

MODELING VARIABLE SPACE WITH RESIDUAL TENSOR NETWORKS FOR MULTIVARIATE TIME SERIES

Anonymous authors

Paper under double-blind review

ABSTRACT

Multivariate time series involve a series of valuable applications in the real world, and the basic premise of which is that multiple variables are interdependent. However, the relationship between variables in the latent space is dynamic and complex, and as the time window increases, the size of the space also increases exponentially. For fully exploiting the dependencies in the variable space, we propose *Modeling Variable Space with Residual Tensor Networks* (MVSRTN) for multivariate time series. In this framework, we derive the mathematical representation of the variable space, and then use a tensor network based on the idea of low-rank approximation to model the variable space. The tensor components are shared to ensure the translation invariance of the network. In order to improve the ability to model long-term sequences, we propose an N-order residual connection approach and couple it to the space-approximated tensor network. Moreover, the series-variable encoder is designed to improve the quality of the variable space, and we use the skip-connection layer to achieve the dissemination of information such as scale. Experimental results verify the effectiveness of our proposed method on four multivariate time series forecasting benchmark datasets.

1 INTRODUCTION

Multivariate time series are ubiquitous in the real world, such as energy collection and consumption (Rangapuram et al., 2018), traffic occupancy (Lai et al., 2018), financial market trends (Shih et al., 2019), and aerospace data (Zhao et al., 2020). They mainly consist of sensor data sampled on multiple variables at different timestamps. In order to accurately forecast future time series, it is necessary to learn the laws and the dynamic interaction information between variables that exist in the series history (Wu et al., 2020; Xu et al., 2020; Cheng et al., 2020).

For illustrating the dependencies in the multivariate time series, we focus on examples through case study. In Figure 1(a), a certain variable itself has periodic laws, and the dependency relationship in this respect can be well captured by the short-term and long-term model of the modeling sequence. As shown in Figure 1(b), there are dynamic dependency relations between the variables. The “arrow” indicates the trend of data changes. In 2011, the “Switzerland” variable changed significantly earlier than the other in 2013, the trends of the three variables occurred almost simultaneously.

In literature, there have been some approaches to capture interactions in series history from different perspectives. research utilizing traditional deep learning components such as CNNs, RNNs, and attention mechanisms are more inclined to learn long-term and short-term sequence states, and fail to make full use of the entanglement between variables in the latent space (Lai et al., 2018; Shih et al., 2019; Cheng et al., 2020). New mathematical tools such as graph neural networks and tensor calculation methods have inspired the research on the spatial Relations in multivariate time series. (Wu et al., 2020) uses graph neural network to extract one-way relationships between variable pairs in the end-to-end framework. The TLASM (Xu et al., 2020) adopts tensor calculation to model the trend in the multivariate time series, and combines the core tensor components into the LSTM.

Nevertheless, most of the above-mentioned schemes for explicitly modeling spatio-temporal dependencies tend to build low-dimensional variable spaces, but this is still incomplete for the variable spaces shown in Figure 1(c) of. In other words, in the high-dimensional variable space evolved over time, the ideal state for multivariate time series is that the model can capture the effective relationship between variables as fully as possible. To address the above challenges, we first define

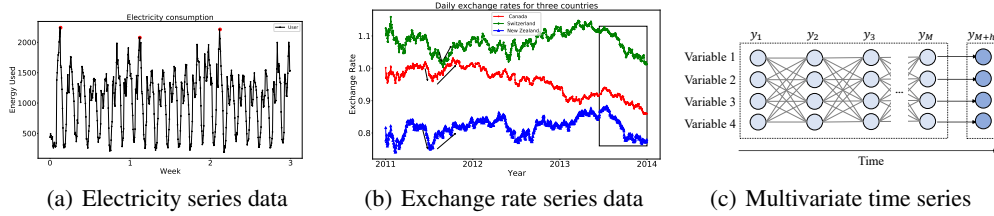


Figure 1: (a) The electricity consumption of an area for 3 weeks, which represents periodic dependence of a single variable. (b) Daily exchange rates for 3 years to show the interaction between variables. (c) Schematic diagram of variable space for multivariate time series.

the high-dimensional variable space mathematically through the tensor product operator. Then, the proposed MVSRTN model models the variable space of multivariate time series.

In the proposed scheme, *N-Order Residual Tensor Network* is the core component. The tensor network is based on the idea of low-rank approximation, which refers to the mode of decomposing and truncating the extremely large dependent space through tensor decomposition methods (Ma et al., 2019; Zhang et al., 2019) in the case of a certain error. For modeling the exponential variable space, we first adopt the Tensor-Train (Oseledets, 2011) algorithm satisfying the demands of processing different timestamps to decompose the variable space representation, and then obtain the space-approximated tensor network by tensor contraction. However, for long-term sequences, the continuous multiplication in the tensor contraction calculation would result in series of gradient problems. Inspired by ordinary differential equations, we extend and introduce N-order Euler discretization method, and the first-order approach can be thought of as residual connection. Finally, the space-approximated tensor network and the proposed N-order residual connection are coupled to model the effective interdependence in the variable space.

In addition, *Series-Variable Encoder* aims at improving the quality of the variable space by reintegrating the time series. In this component, the combination of 1D CNN and self-attention mechanism at series level and variable level effectively enrich the dependency information mapped to the latent space. For making MVSRTN more robust, we use the linear *Skip-Connection Layer* to directly propagate the input scale and coding scale information to the *Output Layer*. Our contributions are of three-folds:

- We represent the high-dimensional variable space, and through theoretical analysis, we demonstrate the space-approximated tensor network has the ability to model the latent variable space.
- We propose the MVSRTN. In this architecture, a designed encoder combines long-term and short-term interactive information to optimize the variable space, a tensor network coupled by N-order residual connection captures the dynamically changing dependencies, and an additional layer propagates the scale features.
- We perform extensive experiments on four benchmark datasets and the results demonstrate the effectiveness of MVSRTN in evaluation metrics.

2 RELATED WORK

Multivariate Time Series In the real world, a large number of multivariate time series tasks are derived from the fields of finance, economy, and transportation. The purpose of this type of task is to capture or predict the trend of a set of variables that change dynamically over time. In the research history of multivariate time series, researchers have reached a basic consensus that multiple variables are interdependent. After a period of development, statistical machine learning models such as Vector Auto-Regressive (Box et al., 2015), Gaussian Process (Frigola, 2015) algorithm can linearly model multivariate time series data, but the complexity of the model expands dramatically with the increase of time or variables. In recent years, with the rise of deep learning, neural networks implicitly capture the dependence of data in a non-linear mode. LSTNet (Lai et al., 2018) uses CNN and RNN to extract the local-term dependent mode of the time series, and extracts the long-term

mode by recurrent-skip layer. Based on the attention mechanism and the LSTNet model, the TPA-LSTM (Shih et al., 2019) model can capture long-term dependencies across multiple time steps. However, algorithms based on neural networks only explicitly model the interaction in the time dimension, and fail to fully analyze the relationships in the latent variable space. The MTCNN (Wu et al., 2020) model based on graph neural network technology explicitly extracts the relationship between variables and achieves the effect of SOTA. However, only the one-way relationship between variable pairs is processed, which is not enough for the dynamic and complex variable space. Therefore, we consider using the idea of approximation to model the variable space.

Tensor Network for Dependent Space The tensor network is a powerful mathematical modeling framework originated from many-body physics, which learns the effective associations in the exponential Hilbert space through the idea of low-rank approximation (Stoudenmire & Schwab, 2016). Based on the training approach of deep learning, Tensor networks just need to approximate the dependent space by training the decomposed parameter space. In the field of natural language, the TSLM (Zhang et al., 2019) model proves that the theoretical model obtained by tensor decomposition of the high-dimensional semantic space has a stronger ability to capture semantic dependence than the RNNs network. The u-MPS (Zauner-Stauber et al., 2018) tensor network based on Tensor-Train decomposition (Oseledets, 2011) is also effective for probabilistic sequence modeling. Tensor networks have natural advantages in the field of modeling high-dimensional dependent spaces (Miller et al., 2021). However, due to the gradient problem, the existing tensor network modeling schemes are still difficult to deal with real-world sequence data with long periods.

Euler Discretization and Residual Connection The residual connection method can solve the problem of the gradient disappearance of the deep network (He et al., 2016). When the gradient is saturated, it can maintain the stability of the network according to the principle of identity mapping. Based on the theory of differential dynamics, the residual connection can be considered as a discretization application of the first-order Euler solver (Chen et al., 2018). Higher-order solvers (for example, the second-order Runge-Kutta) reduce the approximation error caused by the identity mapping (Zhu et al., 2019; Li et al., 2021), and also have better ability to process continuously sampled data (Demeester, 2020).

3 PROBLEM STATEMENT

In this paper, we denote the multivariate time series task as follows. Formally, given the time series signal of T points of time $\mathbf{X} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_T\}^\top$, where $\mathbf{y}_t = \{v_{t,1}, v_{t,2}, \dots, v_{t,D}\}^\top \in \mathbb{R}^D$, T is the number of time steps and D is the number of variables. We leverage $\mathbf{X} \in \mathbb{R}^{T \times D}$ as time series history to forecast the series of next time stamp $\mathbf{y}_{T+h}$, where $h \geq 0$ and it is the desirable horizon which is ahead of the current time stamp. It is worth mentioning that the range of the prediction task h varies on different tasks.

The main purpose of this paper is to model the effective variable dependency in latent space. Hence, we formulate the variable space as shown in Definition 1.

Definition 1. *The tensor product operator is adopted to fully interact with the variable series $\mathbf{y}_t$ indexed by different timestamps. Taking $\mathbf{y}_1$ and $\mathbf{y}_2$ as an example, the interactive function is:*

$$\mathcal{T}_{\mathbf{y}_1, \mathbf{y}_2} = \mathbf{y}_1 \otimes \mathbf{y}_2 = \begin{bmatrix} v_{1,1}v_{2,1} & \cdots & v_{1,D}v_{2,1} \\ \vdots & \ddots & \vdots \\ v_{1,1}v_{2,D} & \cdots & v_{1,D}v_{2,D} \end{bmatrix} \quad (1)$$

where $\mathcal{T}_{\mathbf{y}_1, \mathbf{y}_2} \in \mathbb{R}^{D \times D}$, and we define the whole variable space $\mathcal{T} \in \mathbb{R}^{D^T}$ as:

$$\mathcal{T}_{\mathbf{y}_1, \dots, \mathbf{y}_T} = \mathbf{y}_1 \otimes \mathbf{y}_2 \otimes \cdots \otimes \mathbf{y}_T \quad (2)$$

In the model architecture, we will detail how to model the proposed variable space.

4 MODEL ARCHITECTURE

In this section, we introduce the proposed MVSRTN architecture as shown in Figure 2. The *Series-Variable Encoder* is utilized to optimize the original series data. The *N-Order Residual Tensor*

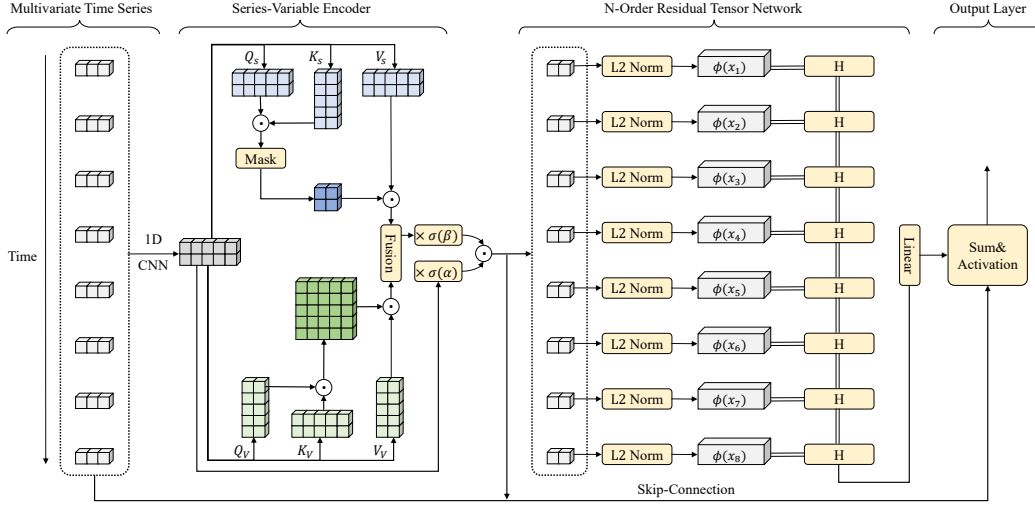


Figure 2: An overview of the proposed MVSRTN.

Network approximates the variable space, captures the dynamic dependence between variables over time, and ensures the ability to model long-term sequences. We leverage *Skip-Connection Layer* components to memory coding and scale information. Finally, the loss function exploited by MVS-RTN is mentioned.

4.1 SERIES-VARIABLE ENCODER

The *Series-variable Encoder* is the first part of MVSRTN, in this component, 1D CNN layer is first exploited to extract short-term patterns in the time dimension as well as local dependencies between variables:

$$c_t = \text{GeLU}(\mathbf{W}_C * \mathbf{y}_t + \mathbf{b}_C) \quad (3)$$

where $*$ represents convolution operation, $\mathbf{W}_C \in \mathbb{R}^{kD \times d}$ and $\mathbf{b}_C \in \mathbb{R}^d$ are all the weight parameters, d is the number of filters, k is the kernel size, and GeLU is an activation function (Hendrycks & Gimpel, 2016). After the above operations and zero-padding, the new sequence representation $\mathbf{C} = \{c_1, \dots, c_T\}^\top \in \mathbb{R}^{T \times d}$ can be obtained.

In addition, the proposed model gains the representation that enriches the global interaction information by self-attention mechanism at series level and variable level, denoted as $\mathbf{G} \in \mathbb{R}^{T \times d}$,

$$\begin{aligned} \mathbf{G}_S &= \text{Softmax}(\text{Tril}(\frac{\mathbf{Q}_S \mathbf{K}_S^\top}{\sqrt{d}})) \mathbf{V}_S \\ \mathbf{G}_V &= \text{Softmax}(\frac{\mathbf{Q}_V \mathbf{K}_V^\top}{\sqrt{T}}) \mathbf{V}_V \\ \mathbf{G} &= \text{ReLU}(\text{Concat}(\mathbf{G}_S; \mathbf{G}_V^\top) \mathbf{W}_G + \mathbf{b}_G) \end{aligned} \quad (4)$$

where $\mathbf{G}_S$ is the output of series-level self-attention, $\mathbf{Q}_S = \mathbf{C} \mathbf{W}_S^Q$, $\mathbf{K}_S = \mathbf{C} \mathbf{W}_S^K$, $\mathbf{V}_S = \mathbf{C} \mathbf{W}_S^V$, and $\mathbf{W}_S \in \mathbb{R}^{d \times d}$ are all weight parameters. $\mathbf{G}_V$ is from variable-level self-attention, $\mathbf{Q}_V = \mathbf{C}^\top \mathbf{W}_V^Q$, $\mathbf{K}_V = \mathbf{C}^\top \mathbf{W}_V^K$, $\mathbf{V}_V = \mathbf{C}^\top \mathbf{W}_V^V$, and $\mathbf{W}_V \in \mathbb{R}^{T \times T}$. Moreover, ReLU is an activation function (Nair & Hinton, 2010), Softmax means normalization, Tril refer to lower triangular matrix and $\mathbf{W}_G \in \mathbb{R}^{2d \times d}$. A gate structure is leveraged to fuse information :

$$\mathbf{X}' = \text{Sigmoid}(\alpha) \times \mathbf{C} + \text{Sigmoid}(\beta) \times \mathbf{G} \quad (5)$$

where α and β are all scale parameters, and Sigmoid is an function (Leshno et al., 1993) commonly used as gates. Finally, we achieve the encoded high-quality multivariate time series, and then map it to the latent variable space.

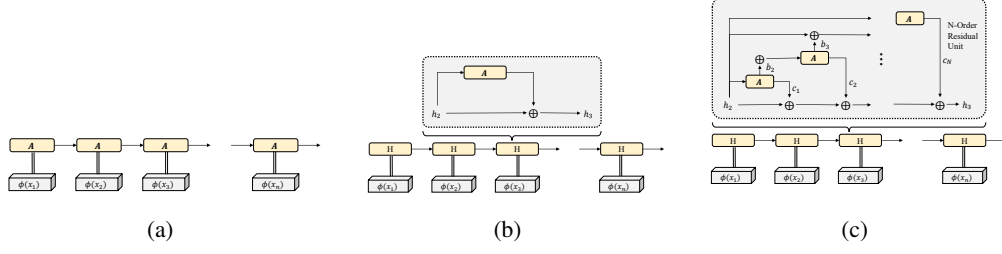


Figure 3: (a) A space-approximated tensor network. (b) A first-order residual tensor network. (c) A N-order residual tensor network.

4.2 N-ORDER RESIDUAL TENSOR NETWORK

In this section, we first give a theoretical analysis of the approximate variable space of the tensor network, and secondly, introduce the algorithm flow of the component.

In the theoretical analysis part, we build the encoded series $\mathbf{X}' = \{\mathbf{x}_1, \dots, \mathbf{x}_T\}^\top \in \mathbb{R}^{T \times d}$ into a variable space. Compared to the Definition 1, based on the time series history $\mathbf{X}'$, we represent the final variable space as follows:

$$\mathcal{T}_{\mathbf{x}_1, \dots, \mathbf{x}_T} = \mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \dots \otimes \mathbf{x}_T \quad (6)$$

In order to learn the effective interactive information in the variable space $\mathcal{T}$, we decompose its representation into a tensor network $\mathcal{A}$ that is contracted from T parameter tensors $\mathcal{W}(t) \in \mathbb{R}^{r_{t-1} \times r_t \times d}$:

$$\begin{aligned} \mathcal{A} &= \mathcal{W}(1) \odot \dots \odot \mathcal{W}(T) \\ &= \sum_{h_1=1}^{r_1} \sum_{h_2=1}^{r_2} \dots \sum_{h_{T-1}=1}^{r_{T-1}} \mathcal{W}_{x_1}^{h_0 h_1} \mathcal{W}_{x_2}^{h_1 h_2} \dots \mathcal{W}_{x_T}^{h_{T-1} h_T} \end{aligned} \quad (7)$$

where $x_t \in \mathbb{R}^d$ and $h_t \in \mathbb{R}^{r_t}$ refer to the indexes of $\mathcal{W}(t)$, and the dimension of the index h_t is the truncated rank of the tensor. This set of values $[r_t]$ determines the degree to which $\mathcal{A}$ approximates the variable space $\mathcal{T}$. $\odot$ is tensor contraction, which can be understood as the dot product between tensors. As shown in Eq.(7), tensor contraction is essentially an addition operation on indexes with the same dimensions to eliminate this index.

It is worth noting that $\mathcal{A}$ and $\mathcal{T}$ are equal when the error is allowed. The above theory is demonstrated in Claim 1, which illustrates that the decomposed parameter space $\mathcal{A}$ has the ability to model the variable space.

Claim 1. *The tensor network $\mathcal{A}$ is obtained by tensor-train decomposition of the variable space $\mathcal{T}$. When the rank of the parameter tensors in Eq.(7) is not truncated, $\mathcal{A}=\mathcal{T}$, and their complexity are all equal to $O(d^T)$; When the following inequalities are satisfied, for the prescribed accuracy ϵ , $\mathcal{A}$ can be considered as the optimal approximation of the space $\mathcal{T}$.*

$$\|\mathcal{T} - \mathcal{A}\|_F \leq \epsilon \|\mathcal{T}\|_F \quad (8)$$

The above is the theoretical discussion of modeling variable space through the idea of low-rank approximation. From a practical point of view, updating the tensor network through automatic differentiation is an effective way to reduce its loss with the variable space. Next, we give the specific modeling process of the *N-Order Residual Tensor Network* component.

For ensuring the probability normalization of the tensor network input (Stoudenmire & Schwab, 2016), L2 regularization is performed on the sequence vector $\mathbf{x}_t$ in $\mathbf{X}'$ at first:

$$\phi(\mathbf{x}_t) = \text{L2}(\mathbf{x}_t) \quad (9)$$

As shown in Figure 3(a), the normalized series $\phi(\mathbf{x}_t)$ is condensed with the network to forecast the future distribution of variables. For avoiding confusion in the position of different time-stamps, the network is modeled step by step in the direction of time, and the parameter tensors

are shared (Zauner-Stauber et al., 2018), namely $\mathcal{W}(1) = \dots = \mathcal{W}(T) = \mathcal{W} \in \mathbb{R}^{r \times r \times d}$:

$$\begin{aligned} \mathbf{h}_t &= \mathbf{A}(\phi(\mathbf{x}_t), \mathbf{h}_{t-1}; \mathcal{W}) \\ &= \mathbf{h}_t \odot \mathcal{W} \odot \phi(\mathbf{x}_t), t \in [1, \dots, T] \end{aligned} \quad (10)$$

where $\mathbf{h}_t \in \mathbb{R}^r$, $\mathbf{h}_0$ refers to all one vector, $\odot$ refers to tensor contraction and r is set as a hyper-parameter bond-dimension (which is adopted to control the ability to approximate the variable space of the tensor network, and the upper limit is $d^{\frac{T}{2}}$).

For long-term sequences, only the space-approximated tensor network will result in gradient problems. In response to the issue, we propose the N-order residual connection approach. First, compared with the tensor network structure mentioned above, we add the residual term when modeling the series information of the current time stamp (as shown in Figure 3(b)):

$$\begin{aligned} \mathbf{h}_t &= \mathbf{H}(\phi(\mathbf{x}_t), \mathbf{h}_{t-1}; \mathcal{W}) \\ &= \mathbf{h}_{t-1} + \mathbf{A}(\phi(\mathbf{x}_t), \mathbf{h}_{t-1}; \mathcal{W}), t \in [1, \dots, T] \end{aligned} \quad (11)$$

Inspired by the numerical solution of ordinary differential equations, its high-order solvers have lower truncation error and higher convergence stability, we propose a tensor network combined with the N-order residual connection as shown in Figure 3(c):

$$\begin{aligned} \mathbf{h}_t &= \mathbf{H}(\phi(\mathbf{x}_t), \mathbf{h}_{t-1}; \mathcal{W}) \\ &= \mathbf{h}_{t-1} + \sum_{j=1}^N a_j \mathbf{z}_j, t \in [1, \dots, T] \end{aligned} \quad (12)$$

$$\begin{cases} \mathbf{z}_j = \mathbf{A}(\phi(\mathbf{x}_t), \mathbf{h}_{t-1}; \mathcal{W}) & j = 1 \\ \mathbf{z}_j = \mathbf{A}(\phi(\mathbf{x}_t) + b_j, \mathbf{x}_{t-1} + b_j \mathbf{z}_{j-1}; \mathcal{W}) & 2 \leq j \leq N \end{cases}$$

where a_j and b_j are all fixed coefficients. Please see the Appendix A for the coefficient values calculated by the Taylor's Formula of the different order residual connections.

$$\mathbf{x}^U = \mathbf{h}_T \mathbf{W}_U + \mathbf{b}_U \quad (13)$$

Finally, a linear component maps $\mathbf{h}_T \in \mathbb{R}^r$ to $\mathbf{x}^U \in \mathbb{R}^D$, where $\mathbf{W}_U \in \mathbb{R}^{r \times D}$ and $\mathbf{b}_U$ are all parameter weights.

4.3 SKIP CONNECTION LAYER

The layer is added after the input layer and encoder layer respectively, which directly connected to the *Output Layer*. For multivariate time series, numerous nonlinear structures in the model make the output scale insensitive to the input scale (Lai et al., 2018), resulting in a significant reduction in the prediction accuracy of the model. In addition, in order to memorize the encoder states, we also adopt the layer to directly propagate the encoding information to the output.

$$\mathbf{x}^L = \mathbf{L} \mathbf{W}_L + \mathbf{b}_L \quad (14)$$

the known input data is $\mathbf{X} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_T\}^\top \in \mathbb{R}^{T \times D}$, and the selected series is $\mathbf{L} = \{\mathbf{y}_{T-l+1}, \dots, \mathbf{y}_T\} \in \mathbb{R}^{D \times l}$, $\mathbf{W}_L \in \mathbb{R}^{l \times 1}$ and $\mathbf{b}_L \in \mathbb{R}^1$ are all weight parameters.

$$\mathbf{x}^M = (\mathbf{M} \mathbf{W}_M + \mathbf{b}_M) \mathbf{W}_O + \mathbf{b}_O \quad (15)$$

the selected series is $\mathbf{M} = \{\mathbf{x}_{T-m+1}, \dots, \mathbf{x}_T\} \in \mathbb{R}^{d \times m}$, where the encoder series is $\mathbf{X}' = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T\}^\top \in \mathbb{R}^{T \times d}$, and weight parameters are $\mathbf{W}_M \in \mathbb{R}^{m \times 1}$, $\mathbf{b}_M \in \mathbb{R}^1$, $\mathbf{W}_O \in \mathbb{R}^{d \times D}$ and $\mathbf{b}_O \in \mathbb{R}^D$.

4.4 OUTPUT LAYER AND LOSS FUNCTION

Finally, the prediction result at time stamp $s = T + h$ of the network is $\mathbf{Y}'_s \in \mathbb{R}^D$:

$$\mathbf{Y}'_s = \mathbf{x}^U + \mathbf{x}^L + \mathbf{x}^M \quad (16)$$

Moreover, we utilize the validation set to decide which of the two loss functions is better for our model. we leverage Mean Absolute Error (MAE, also denoted as L1 loss function) and Mean Square

Error (MSE, also denoted as L2 loss function) to update the network parameters. MSE is simple to calculate, which is the default loss function of many prediction tasks, and MAE has better robustness for outliers.

$$\begin{aligned} \text{MSE} &= \sum_{s \in \Omega_{train}} \sum_{i=1}^D (\mathbf{Y}_{s,i} - \mathbf{Y}'_{s,i})^2 \\ \text{MAE} &= \sum_{s \in \Omega_{train}} \sum_{i=1}^D |\mathbf{Y}_{s,i} - \mathbf{Y}'_{s,i}| \end{aligned} \quad (17)$$

where Ω_{train} is a set of time stamps for model training.

5 EXPERIMENTS

In the section, we perform extensive experiments on four multivariate time series forecasting benchmark datasets between MVSRTN and seven baselines. Moreover, in order to evaluate the effectiveness of the proposed model, we conduct ablation experiments and order experiments.

5.1 DATASETS AND BASELINES

Our experiments are based on three datasets which are publicly available:

- Traffic ¹: This dataset is composed of 48 months hourly data from the California Department of Transportation during 2015 and 2016. The data describes the road occupancy rates (between 0 and 1) measured by different sensors on San Francisco Bay area freeways.
- Exchange-Rate: A dataset that consist of the daily exchange rates of eight countries including Australia, British, Canada, Switzerland, China, Japan, New Zealand and Singapore ranging from 1990 to 2016.
- Solar-Energy ²: This dataset consists of the solar power production records in the year of 2006 and these records are sampled every 10 minutes from 137 PV plants in Alabama State.
- Electricity ³: This dataset from the UCI Machine Learning Repository is composed of the electricity consumption in kWh which was recorded every 15 minutes from 2012 to 2014 for 321 clients.

We compare with a number of forecasting methods. **AR**: An auto-regressive model for time series forecasting. **GP** (Frigola, 2015; Roberts et al., 2012): A Gaussian Process time series model for prediction. **LSTNet-skip** (Lai et al., 2018): A LSTNet model (A deep neural network that combines convolutional neural networks and recurrent neural networks, and could catch the long-term and short-term patterns of time series) with skip-RNN layer. **LSTNet-Attn** (Lai et al., 2018): A LSTNet model with temporal attention layer. **TPA-LSTM** (Shih et al., 2019): An attention-recurrent neural network for multivariate time series forecasting. **MTGNN** (Wu et al., 2020): A graph neural network model for the forecasting of multivariate time series. **MTGNN-sampling** (Wu et al., 2020): MTGNN model trained on a sampled subset of a graph in each iteration.

All methods are evaluated with two metrics: root relative squared (RSE) and empirical correlation Coefficient (CORR). Lower value is better for RSE while higher value is better for CORR. In addition, the details of the datasets and metrics are in Appendix B.

5.2 TRAINING DETAILS

For all tasks, we implement our model with Pytorch-1.20, and train them on a NVIDIA GeForce RTX 2080 Ti GPU. For Traffic and Solar datasets, the size of the series history window T is 168, for Solar-Energy it is 144, and for Exchange-rate it is 120. All the weight parameters are initialized with Xavier (Glorot & Bengio, 2010) and normalized by weight normalization (Salimans &

¹<http://pems.dot.ca.gov>

²<http://www.nrel.gov/grid/solar-power-data.html>

³<https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014>

Table 1: Results summary (in RSE and CORR) of all methods on four datasets.

Dateset		Solar-Energy		Traffic		Electricity		Exchange-Rate	
		Horizon		Horizon		Horizon		Horizon	
Models	Metrics	6	24	6	24	6	24	6	24
AR	RSE	0.379	0.8699	0.6218	0.63	0.1035	0.1054	0.0279	0.0445
	CORR	0.9263	0.5314	0.7568	0.7519	0.8632	0.8595	0.9656	0.9357
GP	RSE	0.3286	0.7973	0.6772	0.5995	0.1907	0.1273	0.0272	0.058
	CORR	0.9448	0.5971	0.7406	0.7909	0.8334	0.8818	0.8193	0.8278
LSTNet-skip	RSE	0.2559	0.4643	0.4893	0.4973	0.0931	0.1007	0.028	0.0449
	CORR	0.969	0.887	0.869	0.8588	0.9135	0.9119	0.9658	0.9354
LSTNet-Attn	RSE	0.2538	0.4403	0.4973	0.53	0.0953	0.1059	0.0321	0.059
	CORR	0.9696	0.8995	0.8669	0.8429	0.9095	0.9025	0.9656	0.9339
TPA-LSTM	RSE	0.2347	0.4389	0.4658	0.4765	0.0916	0.1006	0.0241	0.0444
	CORR	0.9742	0.9081	0.8717	0.8629	0.9337	0.9133	0.9709	0.9381
MTGNN	RSE	0.2348	0.427	0.4754	0.4535	0.0878	0.0953	0.0259	0.0456
	CORR	0.9726	0.9031	0.8667	0.881	0.9316	0.9234	0.9708	0.9372
MTGNN-sampling	RSE	0.2521	0.4386	0.4435	0.4537	0.0862	0.0976	0.0271	0.0454
	CORR	0.9687	0.899	0.8815	0.8758	0.9354	0.9219	0.9704	0.9382
MVSRTN	RSE	0.2383	0.4165	0.4423	0.4598	0.0912	0.1003	0.0247	0.044
	CORR	0.9720	0.9070	0.8818	0.8714	0.9317	0.9242	0.9709	0.9384

Kingma, 2016). As for learning method, we use the Adam optimizer (Kingma & Ba, 2014) and an exponentially decaying learning rate with a linear warm up. The initial learning rate is set from 0.0005 to 0.001 and the batch size is 128. We perform dropout after each layer, except input and output ones, and the rate usually is set to 0.1 or 0.2. The 1D CNN component has 50 filters, the filter size is set from 3 to 6. In the tensor network, the bond-dimension r is tuned range from 50 to 100, and the order of the residual connection is chosen from $\{1,2,4\}$. For the chosen size l and m in *Skip-Connection Layer*, the l is 24 for all datasets. and the m is set to the history window size T for different datasets.

5.3 MAIN RESULTS

Table 1 shows the prediction performances of all the models on all datasets, and the metrics are RSE and CORR. We set horizon = $\{6, 12\}$, respectively, which means the horizons are set to 6 or 24 days over the Exchange-Rate dataset, to 6 or 24 hours over the Electricity and Traffic task, to 60 or 240 minutes over the Solar-Energy data. It is worth noting that the larger the horizons, the harder the prediction tasks. The best result for each metric of different datasets is highlighted in bold face in the table.

Compared with the MTGNN model (state-of-the-art method), our proposed model achieves competitive results. In particular, for the exchange-rate dataset, the results of MVSRTN on the four (horizon, metric) pairs all outperform the MTGNN model. For the Solar-Energy data when horizon=24, MVSRTN lowers down RSE by 2.46%. In general, in addition to the traffic dataset, our method has a greater advantage when the horizon is 24. The reasons are that the graph modeling method of MTGNN is more suitable for traffic data, and when the series data is difficult to predict (horizon=24), our model plays the role of modeling semantic space.

5.4 ABLATION EXPERIMENTS

To demonstrate the effectiveness of each model structure on Solar-Energy, Traffic, Electricity and Exchange-Rate data, we compare MVSRTN with 4 variants as follows:

- **MVSRTN-nA**: We remove the self-attention at series and variable level component such that there is a lack of global interactive information at series level and variable level.

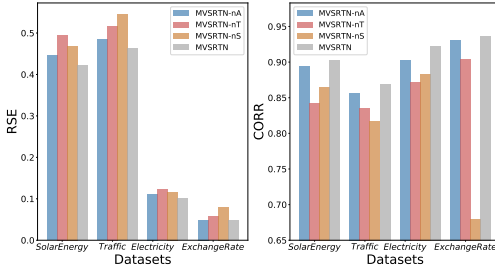


Figure 4: Ablation Experiments on four datasets when horizon=24.

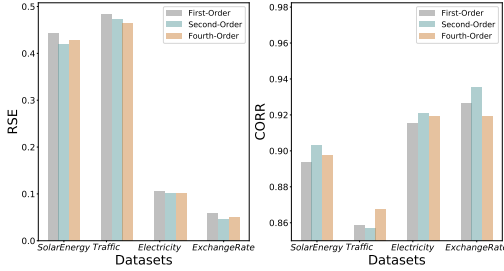


Figure 5: Order Analysis of residual connection on four datasets when horizon=24.

- **MVSRTN- nT** : We remove all the proposed tensor network such that the model cannot learn the variable space we defined.
- **MVSRTN- nS** : MVSRTN without skip-connection layer, which means that scale and encoder information cannot be directly propagated to the output layer.
- **MVSRTN**: Modeling Variable Space with Residual Tensor Networks.

As shown in Figure 4, reported results are the average of 10 runs. For all the datasets, removing the residual tensor networks and removing the skip-connection layer both cause significant performance degradation. In particular, for the CORR metric of the Exchange-Rate dataset, the MVSRTN- nS variant basically loses the ability to accurately forecast, which means the exchange rate data is more sensitive to scale information. Moreover, solar energy and electricity data are more obviously affected by residual tensor network components.

5.5 ORDER ANALYSIS

In order to verify the effects of different orders of the residual tensor network, as shown in Figure 5, we design the comparative experiments over the orders of 1, 2, 4 on the Solar-Energy, Traffic, Electricity and Exchange-Rate datasets. The experimental results show that the second-order residual tensor network (RTN) achieves the best results on both RSE and CORR metrics for solar energy data and exchange rate data. For traffic data, the effect of fourth-order RTN is better than that of first-order and second-order. In addition, the result of the fourth-order RTN is similar to the second-order RTN on the Electricity dataset. This verifies the effectiveness of the proposed higher-order residual connection.

6 CONCLUSION

In this paper, we propose a novel framework (MVSRTN) for fully exploiting the dependencies in the variable space in the task of multivariate time series forecasting. We exploit a tensor network based on the idea of low-rank approximation to model the variable space and shared tensor components to ensure the translation invariance of the network. Additionally, we propose an N -order residual connection approach and couple it to the space-approximated tensor network to improve the ability to model long-term sequences. Moreover, a series-variable encoder is designed to improve the quality of the variable space, and we leverage the skip-connection layer to achieve the dissemination of information such as scale. According to the comparison with 7 baselines, we show the efficiency of the framework of MVSRTN, and experimental results show the effect of modeling variable space for multivariate time series forecasting.

REFERENCES

- George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and

- R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL <https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf>.
- Jiezhu Cheng, Kaizhu Huang, and Zibin Zheng. Towards better forecasting by fusing near and distant future visions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3593–3600, 2020.
- Thomas Demeester. System identification with time-aware neural sequence models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3757–3764, 2020.
- Roger Frigola. *Bayesian time series learning with Gaussian processes*. PhD thesis, University of Cambridge, 2015.
- Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. JMLR Workshop and Conference Proceedings. URL <http://proceedings.mlr.press/v9/glorot10a.html>.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, 2018.
- Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.
- Bei Li, Quan Du, Tao Zhou, Shuhan Zhou, Xin Zeng, Tong Xiao, and Jingbo Zhu. Ode transformer: An ordinary differential equation-inspired model for neural machine translation. *arXiv preprint arXiv:2104.02308*, 2021.
- Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. A tensorized transformer for language modeling. *Advances in Neural Information Processing Systems*, 32:2232–2242, 2019.
- Jacob Miller, Guillaume Rabusseau, and John Terilla. Tensor networks for probabilistic sequence modeling. In *International Conference on Artificial Intelligence and Statistics*, pp. 3079–3087. PMLR, 2021.
- Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.
- I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011. doi: 10.1137/090752286. URL <https://doi.org/10.1137/090752286>.
- Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31:7785–7794, 2018.
- S. Roberts, M. Osborne, M. Ebdon, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions*, 371(1984):20110550, 2012.
- Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in neural information processing systems*, pp. 901–909, 2016.

- Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8):1421–1441, 2019.
- Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL <https://proceedings.neurips.cc/paper/2016/file/5314b9674c86e3f9d1ba25ef9bb32895-Paper.pdf>.
- Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 753–763, 2020.
- Dongkuan Xu, Wei Cheng, Bo Zong, Dongjin Song, Jingchao Ni, Wenchao Yu, Yanchi Liu, Haifeng Chen, and Xiang Zhang. Tensorized lstm with adaptive shared memory for learning trends in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 1395–1402, 2020.
- Valentin Zauner-Stauber, Laurens Vanderstraeten, Matthew T Fishman, Frank Verstraete, and Jutho Haegeman. Variational optimization algorithms for uniform matrix product states. *Physical Review B*, 97(4):045145, 2018.
- Lipeng Zhang, Peng Zhang, Xindian Ma, Shuqin Gu, Zhan Su, and Dawei Song. A generalized language model in tensor space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7450–7458, 2019.
- Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Multivariate time-series anomaly detection via graph attention network. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 841–850. IEEE, 2020.
- Mai Zhu, Bo Chang, and Chong Fu. Convolutional neural networks combined with runge-kutta methods, 2019. URL <https://openreview.net/forum?id=HJNJws0cF7>.

A THE COEFFICIENT VALUES OF RESIDUAL TENSOR NETWORK

Table 2: The coefficients of first-order, second-order, and fourth-order residual tensor networks.

Coefficients	First-order	Second-order	Fourth-order
a_1	1.0	1/2	1/6
a_2		1/2	1/3
a_3	–	–	1/3
a_4	–	–	1/6
b_2	–	1.0	1/2
b_3	–	–	1/2
b_4	–	–	1.0

Table 2 shows the coefficients used in Equation 4.2.

B DATASETS AND METRICS

In Table 3, we summarize statistics of benchmark datasets. Moreover, two conventional evaluation metrics exploited by us are defined as follows:

Root Relative Squared Error (RSE):

$$\text{RSE} = \frac{\sqrt{\sum_{s \in \Omega_{test}} (\mathbf{Y}_{s,i} - \mathbf{Y}'_{s,i})^2}}{\sqrt{\sum_{s \in \Omega_{test}} (\mathbf{Y}_{s,i} - \text{mean}(\mathbf{Y}))^2}} \quad (18)$$

Table 3: Dataset Statistics

Datasets	# Sample	# Variables	# Sample Rate
Traffic	17544	862	1 hour
Solar-Energy	52560	137	10 minutes
Electricity	26304	321	1 hour
Exchange-Rate	7588	8	1 day

Empirical Correlation Coefficient (CORR):

$$\text{CORR} = \frac{1}{n} \sum_{i=1}^n \frac{\sum_s (\mathbf{Y}_{s,i} - \text{mean}(\mathbf{Y}_i))(\mathbf{Y}'_{s,i} - \text{mean}(\mathbf{Y}'_i))}{\sqrt{\sum_s (\mathbf{Y}_{s,i} - \text{mean}(\mathbf{Y}_i))^2 (\mathbf{Y}'_{s,i} - \text{mean}(\mathbf{Y}'_i))^2}} \quad (19)$$

where Ω_{test} is a set of time stamps for model test, and lower value is better for RSE while higher value is better for CORR.