DSANet: Dual Self-Attention Network for Multivariate Time Series Forecasting

Siteng Huang
Westlake University
huangsiteng@westlake.edu.cn

Donglin Wang*
Westlake University
wangdonglin@westlake.edu.cn

Xuehan Wu
Huawei Technologies Co., Ltd.
wuxuehan2@huawei.com

Ao Tang
WeCar Technology Co., Ltd.
ao.tang@weicheche.cn

ABSTRACT

Multivariate time series forecasting has attracted wide attention in areas, such as system, traffic, and finance. The difficulty of the task lies in that traditional methods fail to capture complicated non-linear dependencies between time steps and between multiple time series. Recently, recurrent neural network and attention mechanism have been used to model periodic temporal patterns across multiple time steps. However, these models fit not well for time series with dynamic-period patterns or nonperiodic patterns. In this paper, we propose a dual self-attention network (DSANet) for highly efficient multivariate time series forecasting, especially for dynamic-period or nonperiodic series. DSANet completely dispenses with recurrence and utilizes two parallel convolutional components, called global temporal convolution and local temporal convolution, to capture complex mixtures of global and local temporal patterns. Moreover, DSANet employs a self-attention module to model dependencies between multiple series. To further improve the robustness, DSANet also integrates a traditional autoregressive linear model in parallel to the non-linear neural network. Experiments on real-world multivariate time series data show that the proposed model is effective and outperforms baselines.

CCS CONCEPTS

• **Mathematics of computing → Time series analysis**;

KEYWORDS

Time Series Forecasting, Deep Learning, Self-Attention

ACM Reference Format:
Siteng Huang, Donglin Wang, Xuehan Wu, and Ao Tang. 2019. DSANet: Dual Self-Attention Network for Multivariate Time Series Forecasting. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3357384.3358132

*Corresponding author.

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3358132

1 INTRODUCTION

As multivariate time series are ubiquitous in our daily life, research on multivariate time series forecasting has been carried out in many areas, such as sensor networks [11], road occupancy rates forecasting [10], and financial market prediction [16]. However, complex and non-linear dependencies exist not only between time steps but also in a variety of variables. Furthermore, the dependencies may change dynamically with time, which significantly increases the difficulty of analysis. Therefore, a major challenge of multivariate time series forecasting is how to capture dynamic dependencies between time steps and multiple variables.

Advanced statistical methods have been proposed for time series forecasting, such as vector autoregression (VAR) [4] and Gaussian process (GP) [11]. However, they usually assume certain distribution or function form of time series, which makes them unable to capture complicated underlying non-linear relationships and reflect reality. In addition, most of them ignore the dependencies between variables when addressing multivariate time series, which reduces the accuracy of forecasting. Recently, deep neural networks have drawn great attention in related domains due to their flexibility in capturing nonlinearity. In particular, recurrent neural network (RNN) [12] has been considered as the default starting point for sequence modeling. However, traditional RNNs have difficulty in capturing long-range dependencies due to gradient vanishing [3]. As its variants, long short-term memory (LSTM) [7] and gated recurrent unit (GRU) [5], have overcome the limitation. Attention mechanism [2] also helps RNN to model temporal patterns, which allows modeling on dependencies of the input and output by focusing on the selective parts of the input sequence. Models based on LSTM or GRU with attention mechanism have been proposed for time series forecasting and show good performances in exploiting long-term dependencies and handling non-linear dynamics [10, 13]. However, due to the unsatisfactory performance, the structure might not be suitable for those data with dynamic-period patterns or nonperiodic patterns, which is common in a complex environment.

To enable accurate and robust forecasting for multivariate time series, we propose a dual self-attention network (DSANet) for highly efficient multivariate time series forecasting without exogenous information. In DSANet, we first feed each of the univariate time series independently into two parallel convolutional components, called global temporal convolution and local temporal convolution, to model complex mixtures of global and local temporal patterns. Next, the learned time series representations from each convolutional component are fed into an individual self-attention module,

with the aim of learning the dependencies among different series. To further improve the robustness, an autoregressive linear model is integrated in parallel to the non-linear attention network of DSANet. To the best of our knowledge, this is the first work to apply self-attention mechanism in time series forecasting with the help of well-designed dual branches architecture. Experiments on a real-world time series data set demonstrate the accuracy and robustness of the proposed method.

2 RELATED WORK

We first consider statistical linear methods for multivariate time series. Here, the vector autoregression (VAR) [4] model is widely considered as a baseline method, which generalizes the univariate autoregressive (AR) model by allowing for more than one evolving variable. To model non-linear relationships, some variants of the autoregressive model are used, such as LRidge [8], LSVR [14] and Gaussian process (GP) [11]. However, they assume certain distribution or function form of time series and fail to capture different forms of nonlinearity.

Due to the ability to flexibly model various non-linear relationships, neural networks are often applied to enable non-linear forecasting models. For example, recurrent neural network models using LSTM or GRU are often used to provide non-linear time series forecasting. To predict more accurately, complex structures such as recurrent-skip layer (LSTNet-S), temporal attention layer (LSTNet-A) [10], and a novel temporal pattern attention mechanism (TPA) [13] have been proposed. However, when working on data with dynamic-period patterns or nonperiodic patterns, their performance drops significantly.

3 PRELIMINARIES

A *time series* $X^{(i)} = \langle x_1^{(i)}, x_2^{(i)}, \ldots, x_T^{(i)} \rangle$ is a fully observed time-ordered sequence of measurements, where measurement $x_T^{(i)}$ is recorded at time stamp T. Usually, the time interval between two consecutive measurements is constant. A *multivariate time series* is denoted as $X = \langle X^{(1)}, X^{(2)}, \ldots, X^{(D)} \rangle$, where time series in X are correlated with each other, and measurements $X_T \in \mathbb{R}^D$ are recorded at time stamp T.

Problem Statement: Given a set of multivariate time series $X = \langle X^{(1)}, X^{(2)}, \ldots, X^{(D)} \rangle$, where D is the number of univariate time series and $X^{(D)} \in \mathbb{R}^T$ with T being the length of the input window size, we aim at predicting in a rolling forecasting fashion. That being said, we predict X_{T+h} based on the known $\langle X_1, X_2, \ldots, X_T \rangle$, where h is the desirable horizon ahead of the current time stamp. Likewise, we predict the future value of X_{T+h+k} based on $\langle X_{1+k}, X_{2+k}, \cdots, X_{T+k} \rangle$, $k \in \mathbb{R}^+$, with an assumption that the information within the window is sufficient for prediction and the window size is fixed. We hence formulate that for the the forecasting target $X_{T+h} \in \mathbb{R}^D$, the input matrix at time stamp T is $X = \langle X^{(1)}, X^{(2)}, \ldots, X^{(D)} \rangle \in \mathbb{R}^{D \times T}$.

4 METHODOLOGY

Figure 1 presents an overview of our proposed DSANet. DSANet utilizes two convolutional structures, namely global temporal convolution and local temporal convolution, to embed each univariate series in X into two representation vectors with temporal information of different scale. Each vector forms a matrix and then enter an elaborate self-attention module to capture the dependencies between multiple series. In the end, the model generates the final prediction by summing up the output of both self-attentional network and the AR component. The details of the building blocks are introduced in the following paragraphs.

Global Temporal Convolution: Deep learning methods in previous work mainly use RNNs to capture temporal patterns. However, due to the inherently sequential nature, it is difficult for RNNs to model long sequences and compute in parallel, which ultimately damages the computing speed and forecasting effect. As convolutional structure has demonstrated its power in capturing features as well as parallel computing, we use it together with multiple $T \times 1$ filters, called global temporal convolution, to extract time-invariant patterns of all time steps for univariate time series.

Each filter of global temporal convolution module sweeps through the input matrix X and produces a vector with size of $D \times 1$, where the activation function is the ReLU function. Merged by the vectors, The convolutional structure finally obtains an output matrix H^G of size $D \times n_G$, where n_G is the number of filters in global temporal convolution. Note that each row of the matrix can be considered as a learned representation of a univariate series.

Local Temporal Convolution: Considering that time steps with a shorter relative distance have a larger impact on each other, DSANet also utilizes a convolutional structure in parallel to global temporal convolution, which is called local temporal convolution. While global temporal convolution captures long-term dependencies between time steps, local temporal convolution focuses on modeling local temporal patterns, which can be more helpful for forecasting.

Different from global temporal convolution, the length of filters used in local temporal convolution is l, where $l < T$ is a hyper-parameter. The k-th filter of local temporal convolution slides along the time dimension and produces a matrix M_k^L. In order to map local temporal relations in each univariate time series to a vector representation, DSANet uses a 1-D max-pooling layer over each column of the matrix M_k^L to capture the most representative features. Thus, we obtain an output matrix H^L of size $D \times n_L$, where n_L is the number of filters in local temporal convolution.

Self-Attention Module: Due to the strong feature-extraction capability of self-attentional networks, we apply a self-attention module inspired by the Transformer [15] to capture the dependencies between different series. For each learned representation of a univariate series, the self-attention module learns its relationship with other learned representations including itself. As shown in Figure 1, the self-attention module is composed of a stack of N identical layers, and each layer has two sub-layers: a self-attention layer and a position-wise feed-forward layer.

In general, an attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and the output are all vectors. The output is computed as a weighted sum of the values, where the weight for each position is computed as the inner product between the query and keys at every other position in time series. In the self-attention module following the *global temporal convolution*, a set of queries, keys, and values are packed together into matrices Q^G, K^G, and V^G, obtained by applying projections to the input H^G.

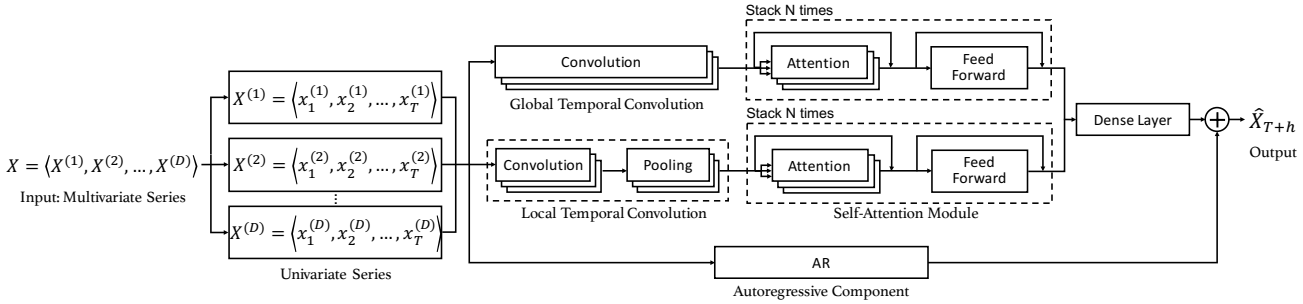

Figure 1: Dual Self-Attention Network (DSANet)

Mathematically, the scaled dot product self-attention computation can be expressed as

$$Z^G = \text{softmax}\left(\frac{Q^G(K^G)^T}{\sqrt{d_k}}\right)V^G, \tag{1}$$

where d_k is the dimension of keys. A multi-head attention is utilized to allow the model to jointly deal with information from different representation subspaces at different position. The resulting weighted representations are concatenated and linearly projected to obtain the final representation Z_O^G.

The position-wise feed-forward layer consists of two linear transformations with a ReLU activation in between, which can be expressed as

$$F^G = \text{ReLU}(Z_O^G W_1 + b_1)W_2 + b_2. \tag{2}$$

While the linear transformation is the same across different positions, they use different parameters. Followed by layer normalization [1], residual connections around each of the sub-layers make training easier and improve generalization.

We have a similar procedure for the self-attention module following the *local temporal convolution*, where we input H^L into the module and finally get the output F^L.

Autoregressive Component: Due to the nonlinearity of both convolutional and self-attention components, the scale of neural network output is not sensitive to that of input. To address the drawback, we consider the final prediction of DSANet as a mixture of a linear component and a non-linear component. Apart from the non-linear component introduced above, the classical AR model [6] is taken as the linear component. The forecasting of the AR component is expressed as $\hat{X}_{T+h}^L \in \mathbb{R}^D$.

Generation of Prediction: In the forecasting stage, we first use a dense layer to combine the outputs of two self-attention modules and get the self-attention based prediction $\hat{X}_{T+h}^D \in \mathbb{R}^D$. The final prediction of DSANet $\hat{X}_{T+h}$ is then obtained by summing the self-attention based prediction $\hat{X}_{T+h}^D$ and the AR prediction $\hat{X}_{T+h}^L$.

5 EXPERIMENTS

Data Sets: We use a large time series data set provided by a gas station service company. The data set contains the daily revenue of five gas stations ranging from Dec.1, 2015 to Dec.1, 2018. The stations are geographically close, which means a complex mix of revenue promotion and mutual exclusion exists between them. Thus we consider the five time series of each gas station as a multivariate time series. Data visualization analysis is performed to ensure that the data set does not contain distinct repetitive patterns.

In our experiments, the data set is chronologically split into training (60%), validation (20%) and test (20%) sets. In each set, we further segment the data into multiple cases using sliding windows, which means in each segment, we use a multivariate time series of T length as the *input data* to the models and use the measurements of the time stamp $T + h$ as the *ground truth data*.

Comparison Methods: In our comparative evaluations, we consider 8 baselines: *VAR*, *LRidge*, *LSVR*, *GP*, *GRU*, *LSTNet-S*, *LSTNet-A*, *TPA*. All the methods are covered in Section 2.

Experimental Settings: All neural network models are optimized by performing mini-batch stochastic gradient descent (SGD) with the Adam optimizer [9], and the loss is calculated by the mean square error (MSE). We conduct a grid search over all tunable hyperparameters on the validation set for each method. Specifically, for LRidge and LSVR, the regularization coefficient λ is chosen from $\{2^{-10}, 2^{-8}, \cdots, 2^8, 2^{10}\}$. For GP, the RBF kernel bandwidth σ and the noise level α are chosen from $\{2^{-10}, 2^{-8}, \cdots, 2^8, 2^{10}\}$. For all neural network models, the hidden dimension size of recurrent layers and convolutional layers are chosen from {32, 50, 100}. For LSTNet-S, we conduct a grid search over {20, 50, 100} for recurrent-skip layers. For DSANet, the length of filters used in local temporal convolution is chosen from {3, 5, 7}. We perform dropout and the rate is set as 0.1.

Implementation Details: All methods are implemented in Python 3.6, where the deep learning methods are implemented using PyTorch 1.0. A computer with Intel i7-8700 CPU, GTX1060 GPU, 6 cores, 32 GB RAM is used to conduct all experiments.

Main Results: To measure the effectiveness of various methods for multivariate time series forecasting, we use root relative squared error (RRSE), mean absolute error (MAE) and empirical correlation coefficient (CORR) as evaluation metrics. A lower value is better for RRSE and MAE while a higher value is better for CORR. We set the problem parameter $window = \{32, 64, 128\}$ and $horizon = \{3, 6, 12, 24\}$, respectively, which means the window length is set from 32 to 128 days and the horizon is set from 3 to 24 days over the dataset. Due to space limitation, we report on results only based on RRSE and MAE with $window = 32$. More experimental results and code are available online[1].

Table 1 summarizes the evaluation results of all the methods on the test set. Each row in Table 1 compares the results of all methods

[1] https://github.com/bighuang624/DSANet

Table 1: Evaluation Results of Multivariate Time Series Forecasting

		Methods								
window-horizon	Metrics	VAR	LRidge	LSVR	GP	GRU	LSTNet-S	LSTNet-A	TPA	DSANet
32-3	RRSE	0.9401	0.8114	0.8934	1.0564	0.8297	0.8222	0.8120	0.8441	**0.7817**
	MAE	0.4914	0.4302	0.4687	0.5676	0.4311	0.4214	0.4220	0.4348	**0.4074**
32-6	RRSE	0.9170	0.8094	0.9144	1.0677	0.8524	0.8544	0.8718	0.8482	**0.7713**
	MAE	0.4743	0.4374	0.4778	0.5616	0.4380	0.4371	0.4465	0.4336	**0.4102**
32-12	RRSE	0.9335	0.9132	0.9600	1.0878	0.8938	0.8753	0.9033	0.8887	**0.8297**
	MAE	0.4746	0.4619	0.4956	0.5580	0.4536	0.4524	0.4529	0.4487	**0.4367**
32-24	RRSE	1.0188	0.9789	1.0178	1.1280	0.9457	0.9941	0.9814	0.9310	**0.9277**
	MAE	0.4988	0.4811	0.5174	0.5611	0.4807	0.4916	0.4921	0.4499	**0.4422**

in a particular metric with a specific window-horizon pair, and each column shows the results of a specific method in all cases. Boldface indicates the best result of each row in a particular metric.

From Table 1, a common phenomenon is that when the horizon increases, both RRSE and MAE of the same method increase on the whole, which shows that the larger the horizon, the harder the forecasting task. Note that GRU, LSTNet-S, LSTNet-A, TPA and DSANet often achieve a better performance in comparison to others, which shows that due to the ability to learn complicated non-linear dependencies between time steps and between multiple time series, deep learning methods can solve complex forecasting tasks better than traditional methods. However, it is observed that compared to other methods, DSANet achieves better results in all cases, indicating that taking advantage of the well-designed architecture, DSANet is more robust to deal with multivariate time series with dynamic-period patterns or nonperiodic patterns.

Ablation Study: To justify the efficiency of our architecture design, a careful ablation study is conducted. Specifically, we remove each of the global temporal convolution branch, the local temporal convolution branch, and the AR component one at a time in our DSANet model, and each new model is named DSAwoGlobal, DSAwoLocal, and DSAwoAR. The test results measured using RRSE and MAE with $window = 32$ are shown in Figure 2, from which several observations are worth highlighting: (1) The best result on each window-horizon pair is obtained by complete DSANet, showing all components have contributed to the effectiveness and robustness of the whole model; (2) The performance of DSAwoAR significantly drops, showing that the AR component plays a crucial role. The reason is that AR is generally robust to the scale changing in data according to [10]; (3) DSAwoGlobal and DSAwoLocal also suffer from performance loss but less than removing the AR component. This is because features learned by the two branches coincide. In other words, when one branch is removed, some of the lost features can be obtained from the other branch.

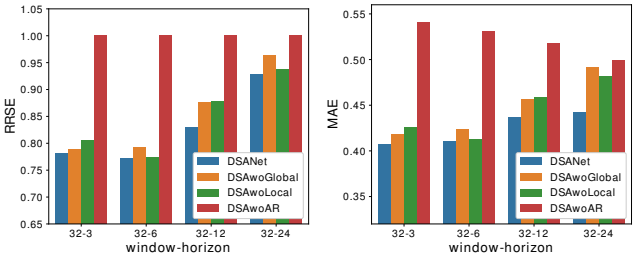

Figure 2: Ablation Test Results of DSANet

6 CONCLUSION

We present a novel deep learning framework, dual self-attention network (DSANet), for the task of multivariate time series forecasting, especially for those data with dynamic-period or nonperiodic patterns. Experiments on a large real-world dataset show promising results.

REFERENCES

[1] Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *CoRR* abs/1607.06450 (2016).
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.
[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5, 2 (1994), 157–166.
[4] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM TIST* 2 (2011), 27:1–27:27.
[5] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.
[6] Lanette Moreau Harrison, W. D. Penny, and Karl J. Friston. 2003. Multivariate autoregressive modeling of fMRI time series. *NeuroImage* 19 (2003), 1477–1491.
[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[8] Arthur E. Hoerl and Robert W. Kennard. 2000. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* 42 (2000), 80–86.
[9] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[10] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. In *SIGIR*.
[11] S. J. Roberts, Matt Osborne, Mark Ebden, Steve Reece, Neal Gibson, and Suzanne Aigrain. 2013. Gaussian processes for time-series modelling. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences* 371 1984 (2013), 20110550.
[12] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature* 323, 6088 (1986), 533.
[13] Shun-Yao Shih, Fan-Keng Sun, and Hung yi Lee. 2018. Temporal Pattern Attention for Multivariate Time Series Forecasting. *CoRR* abs/1809.04206 (2018).
[14] Vladimir Vapnik, Steven E. Golowich, and Alexander J. Smola. 1996. Support Vector Method for Function Approximation, Regression Estimation and Signal Processing. In *NIPS*.
[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.
[16] Yue Wu, José Miguel Hernández-Lobato, and Zoubin Ghahramani. 2013. Dynamic Covariance Models for Multivariate Financial Time Series. In *ICML*.