Ozbayoglu, "Financial time series forecasting with deep learning : A systematic literature review: 2005–2019," Appl. Soft Comput., 90: 106181, 2020.

[14] S. Hochreiter, J. Schmidhuber, "Long short-term memory," Neural Comput., 9: 1735–1780, 1997.

[15] H. Hewamalage, C. Bergmeir, K. Bandara, "Recurrent Neural networks for time series forecasting: current status and future directions," Int. J. Forecast., 37(1): 388–427, 2021.

[16] Z. Cui, R. Ke, Y. Wang, "Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction," ArXiv Prepr. ArXiv180102143, 2018.

[17] M. Schuster, K.K. Paliwal, "Bidirectional recurrent neural networks," IEEE Trans. Signal Process., 45: 2673–2681, 1997.

[18] Z. Cui, R. Ke, Y. Wang, "Deep stacked bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction," presented at the 6th International Workshop on Urban Computing (UrbComp 2017), Nova Scotia, Canada, 2016.

[19] H. Yao, X. Tang, H. Wei, G. Zheng, Y. Yu, Z. Li, "Modeling spatial-temporal dynamics for traffic prediction," ArXiv Prepr. ArXiv180301254, 2018.

[20] J. Ke, H. Zheng, H. Yang, X. M. Chen, "Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach," Transp. Res. Part C Emerg. Technol., 85: 591–608, 2017.

[21] Y. Liu, H. Zheng, X. Feng, Z. Chen, "Short-term traffic flow prediction with Conv-LSTM," presented at International Conference on Wireless Communications and Signal Processing, Nanjing, China, 2017.

[22] S. Shastri, K. Singh, S. Kumar, P. Kour, V. Mansotra, "Time series forecasting of Covid-19 using deep learning models: India-USA comparative case study," Chaos Solitons Fractals, 140: 110227, 2020.

[23] R.K. Pathan, M. Biswas, M.U. Khandaker, "Time series prediction of COVID-19 by mutation rate analysis using recurrent neural network-based LSTM model," Chaos Solitons Fractals, 138: 1-7, 2020,

[24] N. Wu, B. Green, X. Ben, S. O'Banion, "Deep transformer models for time series forecasting: the influenza prevalence case," ArXiv200108317 Cs Stat, 2020.

[25] M.T. Luong, H. Pham, C.D. Manning, "Effective approaches to attention-based neural machine translation," ArXiv Prepr. ArXiv150804025, 2015.

[26] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," ArXiv Prepr. ArXiv170402971, 2017.

[27] Y. Liang, S. Ke, J. Zhang, X. Yi, Y. Zheng, "GeoMAN: multi-level attention networks for geo-sensory time series prediction," in Proc. International Joint Conference on Artificial Intelligence: 3428-3434, 2018.

[28] N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, "Attention is all you need," ArXiv Prepr. ArXiv1706.03762, 2017.

[29] N. Parmar, A. Vaswani, J. Uszkoreit, Ł. Kaiser, N. Shazeer, A. Ku , D. Tran, "Image transformer," ArXiv Prepr. ArXiv180205751, 2018.

[30] D. Povey, H. Hadian, P. Ghahremani, K. Li, S. Khudanpur, "A time-restricted self-attention layer for asr," presented at IEEE International Conference on Acoustics, Speech and Signal Processing, Calgary, AB, Canada, 2018.

[31] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, N. Shazeer, "Generating wikipedia by summarizing long sequences," ArXiv Prepr. ArXiv180110198, 2018.

[32] C.Z.A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A.M. Dai, M. D. Hoffman, M. Dinculescu, D. Eck, "Music transformer: Generating music with long-term structure," ArXiv Prepr. ArXiv180904281, 2018.

[33] P. Shaw, J. Uszkoreit, A. Vaswani, "Self-attention with relative position representations," ArXiv Prepr. ArXiv180302155, 2018.

[34] A. Graves, "Generating sequences with recurrent neural networks," ArXiv Prepr. ArXiv13080850, 2013.

[35] D. Salinas, V. Flunkert, J. Gasthaus, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," ArXiv Prepr. ArXiv170404110, 2017.

[36] H.-F. Yu, N. Rao, I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in Advances in neural information processing systems, 29: 847–855, 2016.

[37] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y. X. Wang, X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," ArXiv Prepr. ArXiv190700235, 2019.

[38] D.P. Kingma, J. Ba, "Adam: A method for stochastic optimization," ArXiv Prepr. ArXiv14126980, 2014.

Biographies

Reza Mohammadi Farsani is a graduate of Shahid Rajaee Teacher Training University in the field of computer engineering majoring in artificial intelligence at the master's level. (e-mail: r.mohammadifarsani@gmail.com)

Ehsan Pazouki is an Assistant Professor in the school of Computer Engineering at Shahid Rajaee Teacher Training University where he has been a faculty member since 2016. Ehsan completed his Ph.D. and M.S. at Amirkabir University. His research interests lie in the area of wide area surveillance, ranging from theory to design to implementation. He has collaborated actively with researchers in several other disciplines of computer science, particularly Cognitive Science. Ehsan has experience in various industries related to his specialization for more than 10 years. For additional information see https://www.sru.ac.ir/en/school-of-computer/ehsan-pazouki/

How to cite this paper:
R. Mohammdi Farsani, E. Pazouki, "A transformer self-attention model for time series forecasting," J. Electr. Comput. Eng. Innovations, 9(1): 1-10, 2021.

DOI: 10.22061/JECEI.2020.7426.391

URL: http://jecei.sru.ac.ir/article_1477.html