# Multivariate Time Series Forecasting exploiting Tensor Projection Embedding and Gated Memory Network

Zhenxiong Yan[1], Kun Xie[1*], Xin Wang[2], Dafang Zhang[1], Gaogang Xie[3,4], Kenli Li[1], Jigang Wen[5]
[1] College of Computer Science and Electronics Engineering, Hunan University, Changsha 410082, China
[2] Department of Electrical and Computer Engineering, State of New York University at Stony Brook, USA
[3] Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China
[4] School of Computer Science and Technology, University of Chinese Academy of Sciences, China
[5] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
Corresponding author *

2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS) | 978-1-6654-1494-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/IWQOS52092.2021.9521337

***Abstract*—Time series forecasting is very important and plays critical roles in many applications. However, making accurate forecasting is a challenge task due to the requirements of learning complex temporal and spatial patterns and combating noise during the feature learning. To address the challenge issues, we propose TEGMNet, a Tensor projection Embedding and Gated Memory Network for multivariate time series forecasting. To more accurately extract local features and reduce the influence of noise, we propose to amplify the data using several data transformation techniques based on MDT (Multi-way delay embedding transform) and TFNN (tensor factorized neural network) to transform the original 2D matrix data to low dimensional 3D tensor data. The local features are then extracted through convolution and LSTM upon the 3D tensor. We also design a long-term feature extraction module based on the structure of gated memory network, which can largely enhance the long-term pattern feature learning ability when the multivariate time series has complex long-term dependencies with dynamic-period patterns. We have done extensive experiments by comparing our TEGMNet with 7 baseline algorithms using 4 real data sets. The experiment results demonstrate that TEGMNet can achieve very good prediction performance even through the data are polluted with noise.**

***Index Terms*—Multivariate time series, Forecasting, Neural network, Tensor projection, Memory network**

## I. Introduction

Time series forecasting is very important and plays critical roles in many applications. Some example applications are the forecasting of traffic flows on the road, output of solar power plants, air quality index of different cities. In these applications, users are usually interested in predicting new trends or potential incidents based on historical observations of time-series signals. Accurate forecasting results can facilitate the implementation of advanced applications. For example, better travel routes can be selected based on the prediction of traffic congestion patterns in advance, and a reasonable power generation plan can be set up based on prediction of electricity consumption in an area.

However, making accurate forecasting is a challenging task because of the following issues.

- **Learning complex temporal periodic patterns.** Applications in the real-world usually entail a mixture of short-term and long-term repetitive patterns [15].
- **Learning both temporal and spatial patterns.** In real life, data are often presented in the form of multivariate [23]. There are some correlations between these variables (the correlations are usually called spatial features).
- **Combating noise during the feature learning** With the impact of unavoidable noises, the normal time series data will not be smooth and with some spikes, which makes the temporal and spatial patterns difficult to extract.

Based on the methodologies taken, existing time series forecasting algorithms can be divided into three categories.

The first category takes the advanced statistical method, including autoregression (AR) [5], autoregressive integrated moving average (ARIMA) [13], Gaussian process (GP) [18] and their variants. These methods usually assume certain distribution or function form of time series, which makes them unable to capture complicated underlying non-linear relationships. The second category applies neural networks in the sequential modeling. Although RNN (recurrent neural network) [9] and its variants LSTM (long short-term memory) [11] and GRU (gated recurrent unit) [8] can achieve good performance in learning nonlinearity and temporal features, due to the effect of gradient vanishing [4], they may fail to learn the long-term information. In the third category, several hybrid neural networks are proposed to extract complex temporal and spatial patterns by combining multiple neural networks. LSTNet [15] leverages both the convolutional layer and the recurrent-skip layer to capture complex long-term dependencies. MTNet [6] is inspired by End-To-End Memory Network [20], uses a single layer memory component to help mine the long-term dependencies. Taking advantages of different neural networks, this type of forecasting algorithms achieves much better forecasting performance.

However, existing time series prediction studies often ignore the noise interference in the feature learning and can hardly work robustly. To address above challenging issues, we propose TEGMNet, a Tensor Projection Embedding with Gated Memory Network for multivariate time series forecasting. To enable accurate and robust forecasting, TEGMNet divides a long period of time series into multiple time series blocks. To achieve accurate multivariate time series forecasting, we propose a set of techniques to extract temporal and spatial pattern within the block and across the blocks:

- To more accurately extract the short-term patterns, we propose to transform the block from 2D matrix to 3D tensor exploiting MDT (Multi-way delay embedding transform). The local information in the multivariate time series can be amplified, which provides the possibility to more accurately extract the short-term hidden features.
- To make our algorithm more robust despite of unavoidable noise in the multivariate time series data, we design tensor factorization neural network to transform the high-dimensional 3D tensor to the low-dimensional 3D core tensor. By referring to the idea of tensor factorization, the influence of noise is reduced, and the local patterns can be mined more effectively.
- We have conducted extensive experiments to compare our TEGMNet with 7 baseline algorithms using 4 real datasets. The results demonstrate that our TEGMNet can achieve very good prediction performance even though the data are polluted with noise.

The rest of the paper is organized as follows. In Section II, we present the problem and an overview of our TEGMNet. In Section III and Section IV, we provide the detailed design of two key modules in TEGMNet. We evaluate the performance in Section V, and conclude the work in Section VI.

## II. Problem and solution overview

### A. Problem

In this paper, we focus on the problem of multivariate time series forecasting. We intend to predict the signals in the future by leveraging historical time series data. Given a set of observed time series signals $Y = \{y_1, y_2, ..., y_{\mathbb{T}}\}$, where the length of time series is $\mathbb{T}$ and $y_{\mathbb{T}} \in R^d$, and $d$ represents the variable dimension, we aim to predict a series of future signals in a rolling forecasting fashion. That is, given $\{y_1, y_2, ..., y_{\mathbb{T}}\}$, we want to predict $y_{\mathbb{T}+h}$ where $h$ is the desirable horizon ahead of the current time stamp. The input matrix at time stamp $\mathbb{T}$ can be expressed as $\mathbb{X}_{\mathbb{T}} = \{y_1, y_2, ..., y_{\mathbb{T}}\} \in R^{d\times T}$. In most of the cases, the horizon of the forecasting task is chosen according to the demands of the environmental settings.

### B. Method overview

To address above issues in multivariate time series forecasting, we propose Tensor projection Embedding with Gated Memory Network, called TEGMNet. As shown in Fig.1, TEGMNet consists of two parts: Tensorized Projection Embedding and Time Series Network with Gated Memory.

TEGMNet divides a long period of time series $\mathbb{X}_{\mathbb{T}}$ into multiple time series blocks $\{X_I\} = \{X_1, X_2, ..., X_n\}$, where $X_I \in R^{d\times T}$, $n$ is a hyperparameter, $\mathbb{T} = n \times T$. The blocks in $\{X_I\}$ are continuous in time but do not overlap. To mine the short-term and long-term dependencies effectively, $\{X_I\}$ is divided into two parts: $\{X_1, X_2, ..., X_{n-1}\}$ and $X_n$. The first part $\{X_1, X_2, ..., X_{n-1}\}$ corresponds to the long-term historical data that can be regarded as context information and stored in memory. The second part $X_n$ corresponds to the newly arrived data block closest to the target to be predicted.

In the embedding stage, we aim to extract local features in each block, while reducing the sensitivity to noise. To achieve this, as shown in Fig.2, we design the Tensorized Projection Embedding module to embed the original input $X_i \in R^{d\times T}$ into a fixed length embedding representation through a series of embedding components. The detailed design of this module can be found in Section III.

In the memory network stage, our objective is to extract the common long-term pattern features in the datasets for the prediction task. The module implemented with Time Series Network with Gated Memory writes the embedding representations of the long-term historical data $\{X_i \mid 1 \le i \le n-1\}$ into a fixed size of memory.

By combining long-term dependencies with short-term dependencies, TEGMNet finally concatenates the output $o$ of the Gated Memory module and the embedding representation $u$ of the newly arrived $X_n$, and feeds them to a dense layer to generate the forecasting outputs.

## III. Embedding with Tensor Projection

The main purpose of Tensor Projection Embedding module is to extract fine-grained temporal and spatial features in each block and effectively reduce the noise interference. This module consists of four calculation components, namely MDT (multi-way delay embedding transform), TFNN (tensor-factorized neural network), Convolution, and LSTM.

Firstly, through MDT, the original block $X_i \in R^{d\times T}$ is transformed from a 2-D matrix data to a 3-D tensor $X_i^{mdt} \in R^{d\times\tau\times\hat{T}}$. The $X_i^{mdt}$ is then projected into a smaller core tensor $X_i^{tfnn}$ by TFNN component. Through Convolution, the spatial features of the block are extracted and represented by $X_i^{conv}$. Finally, the $X_i^{conv}$ is further feeded into LSTM to extract the temporal feature and get the output $X_i^{lstm}$. The above processes are expressed as follows:

$$\begin{gathered} X_i^{mdt} = MDT(X_i), X_i^{tfnn} = TFNN(X_i^{mdt}), \\ X_i^{conv} = Conv(X_i^{tfnn}), X_i^{lstm} = LSTM(X_i^{conv}), \\ Embedding(X_i) = X_i^{lstm}, 1 \le i \le n. \end{gathered} \tag{1}$$

The detailed procedures of each step will be introduced in the remaining of the section.

### A. MDT Component

Multi-way delay embedding transform (MDT) is an emerging technique to Hankelize available data to a high-order block Hankel tensor [19], [21]. We exploit MDT to transform the

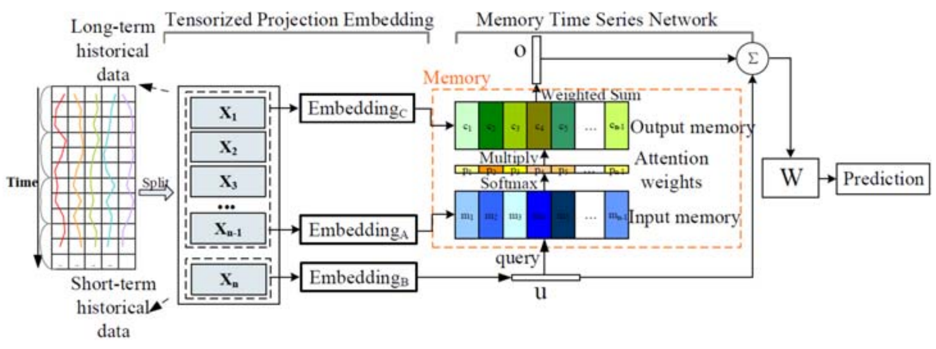

Fig. 1. An overview of our TEGMNet

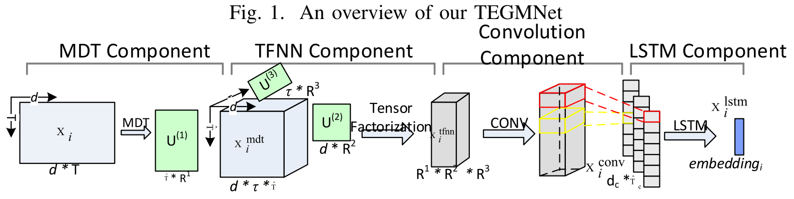

Fig. 2. Components in tensor projection embedding

block from a 2-D matrix to a 3-D tensor so that the local information can be enlarged for the better mining of short-term features in the block. Given a block $X_i \in R^{d\times T}$, we conduct the MDT along the temporal direction, expressed as

$$\begin{aligned} X_i^{mdt} &= MDT(X_i) = H_\tau(X_i) \\ &= fold_\tau(X_i * \mathbf{S}^{\mathrm{T}}) \in R^{d\times\tau\times\hat{T}}, 1 \le i \le n, \end{aligned} \tag{2}$$

where $\tau$ is a hyper-parameter and $\hat{T} = T - \tau + 1$.

A 2-D block $X \in R^{d\times T}$ is transformed to a 3-D tensor $X^{mdt} \in R^{d\times\tau\times(T-\tau+1)}$, called Hankel tensor, using a duplication matrix $S$. The Hankel tensor is always low-rank and smooth, since the time series is inherently correlated, and there are recurring columns in the Hankel tensor.

### B. TFNN Component

In practical applications, there are usually different levels of noise in the data. Tensor factorization [14] is essentially a high-order generalization of matrix factorization, which provides an effective solution to analyze a multiway structural observation. Low rank Tucker decomposition has been proved to be effective tensor factorization algorithm in removing noise [16], [24].

To remove the noise, our model exploits TFNN [7], a tensor factorization scheme, to train the parameters factor matrices in Tucker decomposition through neural network. As shown in Fig.2, the TFNN is constructed by factorizing and activating an input tensor $X_i^{mdt}$ into the a low-dimensional core tensor $X_i^{tfnn}$ by using three-way factor matrices $\{U^{(1)}, U^{(2)}, U^{(3)}\}$ and activation function h(·). The factor matrices are obtained by training as parameters of neural network model.

The process of TFNN can be also expressed as follows.

$$\begin{aligned} X_i^{tfnn} &= TFNN(X_i^{mdt}) \\ &= h(X_i^{mdt} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)}) \in R^{J_1\times J_2\times J_3}, 1 \le i \le n, \end{aligned} \tag{3}$$

where $U^{(n)} \in R^{I_n\times J_n}$ with $n = 1, 2, 3$ are the factor matrices in Tucker decomposition, $J_n$ with $n = 1, 2, 3$ are the hyper-parameters. Note that, to ensure that the temporal mode is not compressed, we make $J_3$ equal to $I_3$.

By parameterized factor matrices, the input tensor $X_i^{mdt}$ is projected into a low dimensional core tensor $X_i^{tfnn}$ from $R^{I_1\times I_2\times I_3}$ to $R^{J_1\times J_2\times J_3}$. In this way, we can not only extract salient features from multilinear subspace, but also effectively reduce the influence of noise in the data, thus significantly improving the robustness of TEGMNet.

### C. Convolution Component

Convolution is a general method to extract features, especially local features [23]. We perform convolution operation on the core tensor obtained by TFNN to extract short-term patterns in the time dimension (i.e., temporal features) and local dependencies between variables (i.e., spatial features). In our model, 3D convolution is used, which can extract features more effectively and reduce the parameters of the model compared with full connectivity layer. The $k$-th filter sweeps through the input tensor $X^{tfnn}$ and produces

$$\begin{aligned} X_i^{conv} &= Conv(X_i^{tfnn}) = \{h_{ik}\} \\ &= \{RELU(W_k * X^{tfnn} + b_k)\} \in d_c \times \hat{T}_c, 1 \le i \le n, \end{aligned} \tag{4}$$

where $*$ denotes the convolution operation. We make each vector $h_{ik}$ of length T by zero-padding on the input tensor $X_i^{tfnn}$. The output of the convolutional layer is of size $d_c \times \hat{T}_c$ where $d_c$ denotes the number of filters, $\hat{T}_c$ is the length of time after the convolution operation.

### D. LSTM Component

We exploit LSTM [11] to learn local short-term dependency in a block since LSTM is good at memorizing temporal information in a short local time.

$$Embedding(X_i) = X_i^{lstm} = LSTM(X_i^{conv}) \in R^{d_e}, 1 \le i \le n. \tag{5}$$

The input $X_i^{conv}$ can be considered as a short time series, with the time length of $\hat{T}_c$ and the input dimension of $d_c$. The values of each time point $x_i$ are input into LSTM in turn, and the output result $X_i^{lstm}$ is obtained after a series of gated-control operations and value transfer within LSTM.

## IV. Time Series Network with Gated Memory

### A. Single Layer structure

For the long-term pattern, it is obviously insufficient to use only the LSTM / RNN and other recurrent neural networks. Enlightened by structure of the Gated End-to-End Memory Networks [17], [20], we design our long term pattern extraction module, called Time Series Network with Gated Memory, which can effectively mine the long-term pattern dependency in the historical blocks and determine which blocks can improve the accuracy of forecasting.

As shown in Fig.1, we use two different Tensor Projection Embedding to obtain the embedded representation $\{m_i \mid 1 \le i \le n-1\}$ and $\{c_i \mid 1 \le i \le n-1\}$, which are marked as $Embedding_A$ and $Embedding_C$ and play different roles respectively. Given a set of long-term historical data $\{X_i \mid 1 \le i \le n-1\} = X_1, \cdots, X_{n-1}$, we take the follow processes

$$\begin{gathered} m_i = Embedding_A(X_i) \\ c_i = Embedding_C(X_i), 1 \le i \le n-1 \end{gathered} \tag{6}$$

The newly arrived data $X_n$ is also embedded via another $Embedding_B$ to get query embedding representation $u$ where $u \in R^{d_e}$, that is,

$$u = Embedding_B(X_n) \tag{7}$$

Note that $Embedding_A$, $Embedding_B$, and $Embedding_C$ adopt the same structure described in Section III, while uses different weight values in the structure for their different roles in our TEGMNet.

In the embedding space, we perform the query to compute the attention weights $\{p_i \mid 1 \le i \le n-1\}$ between $u$ and each memory vector $m_i$ by taking the inner product followed by a softmax function.

$$p_i = Softmax(u^\top m_i) \tag{8}$$

where $Softmax(z_i) = e^{z_i} / \sum_j e^{z_j}$ and we can define a vector $p$ as the attention weight vector over the memory inputs.

The output vector (i.e., $o$) of the Memory Time Series Network module is then calculated as a sum over the transformed inputs $c_i$ weighted by attention weights:

$$o = \sum\nolimits_i p_i c_i \tag{9}$$

### B. Multiple Layers structure

Single layer Memory networks often do not perform well in applications because the degree of freedom of data features is too large, and the parameters of neural networks do not fit the features well to describe the entire sample. To better extract those features, the memory module is extended to the $K$ hop structure. As shown in Fig.3, between different layers, a Gated mechanism [17] is introduced to regularize the memory. It introduces a transform gate $T$ and a carry Gate $C$ ( $C = 1-T$ is often chosen ). The input to layers above the first is the combination of the output $o^k$ and the input $u^k$ from layer $k$:

$$\begin{gathered} T^k(u^k) = \sigma(W_T^k u^k + b_T^k), \\ u^{k+1} = o^k \odot T^k(u^k) + u^k \odot (1 - T^k(u^k)) \end{gathered} \tag{10}$$

where $\sigma$ is the sigmoid function, $W_T^k$ and $b_T^k$ are the hop-specific parameter matrix and bias term for the $k^{th}$ hop and $T^k(x)$ the transform gate for the $k^{th}$ hop.

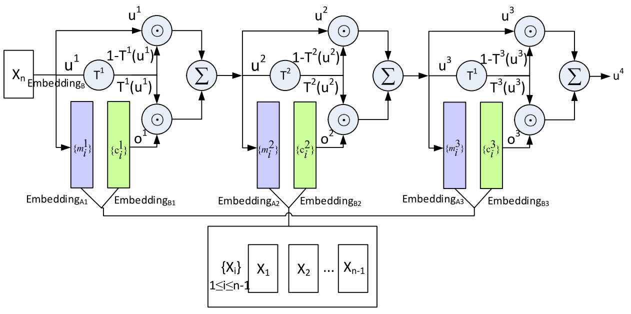

Fig. 3. multiple layer gated memory structure

*1) Generating the final prediction:* We use a dense layer to combine the output vector $o^k$, the embedding representation $u^1$ of $X_n$ and the internal representation vector $u^{k+1}$ of the top memory layer. The concatenation is then passed through a final weight matrix $W^d$. The output of the dense layer is computed as,

$$y = W^d[o^k; u^1; u^{k+1}] + b \tag{11}$$

where $[o^k; u^1; u^{k+1}]$ is the concatenation of $o^k$, $u^1$ and $u^{k+1}$, and $y \in R^d$ is the final prediction of TEGMNet.

*2) Objective function:* In the training process, we adopt the mean absolute error ($L_1$-loss) and the objective function is expressed as follows

$$\mathcal{O}(y, \hat{y}) = \frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{D} |(\hat{y}_i^j - y_i^j)| \tag{12}$$

where $y$ is the ground truth, $\hat{y}$ is the forecasting value, $N$ is the number of training samples and $D$ is the dimension of target data. All neural models are trained using the Adam optimizer.

## V. Experiment

We conducted extensive experiments with 8 methods (including our TEGMNet) on 4 benchmark datasets for multivariate time series forecasting. To verify the advantages of our new methods in reducing the noise interference in the data, we manually add White Gaussian noise with different SNR(signal-noise-ratio) to the original data. Generally speaking, SNR can be understood as the ratio of signal to noise. The smaller the SNR, the greater the proportion of noise in a digital signal; on the contrary, the larger the SNR, the smaller the proportion of noise in a digital signal.

## A. Metrics

We use two conventional evaluation metrics, namely Root mean Square Error (RSE) and empirical CORRelation coefficient (CORR) which are defined as:

$$RSE = \frac{\sqrt{\sum_{(i,t)\in\Omega_{Test}}(Y_{it}-\hat{Y}_{it})^2}}{\sqrt{\sum_{(i,t)\in\Omega_{Test}}(Y_{it}-mean(Y))^2}} \tag{13}$$

$$CORR = \frac{1}{N}\sum_{i=1}^{n}\frac{\sum_t(Y_{it}-mean(Y_i))(\hat{Y}_{it}-mean(\hat{Y}_i))}{\sqrt{\sum_t(Y_{it}-mean(Y_i))^2(\hat{Y}_{it}-mean(\hat{Y}_i))^2}} \tag{14}$$

where $\Omega_{Test}$ means the test set for evaluation, $Y, \hat{Y} \in R^{d\times T}$ are ground true signals and system prediction signals, respectively. For RSE, a lower value is better, while for CORR a higher value is better.

## B. Datasets

We perform experiments on 2 real traffic datasets and 2 datasets in other fields. Table I summarizes some key statistics of these datasets.

TABLE I
DATASET STATISTICS, WHERE LENGTH IS LENGTH OF TIME SERIES, D IS NUMBER OF VARIABLES, UNIT IS THE SAMPLE RATE.

| Datasets | Length | D | Unit |
|---|---|---|---|
| Traffic-Calif [1] | $17,544$ | $862$ | $1h$ |
| Traffic-228 [22] | $12,672$ | $228$ | $5min$ |
| Solar-Energy [3] | $52,560$ | $137$ | $10min$ |
| Electricity [2] | $26,304$ | $321$ | $1h$ |

We perform experiments not only on the original datasets but also on the datasets with different scales of white Gaussian noise (SNR = 15, 30 dB). SNR is the ratio of the power of signal to the power of noise. Its unit is usually dB, and its value is 10 times the log signal to noise power ratio:

$$SNR(dB) = 10\log_{10}(\frac{P_{signal}}{P_{noise}}), \\ P_{signal} = \frac{1}{n}\sum_{k=1}^{n}{s_k}^2, P_{noise} = \frac{P_{signal}}{10^{\frac{SNR}{10}}} \tag{15}$$

where $P_{signal}$ is the power of signal, $S = \{s_1, s_2, \cdots, s_n\}$ is the original dataset signal and $P_{noise}$ is the power of noise. After getting the power of the noise, we can form a sequence of data with the noise following the standard Gaussian distribution (mean value is 0, standard deviation is 1) (same as the signal length), and then we can get the desired Gaussian noise through conversion.

## C. Experiment Results

**Results-I: Time Series Prediction** To demonstrate the effectiveness of the TEGMNet, we compare our TEGMNet model with the 7 benchmark methods mentioned above on the datasets in all the metrics. Table II shows the prediction results. The best result for each setting (data, SNR, metric) is highlighted in boldface in this table. From the table, we can see proposed TEGMNet model gives the best results in most situations. Obviously, the TEGMNet models show greater advantages in the datasets with periodic patterns, especially in the presence of large noise disturbances. Furthermore, TEGMNet outperforms the most in RSE and CORR metric when the SNR is 15 (high noise level), demonstrating the effectiveness of the framework design for combating against the noise disturbance and capturing the complex long and short term repetitive patterns.

**Results-II: Ablation Study** In order to prove the effectiveness of our model design, we conducted a careful ablation study with the result shown in Table III. Specifically, we remove some components in our TEGMNet framework at one time and implement the rest with the same environment and datasets. We name TEGMNet without different parts as follows.

- **TEGMNet-withoutMT**: The MDT and TFNN components are removed, convolution operation is performed on the time series matrix by 2-D convolution kernel.
- **TEGMNet-withoutT**: The TFNN component in our proposed model is removed. After the MDT operation is finished, the data are input into the convolution component.
- **TEGMNet-withoutC**: The convolution component in our proposed model is removed. After the TFNN operation, the data are then input into the LSTM component.
- **TEGMNet-withoutMemo**: The embedding output of one short-term historical data block is treated as the prediction result.

Through this part of the ablation study experiment, it can be further drawn that TEGMNet shows better prediction performance than baselines under most conditions and all the components in the model play a corresponding role.

## VI. CONCLUSION

This paper proposes TEGMNet, a Tensor projection Embedding with Gated Memory Network for multivariate time series forecasting. TEGMNet includes two modules, Embedding with Tensor Projection and a Time Series Network with Gated Memory. To extract fine-grained temporal and spatial features in each block and effectively reduce the noise interference, we design an embedding module that extracts the embedding from tensorized data projection using several techniques, MDT, TFNN, Convolution, and LSTM. We also design a long-term feature extraction module based on the structure of gated end-to-end memory network, which can extract complex long-term dependencies with dynamic-period patterns. Extensive experiments demonstrate that our TEGMNet is a robust forecasting algorithm, which can accurately predict the future data trend

TABLE II
RESULTS SUMMARY OF ALL METHODS ON 4 DATASETS.

| Noise | | SNR=30 | | | | SNR=15 | | | | No Extra Noise | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Datasets | | Traffic-Calif | Traffic-228 | Solar-Energy | Electricity | Traffic-Calif | Traffic-228 | Solar-Energy | Electricity | Traffic-Calif | Traffic-228 | Solar-Energy | Electricity |
| VAR [5] | RSE | 0.6079 | 0.5991 | 0.2812 | 0.1248 | 0.6342 | 0.6218 | 0.3056 | 0.1306 | 0.5897 | 0.5875 | 0.2542 | 0.1723 |
| | CORR | 0.7798 | 0.7569 | 0.9261 | 0.8703 | 0.7678 | 0.7503 | 0.9113 | 0.8639 | 0.7841 | 0.7602 | 0.9421 | 0.8821 |
| VARMA [13] | RSE | 0.6043 | 0.5945 | 0.2765 | 0.1236 | 0.6323 | 0.6176 | 0.2958 | 0.1392 | 0.5849 | 0.5805 | 0.2481 | 0.1701 |
| | CORR | 0.7815 | 0.7483 | 0.9431 | 0.8723 | 0.7681 | 0.7580 | 0.9156 | 0.8714 | 0.7841 | 0.7648 | 0.9465 | 0.8303 |
| RNN-GRU [10] | RSE | 0.5105 | 0.4957 | 0.2588 | 0.1435 | 0.5163 | 0.5036 | 0.2485 | 0.1491 | 0.4963 | 0.4801 | 0.2368 | 0.1382 |
| | CORR | 0.8587 | 0.8122 | 0.9673 | 0.8717 | 0.8439 | 0.8143 | 0.9601 | 0.8702 | 0.8631 | 0.8421 | 0.9729 | 0.8728 |
| RNN-LSTM [11] | RSE | 0.5074 | 0.5079 | 0.2491 | 0.1458 | 0.5121 | 0.5196 | 0.2507 | 0.1531 | 0.4974 | 0.5079 | 0.2311 | 0.1431 |
| | CORR | 0.8586 | 0.8067 | 0.9727 | 0.8724 | 0.8398 | 0.8091 | 0.9689 | 0.8689 | 0.8597 | 0.8067 | 0.9787 | 0.8711 |
| DSANet [12] | RSE | 0.4987 | 0.4818 | 0.2668 | 0.1221 | 0.5078 | 0.4976 | 0.2371 | 0.1285 | 0.4868 | 0.4784 | 0.1984 | 0.1197 |
| | CORR | 0.8508 | 0.8201 | 0.9655 | 0.8687 | 0.8543 | 0.8262 | 0.9631 | 0.8675 | 0.8491 | 0.8483 | 0.9693 | 0.8669 |
| MTNet [6] | RSE | **0.4907** | 0.4821 | 0.2389 | 0.1198 | 0.5067 | 0.4937 | 0.2396 | 0.1216 | 0.4897 | 0.4797 | 0.1823 | 0.1182 |
| | CORR | 0.8551 | 0.8122 | 0.9743 | 0.8742 | 0.8509 | 8486 | 0.9638 | 0.8698 | 0.8631 | 0.8623 | **0.9703** | 0.8721 |
| LSTNet [15] | RSE | 0.4936 | 0.4827 | 0.2476 | 0.1093 | 0.5034 | 0.4952 | 0.2371 | 0.1124 | 0.4941 | 0.4791 | 0.1829 | **0.0931** |
| | CORR | 0.8613 | 0.8171 | 0.9701 | 0.8812 | 0.8512 | 0.8493 | 0.9667 | 0.8779 | 0.8628 | 0.8574 | 0.9687 | 0.8807 |
| TEGMNet | RSE | 0.4915 | **0.4804** | **0.2231** | **0.0912** | **0.4931** | **0.4894** | **0.2315** | **0.0963** | **0.4864** | **0.4784** | **0.1821** | 0.1087 |
| | CORR | **0.8675** | **0.8203** | **0.9762** | **0.8917** | **0.8565** | **0.8572** | **0.9653** | **0.8892** | **0.8652** | **0.8642** | 0.9641 | **0.8942** |

TABLE III
RESULTS SUMMARY OF THE ABLATION STUDY WAYS ON 4 DATASETS.

| Noise | | SNR=30 | | | | SNR=15 | | | | No Extra Noise | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Datasets | | Traffic-Calif | Traffic-228 | Solar-Energy | Electricity | Traffic-Calif | Traffic-228 | Solar-Energy | Electricity | Traffic-Calif | Traffic-228 | Solar-Energy | Electricity |
| withoutMT | RSE | 0.4925 | 0.4841 | 0.2489 | 0.1124 | 0.5086 | 0.4978 | 0.2389 | 0.1145 | 0.4978 | 0.4821 | 0.1858 | 0.1107 |
| | CORR | 0.8625 | 0.8171 | 0.9691 | 0.8824 | 0.8521 | 0.8467 | 0.9652 | 0.8781 | 0.8619 | 0.8582 | 0.9682 | 0.8779 |
| withoutT | RSE | 0.4922 | 0.4825 | 0.2502 | 0.1089 | 0.5027 | 0.4939 | 0.2369 | 0.1128 | 0.4931 | 0.4799 | 0.1856 | 0.1085 |
| | CORR | 0.8621 | 0.8182 | 0.9713 | 0.8826 | 0.8527 | 0.8501 | 0.9661 | 0.8749 | 0.8641 | 0.8570 | 0.9667 | 0.8757 |
| withoutC | RSE | 0.4949 | 0.4875 | 0.2302 | 0.1106 | 0.5124 | 0.4937 | 0.2289 | 0.1106 | 0.4956 | 0.4821 | 0.1805 | **0.1076** |
| | CORR | 0.8701 | 0.8211 | 0.9681 | 0.8842 | 0.8783 | 0.8512 | 0.9465 | 0.8579 | 0.8557 | 0.8598 | 0.9601 | 0.8721 |
| withoutMemo | RSE | 0.5029 | 0.4921 | 0.2571 | 0.1141 | 0.5058 | 0.4978 | 0.2460 | 0.1162 | 0.4989 | 0.4821 | 0.1867 | 0.1109 |
| | CORR | 0.8578 | 0.8213 | 0.9653 | 0.8779 | 0.8491 | 0.8479 | 0.9679 | 0.8782 | 0.8625 | 0.8567 | **0.9693** | 0.8812 |
| TEGMNet | RSE | 0.4915 | **0.4804** | **0.2231** | **0.0912** | **0.4931** | **0.4894** | **0.2315** | **0.0963** | **0.4864** | **0.4784** | **0.1801** | 0.1087 |
| | CORR | **0.8675** | **0.8203** | **0.9762** | **0.8917** | **0.8565** | **0.8572** | **0.9653** | **0.8892** | **0.8652** | **0.8642** | 0.9641 | **0.8942** |

even though there exists unavoidable noise in the multivariate time series.

## VII. ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 62025201, Grant 61972144, and Grant 61976087, in part by the U.S. NSF under Grant ECCS 2030063, 1731238, CCF 2007313, in part by the CERNET Innovation Project under Grant NGII20190118.

## REFERENCES

[1] http://pems.dot.ca.gov.

[2] https://archive.ics.uci.edu/ml/datasets/electricityloaddiagrams20112014.

[3] http://www.nrel.gov/grid/solar-power-data.html.

[4] Y. Bengio. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans Neural Netw*, 5, 2002.

[5] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., USA, 1990.

[6] Yen-Yu Chang, Fan-Yun Sun, Yueh-Hua Wu, and Shou-De Lin. A memory-network based solution for multivariate time-series forecasting, 2018.

[7] J. Chien and Y. Bao. Tensor-factorized neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1998–2011, 2018.

[8] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[9] J. Connor, L. Atlas, and D. Martin. Recurrent networks and narma modeling. *Advances in Neural Information Processing Systems*, 4:301–308, 1992.

[10] R. Fu, Z. Zhang, and L. Li. Using lstm and gru neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 324–328, 2016.

[11] S Hochreiter and J Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[12] Siteng Huang, Donglin Wang, Xuehan Wu, and Ao Tang. Dsanet: Dual self-attention network for multivariate time series forecasting. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 2129–2132, New York, NY, USA, 2019. Association for Computing Machinery.

[13] E. Isufi, A. Loukas, N. Perraudin, and G. Leus. Forecasting time series with varma recursions on graphs. *IEEE Transactions on Signal Processing*, 67(18):4870–4885, 2019.

[14] P. Jing, Y. Su, X. Jin, and C. Zhang. High-order temporal correlation model learning for time-series prediction. *IEEE Transactions on Cybernetics*, 49(6):2385–2397, 2019.

[15] 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*, SIGIR '18, page 95–104, New York, NY, USA, 2018. Association for Computing Machinery.

[16] Jianze Li, Xiaoping Zhang, and Tuan Tran. Point cloud denoising based on tensor tucker decomposition. pages 4375–4379, 2019.

[17] Fei Liu and Julien Perez. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 2017.

[18] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions*, 371(1984):20110550, 2012.

[19] Qiquan Shi, Jiaming Yin, Jiajun Cai, Andrzej Cichocki, and Jia Zeng. Block hankel tensor arima for multiple short time series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(4):5758–5766, 2020.

[20] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. *Computer Science*, 2015.

[21] T. Yokota, B. Erem, S. Guler, S. K. Warfield, and H. Hontani. Missing slice recovery for tensors using a low-rank model in embedded space. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8251–8259, 2018.

[22] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 3634–3640. AAAI Press, 2018.

[23] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *the 24th ACM SIGKDD International Conference*, 2018.

[24] Xinyuan Zhang, Jie Peng, Man Xu, Wei Yang, Zhe Zhang, Hua Guo, Wufan Chen, Qianjin Feng, Ed X. Wu, and Yanqiu Feng. Denoise diffusion-weighted images using higher-order singular value decomposition. *NeuroImage*, 156:128–145, 2017.