MARINA: An MLP-Attention Model for Multivariate Time-Series Analysis

Jiandong Xie
Huawei Cloud Database Innovation Lab
Chengdu, China
xiejiandong@huawei.com

Yue Cui
The Hong Kong University of Science and Technology
Hong Kong, China
ycuias@cse.ust.hk

Feiteng Huang
Huawei Cloud Database Innovation Lab
Chengdu, China
xiejiandong@huawei.com

Chao Liu
Huawei Cloud Database Innovation Lab
Chengdu, China
liuchao171@huawei.com

Kai Zheng
University of Electronic Science and Technology of China
Chengdu, China
zhengkai@uestc.edu.cn

ABSTRACT

The proliferation of real-time monitoring applications such as Artificial Intelligence for IT Operations (AIOps) and the Internet of Things (IoT) has led to the generation of a vast amount of time-series data. To extract the underlying value of the data, both the industry and the academia are in dire need of efficient and effective methods for time-series analysis. To this end, in this paper, we propose a Multi-layer perceptron (MLP)-attention based multivariate time-series analysis model MARINA. MARINA is designed to simultaneously learn the temporal and spatial correlations among multivariate time-series. Also, the model is versatile in that it is suitable for major time-series analysis tasks such as forecasting and anomaly detection. Through extensive comparisons with the representative multivariate time-series forecasting and anomaly detection algorithms, MARINA is shown to achieve state-of-the-art (SOTA) performance in both forecasting and anomaly detection tasks.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

KEYWORDS

Anomaly detection, forecasting, time-series analysis

ACM Reference Format:
Jiandong Xie, Yue Cui, Feiteng Huang, Chao Liu, and Kai Zheng. 2022. MARINA: An MLP-Attention Model for Multivariate Time-Series Analysis. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557386

CIKM '22, October 17–21, 2022, Atlanta, GA, USA.

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557386

1 INTRODUCTION

The last decade have witness time-series data becoming a ubiquitous data form due to the proliferation of real-time monitoring applications. In cloud service scenarios, for example, minute by minute or even second by second, a huge amount of virtual machine monitoring data is generated and stored in time-series databases (TSDBs) for further processing and analysis. In most cases, times-series data is high in volume and low in information density making it impossible to fully extract its underlying value by pure human supervision. As a consequence, the research interest on effective and efficient time-series analysis methods is ever increasing in recent years.

In fact, the industry is the major driver for the development of time-series analysis methods. In industry, the common application scenarios for time-series analysis lies in anomaly detection [1, 6, 22] and forecasting [8, 36, 40]. From the processing capability perspective, time-series analysis algorithms are generally required to have multi-variate capability, i.e. to be able to process multiple time-series in parallel so that high processing speed can be achieved and common features among different time-series can be extracted.

In this paper, to accommodate the need for effective time-series analysis methods, we propose a novel multi-layer perceptron (MLP)-attention based neural network model, MARINA. Such a model is capable of multivariate forecasting and forecasting based anomaly detection.

In multivariate time-series analysis, the spatial and temporal correlations within time-series are the most critical knowledge to learn. Specifically, temporal correlation stands for the dependency between historical points and future points in a time-series that is concealed in the periodicity and tendency of the data. Spatial correlation denotes the similarity between different time-series. In Fig.1 we visualize three time-series which represents the hourly traffic occupancy ratio in different roads from the Traffic dataset[1]. Due to the nature of human activity, the chart exhibits daily and weekly periodicity. The phenomenon that the traffic occupancy rate at a certain time is similar to the data of a day ago and a week ago is known as the temporal correlation. Also, in both the

[1]The Traffic dataset contains the road occupancy rates measured by 862 sensors in San Francisco Bay area and is available on http://pems.dot.ca.gov

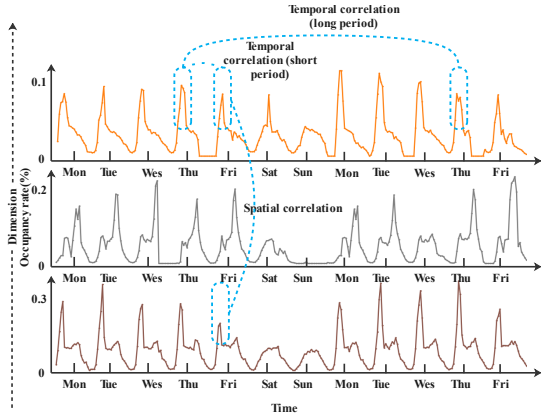

Figure 1: The hourly occupancy rate of the Traffic dataset

first and the third dimension, a clear tendency of a morning traffic peak is demonstrated and such similarity is known as the spatial correlation.

To fully capture the two types of correlations as well as to adjust for different output shape, MARINA is designed with three modules, i.e., the temporal module, the spatial module, and the output reshaping module. For temporal correlation learning, due to the causal and sequential characteristics of time-series data, many existing researches such as the famous Informer and Autoformer, etc., [20, 23, 35, 36] resort to models such as transformer [33], temporal convolutional network (TCN) [4] as well as the recurrent neural network (RNN) [30] in analogy to natural language processing (NLP) tasks. However, whether the NLP models are really suitable for time-series analysis tasks are still questionable given the unsatisfying performance of pure RNN models reported by [25]. For the temporal module of MARINA, we adopt the MLP network with residual connections considering MLP's light-weightedness and significant period learning performance as reported in [25]. The spatial module resorts to graph neural networks to learn the similarity among different time-series and utilizes such correlation to enhance the performance of analysis. The core procedure of graph learning is message passing through a given graph which describes the correlations among the nodes. In the time-series context, a node is usually a time-series. However, for time-series analysis tasks, the topological structure describing spatial correlation is rarely available. In this paper, inspired by the graph attention network [34], we use self-attention structure for graph learning which is independent of predefined graph input. At last, an MLP output module serves to adjust the output into the desired shape.

The major contributions of this paper are summarized as follows:

- We propose a versatile time-series analysis model MARINA that is suitable for major time-series analysis tasks including forecasting and anomaly detection. Compared with the family of NLP models such as RNN, TCN and transformer, MARINA demonstrates the superiority of MLP-attention structure for spatio-temporal correlation learning.
- We propose a novel dynamic normalization method for data preprocessing to solve the concept drift problem between the training set and the test set which undermines the forecasting performance.
- We conduct extensive experiments on long sequence forecasting and anomaly detection. MARINA is able to outperform the current SOTA models with reasonably big margin in all tasks. In our ablation study, the effectiveness of the proposed dynamic normalization method is also verified.
- MARINA is shown to be significantly more efficient in training and testing than other baseline models, demonstrating its high eligibility for industrial level applications such as AIOps and IoT.

2 RELATED WORK

2.1 Time-series Forecasting

Time-series forecasting algorithms can be classified into four categories: non-learning, statistical learning, deep learning and hybrid methods. Typical non-learning algorithms are exponential smoothing [9], Holt-Winters [5] and STL [7]. Prominent statistical learning methods include auto-regression (AR) [13] auto-regression integrated moving average (ARIMA) [16] and Prophet [32]. Among them, Prophet is a Bayesian learning based algorithm that models the times-series as the summation of seasonal, trend and holiday components. Owing to its high explainability and simplicity, Prophet is well accepted in industry. Due to the flexibility and scalability of neural networks, deep learning based forecasting has drawn a lot of research interests in recent years. It is believed that the key to time-series forecasting is to learn the temporal correlation between history and future points as well as the spatial correlation between different dimensions [11, 36, 38]. For spatial correlation, various types of graph convolution modules are adopted [36]. For temporal correlation, most researches tend to borrow the already prevailing modules in NLP tasks such as RNN, transformer and TCN due to the analogy between time-series forecasting and machine translation [20, 23, 40]. However, a recent research also show that simple networks such as MLP can well learn the temporal correlation and obtain good forecasting results [27]. At last, to simultaneously harness the simplicity and robustness of non-learning algorithms as well as to exploit the flexibility of deep learning algorithms, hybrid algorithms such as exponential smoothing-RNN (ES-RNN) [2] are proposed and demonstrate good performance as reported in [25].

2.2 Anomaly Detection

Anomaly detection can also be solved by non-learning algorithms such as exponential smoothing and Holt-Winters [5, 9]. Yet, regardless of their simplicity, non-learning algorithms are generally susceptible to false alarms. For learning based algorithms, if the underlying anomalies are labeled in the training data, supervised models such as EGADS [21] and Opprentice [24] can be adopted. While with label assisted training, supervised models can usually achieve good performance, anomaly labeling requires intensive human resources making it unaffordable in most cases. As a result, unsupervised anomaly detection methods are extensively studied

[2] https://github.com/autonlab/esrnn.git

in recent years which are generally categorized into reconstruction based and forecasting based anomaly detectors. Reconstruction based detectors need to reconstruct the whole network input and are mostly different variants of the basic auto-encoder structure [31, 37]. On the auto-encoder basis, some resort to extra GAN structures [2, 6, 39] or variational learning [17] to enhance reconstruction accuracy. On the other hand, according to the principle that anomalous outliers are those breaking normal periodicity and tendency patterns, therefore unpredictable, forecasting based algorithms are developed for anomaly detection [15].

3 PROBLEM STATEMENT

In this section, we mathematically formulate the problem of multivariate time-series forecasting and anomaly detection.

3.1 Time-Series Forecasting

In time-series forecasting tasks, the input is a multivariate time-series window of length ω: $\mathcal{X}_\omega(t) = \{X(t-\omega+1), ..., X(t)\}$ where $X(i) = [x_1(i), .., x_D(i)] \in \mathbb{R}^D, i \in [t-\omega+1, ..., t]$. D is the number of dimensions in the time-series data. In the AIOps scenario, for example, different dimensions corresponds to the monitoring metrics such as CPU utilization and memory utilization. In sequence forecasting tasks [40], the target is to predict a future sequence with horizon η i.e., $\mathcal{Y}_\eta(t) = \{Y(t+1), ..., Y(t+\eta)\}$, $Y(i) \in \mathbb{R}^D$.

3.2 Time-Series Anomaly Detection

In the anomaly detection tasks, the typical goal is, given a sequence $\mathcal{Z}_{\omega+\eta}$ consisting of $\mathcal{X}_\omega(t) = \{X(t-\omega+1), ..., X(t)\}$ and $\mathcal{Y}_\eta(t) = \{Y(t+1), ..., Y(t+\eta)\}$, determine whether there are anomalies in the target sequence $\mathcal{Y}_\eta(t)$. In the deep learning framework, there are usually two angles to address the anomaly detection problem i.e., by reconstruction and by forecasting. In reconstruction based anomaly detection, the whole sequence $\mathcal{Z}_{\omega+\eta}$ is put into an auto-encoder to obtain a reconstructed sequence. Then, the relative error, for example Frobenius norm error, between the target sequence $\mathcal{Y}_\eta(t)$ and its corresponding part $\hat{\mathcal{Y}}_\eta(t)$ in the reconstruction $\hat{\mathcal{Z}}_{\omega+\eta}(t)$ is taken as the anomaly score and compared with a predetermined threshold to identify an anomaly as

$$\|\mathcal{Y}_\eta(t) - \hat{\mathcal{Y}}_\eta(t)\|_F \underset{H_1}{\overset{H_0}{\lesseqgtr}} \gamma, \tag{1}$$

where we use H_0 and H_1 to denote the normal and abnormal states respectively, F denotes the Frobenius norm and γ stands for the threshold. In forecasting based anomaly detection, $\mathcal{X}_\omega(t)$ is used to make a forecast $\hat{\mathcal{Y}}_\eta(t)$ of the target sequence $\mathcal{Y}_\eta(t)$ and the forecasting error is used to determine anomaly in the same way as in Eq. (1). The rationale behind forecasting based anomaly detection is that, normal data follows the usual pattern of periodicity and tendency while the anomalies are outliers that violate the normal patterns and therefore unpredictable.

In order to address the forecasting and anomaly detection problems with a unified model, in this paper, we address the anomaly detection problem from the forecasting perspective.

4 MODEL DESIGN

In the following, we first introduce our dynamically normalization method which dynamically normalizes the test set. Then, we describe the three modules in MARINA.

4.1 Data Normalization

In the data preprocessing procedure, the training, validation and the test sets are first normalized before sent into the network. In our work we adopt standard normalization which normalizes the input data's mean and variance to zero and one respectively. In most existing work on time series analysis models, the normalization is performed in a static fashion in which the mean and variance of the training set is first used to normalize itself and the validation set, then in the testing phase, they are also used to normalize the whole test set. In fact, there are two drawbacks within this static normalization methods. First, in time series analysis, concept drift happens frequently. In many data sets, the value range of the test set can vary significantly from the training set as shown in Fig. 3. Still normalizing the test set with the training set can severely undermine the forecasting performance. Second, unlike other machine learning tasks such as image classification, in which different inputs are not ordered, time series data are naturally ordered and when forecasting a segment of data after a certain time stamp, the entire data before that time stamp should be considered known and utilized as much as possible. Therefore, only using the training set to normalize the test set does not fully utilize the information on hand.

To overcome the aforementioned drawbacks, we propose a dynamic normalization method. As shown in Fig. 4, in dynamic normalization, for the training set, we still use its mean and variance to normalize itself. In the test set, we use a cache to maintain the current mean and variance. When new input data arrives, we first use the new data to exponentially update the mean and variance stored in cache as show in eq. (2) where x_i is the ith forecasting input, α is the weight coefficient and μ_i and σ_i are statistics after ith update and they are initiated by the training set.

$$\begin{aligned} \mu_0 &= \mu \\ \sigma_0 &= \sigma \\ \mu_i &= (1-\alpha)\mu_{i-1} + \alpha E(x_i) \\ \sigma_i^2 &= (1-\alpha)\sigma_{i-1}^2 + \alpha(E(x_i^2) - E(x_i)^2) \end{aligned} \tag{2}$$

Then, we use them to normalize the input. In such a manner, the forecasting data is normalized by the adjacent latest data and therefore, the negative effect of data concept drift can be alleviated.

4.2 Temporal Correlation Learning Module

Temporal correlation stands for the dependency between historical points and future points. Such dependency is often reflected in the periodicity and tendency of the data. To capture such temporal correlation, many resort to recurrent networks, TCN and transformer which are prevailing in NLP researches as NLP's tasks also involve learning the temporal correlation from ordered data. However, the intrinsic patterns of periodicity and tendency in time-series waveform are not present in word vectors. In recent researches, the power of simple MLP structures in time-series forecasting is

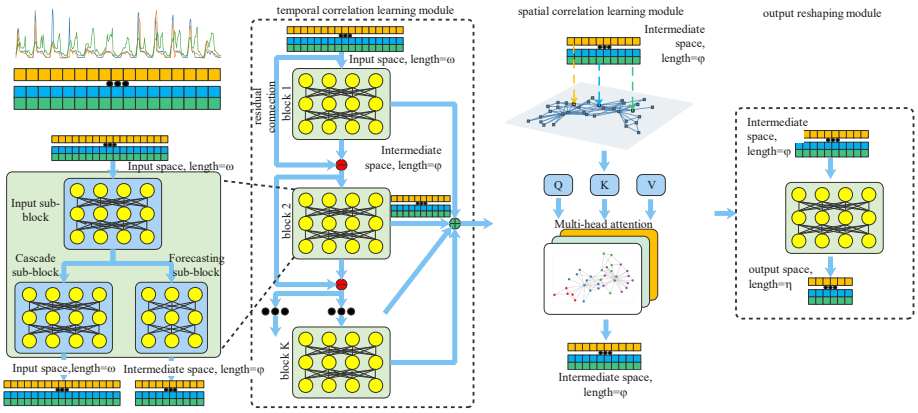

Figure 2: MARINA Overview

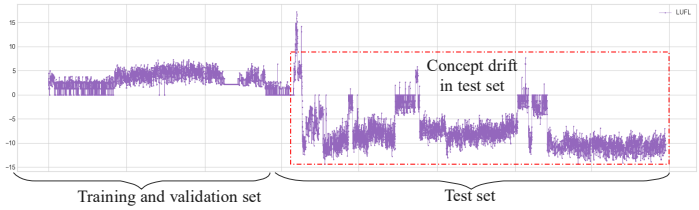

Figure 3: Concept drift example in ETTh2 data set

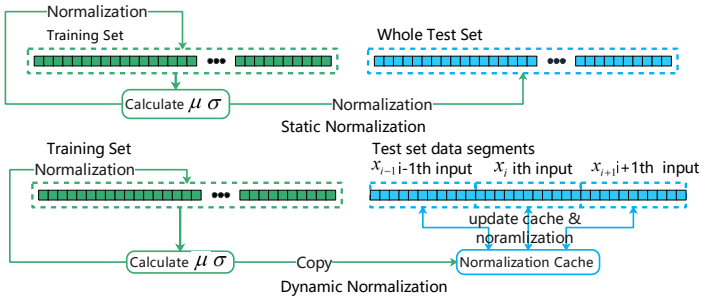

Figure 4: Static and dynamic normalization methods

demonstrated. As reported in [27], an MLP network aided by residual connection achieves SOTA performance in M4 forecasting tasks [25]. Considering the MLP structure's effectiveness, simplicity and efficiency, we adopt the MLP structure for temporal correlation learning.

As illustrated in Fig. 2, the temporal correlation module works on the time dimension and the input is directly the historical time-series. The temporal module consists of K MLP blocks that are connected through residual connections enabling the network to be deep enough to learn the complex temporal patterns while avoiding the gradient vanishing or exploding problems [14]. Each MLP block contains an input sub-block, a cascade sub-block to prepare input for the next block and a forecasting sub-block to perform predictions. Specifically, denoting the original multivariate time-series as $\mathcal{X} \in \mathbb{R}^{D\times\omega}$, the input, cascade output and forecasting output of the kth block respectively as $\mathcal{X}_{k,I}^{temp}$, $\mathcal{X}_{k,C}^{temp}$ and $\mathcal{X}_{k,F}^{temp}$ and the output of the temporal learning module as $\mathcal{X}_{O}^{temp} \in \mathbb{R}^{D\times\varphi}$ where φ is the output length of the temporal module, we have the following equations:

$$\mathcal{X}_{1,I}^{temp} = \mathcal{X}, \tag{3}$$

$$\mathcal{X}_{k+1,I}^{temp} = \mathcal{X}_{k,I}^{temp} - \mathcal{X}_{k,C}^{temp}, \tag{4}$$

$$\mathcal{X}_{O}^{temp} = \sum_{k=1}^{K} \mathcal{X}_{k,F}^{temp}. \tag{5}$$

4.3 Spatial Correlation Learning Module

The temporal correlation learning module works on each of the time-series separatively and therefore does not fully exploit the similarities among them. In existing literatures, the major tool to harness the similarities among multiple objects is the graph neural network (GNN) [28], where the whole dataset is viewed as a graph $G = (V, E)$ with V and E respectively denoting the set of vertexes and edges. The graph can be described numerically by an adjacent matrix $A \in \mathbb{R}^{|V|\times|V|}$ with A_{ij} representing the weight of the edge between node i and node j. In the multivariate time-series analysis scenario, the vertexes correspond to each time-series dimensions and the edge weights are the internal correlations between dimensions. Given the full graph information, graph learning can be performed by graph convolution [18] or graph attention [34] which are essentially message passing among vertexes according to the connectivity and weights defined in the adjacency matrix.

It is worth noting that, unlike tasks with abundant extra information describing vertex dependency such as literature classification where clear citation chains can be used to construct the adjacency matrix, in multivariate time-series analysis, the internal dependency among time dimensions can hardly be accurately quantified. Therefore, it is difficult to accurately construct an adjacency matrix. Inspired by the power of self-attention [34] as well as verified by

Algorithm 1 MARINA Off-line Training and Online Forecasting/Detection

Training Process

Input: Training set $\{\mathcal{X}_\omega(0), ..., \mathcal{X}_\omega(T)\}$, $\{\mathcal{Y}_\eta(0), ..., \mathcal{Y}_\eta(T)\}$, batch size β and other hyperparameters
Output: Model parameters
for randomly sampled batch $\{\mathcal{X}_\omega(t_0), ..., \mathcal{X}_\omega(t_\beta)\}$ **do**
  Temporal Correlation Learning
  Obtain temporal module output $\mathcal{X}_O^{temp}(t_i)$, $i \in \{0, ..., \beta\}$ according to Eq. (3), (4) and (5)
  Spatial Correlation Learning
  Obtain spatial module output $\mathcal{X}_O^{Spat}(t_i)$, $i \in \{0, ..., \beta\}$ according to Eq. (6), (7) and (8)
  Output Shaping
  $\hat{\mathcal{Y}}_\eta(t_i) = \text{MLP}(\mathcal{X}_O^{Spat}(t_i))$
  Perform stochastic gradient descent on model parameters
end for

Online Forecasting

Input: Historical sequence $\mathcal{X}_\omega(t)$
Output: Forecasting sequence $\hat{\mathcal{Y}}_\eta(t)$
$\hat{\mathcal{Y}}_\eta(t) = \text{MARINA}(\mathcal{X}_\omega(t))$

Online Anomaly Detection

Input: Historical sequence $\mathcal{X}_\omega(t)$, target sequence $\mathcal{Y}_\eta(t)$, detection threshold γ
Output: Anomaly label of the target sequence
$\hat{\mathcal{Y}}_\eta(t) = \text{MARINA}(\mathcal{X}_\omega(t))$
Obtain anomaly label according to Eq. (1)

our extensive simulations, we use the multi-head self-attention structure which doesn't rely on adjacency matrix information to conduct graph learning.

Specifically, each row in the output matrix of the temporal module $\mathcal{X}_O^{temp} \in \mathbb{R}^{D\times\varphi}$ is viewed as a vertex and the massage passing process is

$$Q = K = V = \mathcal{X}_O^{temp}, \tag{6}$$

$$\mathcal{X}_{Int}^{spat} = MultiHeadAttention(Q, K, V), \tag{7}$$

$$\mathcal{X}_O^{spat} = FFN(\mathcal{X}_{Int}^{spat}), \tag{8}$$

where $\mathcal{X}_{Int}^{spat} \in \mathbb{R}^{D\times\varphi}$ and $\mathcal{X}_O^{spat} \in \mathbb{R}^{D\times\varphi}$ are respectively the intermediate value and the output of the spatial module. Q, K, V are respectively the query, key, value in the attention process. The attention and position-wise feed forward operations $MultiHeadAttention(\bullet)$ and $FFN(\bullet)$ are as described in [33].

4.4 Output Reshaping Module

At the output part of the neuron network, we adopt a single layer MLP to regulate the output shape.

As we have introduced the full structure of MARINA, the off-line training and online forecasting/detection processes are summarized in Algorithm. 1.

4.5 Complexity Analysis and Discussion

In this subsection, we analyze the forward propagation time complexity of different modules in MARINA.

We assume that the length of the input time-series is ω and there are D dimensions. On the temporal learning module, we suppose that there are K blocks. For each block, we assume that there are respectively l_{in}, l_{ca}, l_{fo} layers for the input, cascade and forecasting sub-blocks and the non-output layers all have ω neurons. Based on such a setting, the time complexity of the temporal module is $O((l_{in} + l_{ca} + l_{fo} - 1)\omega^2 D + \omega\varphi D)$. By viewing l_{in}, l_{ca}, l_{fo} as constants, the time complexity is $O(\omega^2 D + \omega\varphi D)$.

The self-attention operation in spatial learning module is composed of four sub-operations: multi-head reshaping, multi-head attention, multi-head concatenation and point-wise feed forward. Assuming that there are h heads, in each head, the query, key, value dimensions are identical: $d_v = d_k = d_q$, and hidden layer neuron number of the feed forward network is φ_{hid} the time complexity can be calculated as $O(hD\varphi d_v + 2hD^2 d_v + h^2 D d_v \varphi + 2D\varphi\varphi_{hid})$. By assuming that h is a constant and, d_v and φ_{hid} are proportional to φ, the time complexity is $O(D\varphi^2 + D^2\varphi)$.

The time complexity of the output module is $O(D\varphi\eta)$. The time complexity of all modules are summarized in Table 1.

Table 1: MARINA time complexity

Module	Time Complexity
Temporal Learning Module	$O(\omega^2 D + \omega\varphi D)$
Spatial Learning Module	$O(D\varphi^2 + D^2\varphi)$
Output Reshaping Module	$O(D\varphi\eta)$

Unlike many spatial-temporal forecasting models such as [29] and [12] in which the temporal and spatial modules are deeply coupled, the temporal and spatial modules of MARINA are decoupled and the spatial module can be easily turned off with a switch. The spatial learning module requires strong correlation among different time-series to gain observable benefit. Also, it can also be observed that when the number of dimensions is large, the spatial module become a significant source of complexity due to the D^2 term. Therefore, with respect to data with no strong spatial correlation, the benefit of such a decoupled design is that the spatial learning module can be turned off to reduce time complexity without sacrificing the accuracy.

5 EXPERIMENTS AND RESULTS

In this section, to prove the effectiveness of the MARINA model, extensive experiments on the two time-series analysis tasks, i.e. long sequence forecasting and anomaly detection with in total eight datasets are conducted. All experiments are done on a an Intel (R) Xeon(R) Gold 6278C CPU @ 2.60GHz and two Tesla T4 GPUs.

5.1 Long Sequence Forecasting

In long sequence forecasting as studied in [40], the objective is to use a historical sequence $\mathcal{X}_\omega(t)$ of length ω to predict an η step long future sequence $\mathcal{Y}_\eta(t)$.

5.1.1 Datasets and Metrics. For long sequence forecasting tasks, we use the following four datasets, i.e., ETTh1, ETTh2, ETTm1[3] and Electricity. The detailed statistics of the datasets are listed in Table 2

[3]https://github.com/zhouhaoyi/ETDataset.

Table 2: Statistics of Datasets for long sequence forecasting

Dataset	Samples	Dimension	Sample rate
Electricity	26304	321	1 hour
ETTh1	17420	7	1 hour
ETTh2	17420	7	1 hour
ETTm1	69680	7	15 minutes

To make the the experiment results compatible with previous studies on long sequence forecasting, we use the mean square error (MSE), i.e., Eq. (9) and mean absolute error (MAE), i.e., Eq. (10) as the measuring metrics.

$$MSE = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{\eta D}\sum_{t=b}^{\eta+b-1}\sum_{d=1}^{D}(\hat{y}_{t,d} - y_{t,d})^2, \tag{9}$$

$$MAE = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{\eta D}\sum_{t=b}^{\eta+b-1}\sum_{d=1}^{D}|\hat{y}_{t,d} - y_{t,d}|. \tag{10}$$

Both MSE and MAE reflect the average distance between the ground truth and the predicted sequence. η is the forecasting length and B is the number of different η long forecasting targets that the test set contains. For example, the total length of the test set is T and therefore the total number of forecasting targets is $B = T - \eta + 1$.

5.1.2 Baselines. For long sequence forecasting, the baselines are as follows:

- **LSTMa:** An LSTM network enhanced with the attention [3].
- **Reformer:** A transformer model with locality-sensitive hashing and reversible residual layers [19].
- **LogTrans:** A variant of the transformer model, enhanced with causal convolution to better learn local waveforms and logsparse attention to reduce memory cost [23].
- **LSTNet-Skip:** A model combining convolutional neural network (CNN) and long short term memory (LSTM) network for time-series forecasting. Skip indicates it is a variant with skip connections [20].
- **Informer:** A variant of the transformer model customized for long term forcasting[40].
- **Informer-:** A variant of Informer removing probsparse attention and self-attention distilling mechanisms.
- **HI:** Historical inertia (HI) is a simple, invariant yet effective baseline for time-series forecasting [8]

5.1.3 Implementation Details. For the model, the detailed network structure is shown in Table 3. Specifically, in the temporal module, the input sub-block is a one-layer MLP. The cascade sub-block and the forecasting sub-block both have two layers.

On training parameters, for the ETT datasets, the training, validation and test sets are in the span of 12, 4 and 4 months. For the Electricity dataset, the splitting ratios of the training, validation and test sets are 0.7, 0.1 and 0.2. The above two settings are identical to that of [40]. For normalization, the weight α is set to 0.1. We use the Adam optimizer with MSE loss and learning rate is set to 0.0002. The input historical sequence length is set to be larger than the forecasting horizon and the batch size is 32. For each dataset, we run 30 epochs and save the model with the highest validation MSE as the best model.

Table 3: MARINA Network Hyperparameters for Long Sequence Forecasting

Module	Sub-Module	Hyperparameter	Value
Temporal Module	MLP Block	Number of Blocks	2
	Input Sub-block	Neuron Number	200
		Activation	ReLU
	Cascade Sub-Block	Neuron Number	(200, 200)
		Activation	ReLU
	Forecasting Sub-Block	Neuron Number	(200,η)
		Activation	ReLU
Spatial Module	Self-Attention	Head Number h	8
		d_k	$\lfloor \eta/h \rfloor$
		d_v	$\lfloor \eta/h \rfloor$
		Dropout	0.05
	Feed Forward Network	Neuron Number	(200,η)
		Activation	ReLU
		Dropout	0.05
Output Module	MLP	Neuron Number	η

5.1.4 Main Results. The full comparison between MARINA and other baselines are shown in Table 4. The best results are in bold font and the second best are underlined. It is clear that the MARINA model is able to reach the best forecasting performance in all 40 metrics and outperforms the rest of the baselines with a great margin. Similar to [8], we also observe that even the simple baseline HI is able to outperform other deep learning based algorithms in most metrics. This surprising result corroborates our doubts on directly using NLP models in time-series forecasting out of their analogy.

It is fathomable that with larger forecasting horizon, all forecasting algorithms endure a performance decline. However, with larger horizon, the lead margin between MARINA and other baselines also become more significant which further demonstrates the robustness of MARINA in long term forecasting.

5.2 Anomaly Detection

In the anomaly detection tasks as studied in [6, 31], the objective is that given a sequence $\mathcal{Z}_{\omega+1}$ consisting of $\mathcal{X}_\omega(t) = \{X(t-\omega+1), ..., X(t)\}$ and the detection target point $Y(t+1)$, determine whether $Y(t+1)$ is an anomaly point. In this paper, we address this problem from the forecasting point of view and deem the unpredictable points as anomaly points.

5.2.1 Datasets and Metrics. To test the anomaly detection performance, four widely used datasets are selected, i.e., SMD[4], SMAP, MSL[5] and SWaT[6]. The detailed statistics of the used datasets are listed in Table 5.

On the evaluation metrics, we adopt the common metrics, i.e., Precision, Recall and F1-score

$$Precision = \frac{TP}{TP + FP}, \tag{11}$$

$$Recall = \frac{TP}{TP + FN}, \tag{12}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}, \tag{13}$$

to evaluate the anomaly detection performance. TP denotes the number of detected true anomalies, FP is the number of false alarms

[4]https://github.com/NetManAIOps/OmniAnomaly
[5]https://s3-us-west-2.amazonaws.com/telemanom/data.zip
[6]http://itrust.sutd.edu.sg/research/dataset

Table 4: Experimental results on long sequence forecasting. The best results are highlighted in bold and the second best results are underlined.

Dataset		ETTh1					ETTh2					ETTm1					Electricity				
		Horizon					Horizon					Horizon					Horizon				
Method	Metric	24	48	168	336	720	24	48	168	336	720	24	48	96	288	672	48	168	336	720	960
LSTMa	MSE	0.650	0.702	1.212	1.424	1.960	1.143	1.671	4.117	3.434	3.963	0.621	1.392	1.339	1.740	2.736	0.486	0.574	0.886	1.676	1.591
	MAE	0.624	0.675	0.867	0.994	1.322	0.813	0.221	1.674	1.549	1.788	0.629	0.939	0.913	1.124	1.555	0.572	0.602	0.795	1.095	1.128
Reformer	MSE	0.991	1.313	1.824	2.117	2.415	1.531	1.871	4.660	4.028	5.381	0.724	1.098	1.433	1.820	2.187	1.404	1.515	1.601	2.009	2.141
	MAE	0.754	0.906	1.138	1.280	1.520	1.613	1.735	1.846	1.688	2.015	0.607	0.777	0.945	1.094	1.232	0.999	1.069	1.104	1.170	1.387
LogTrans	MSE	0.686	0.766	1.002	1.362	1.397	0.828	1.806	4.070	3.875	3.913	0.419	0.507	0.768	1.462	1.669	0.355	0.368	0.373	0.409	0.477
	MAE	0.604	0.757	0.846	0.952	1.291	0.750	1.034	1.681	1.763	1.552	0.412	0.583	0.792	1.320	1.461	0.418	0.432	0.439	0.454	0.589
LSTNet-Skip	MSE	1.293	1.456	1.997	2.655	2.143	2.742	3.567	3.242	2.544	4.625	1.968	1.999	2.762	1.257	1.917	0.369	0.394	0.419	0.556	0.605
	MAE	0.901	0.960	1.214	1.369	1.380	1.457	1.687	2.513	2.591	3.709	1.170	1.215	1.542	2.076	2.941	0.445	0.476	0.477	0.565	0.599
Informer-	MSE	0.620	0.692	0.947	1.094	1.241	0.753	1.461	3.485	2.626	3.548	0.306	0.465	0.681	1.162	1.231	0.334	0.353	0.381	0.391	0.492
	MAE	0.577	0.671	0.797	0.813	0.917	0.727	1.077	1.612	1.285	1.495	0.371	0.470	0.612	0.879	1.103	0.399	0.420	0.439	0.438	0.550
Informer	MSE	0.577	0.685	0.931	1.128	1.215	0.720	1.457	3.489	2.723	3.467	0.323	0.494	0.678	1.056	1.192	0.344	0.368	0.381	0.406	0.460
	MAE	0.549	0.625	0.752	0.873	0.896	0.665	1.001	1.515	1.340	1.473	0.369	0.503	0.614	0.786	0.926	0.393	0.424	0.431	0.443	0.548
HI	MSE	0.426	0.498	0.653	0.690	0.714	0.266	0.379	0.572	0.567	0.635	1.395	1.668	0.423	0.526	0.655	0.328	0.212	0.247	0.469	0.518
	MAE	0.390	0.423	0.509	0.527	0.563	0.304	0.374	0.481	0.500	0.530	0.720	0.821	0.387	0.444	0.508	0.329	0.279	0.312	0.439	0.471
MARINA	MSE	**0.300**	**0.339**	**0.432**	**0.453**	**0.541**	**0.174**	**0.236**	**0.345**	**0.385**	**0.422**	**0.216**	**0.295**	**0.297**	**0.367**	**0.442**	**0.144**	**0.150**	**0.164**	**0.189**	**0.190**
	MAE	**0.353**	**0.376**	**0.432**	**0.450**	**0.514**	**0.265**	**0.305**	**0.383**	**0.414**	**0.450**	**0.292**	**0.342**	**0.346**	**0.388**	**0.431**	**0.240**	**0.245**	**0.263**	**0.291**	**0.294**

Table 5: Statistics of Datasets for anomaly detection

Dataset	Subset Number	Dimension Number	Training Samples	Test Samples	Anomaly Ratio(%)
SMD	28	38	708405	708420	4.16
SMAP	55	25	135183	427617	13.13
MSL	27	55	58317	73729	10.72
SWaT	1	51	475200	449919	12.14

Table 6: MARINA Network Hyperparameters for Anomaly Detection

Module	Sub-Module	Hyperparameter	Value
Temporal Module	MLP Block	Number of Blocks	2
	Input Sub-block	Neuron Number	(50,50,50,50)
		Activation	ReLU
	Cascade Sub-Block	Neuron Number	(50,50)
		Activation	ReLU
	Forecasting Sub-Block	Neuron Number	(50,24)
		Activation	ReLU
Spatial Module	Self-Attention	Head Number h	8
		d_k	3
		d_v	3
		Dropout	0.05
	Feed Forward Network	Neuron Number	(50,24)
		Activation	ReLU
		Dropout	0.05
Output Module	MLP	Neuron Number	1

and FN denotes the number of anomalies that fails to be detected. As the detection result is affected by both the neural network and the choice of threshold as shown in Eq. (1), to better evaluate the network, we enumerate all possible thresholds to search for the best F1 score. This is a common practice as taken in [6, 31].

Also, in performance evaluation, we adopt the point adjustment approach proposed in [31, 37]. In real scenarios, anomalies are often in the form of time windows instead of isolated points and with the point-adjustment approach, we deem that as long as one or more points in the anomaly window are detected, the points in the whole window are successfully detected as anomalies.

5.2.2 Baselines. In the task of anomaly detection, the benchmark baselines are:

- **VAE:** Variational auto-encoder (VAE), a variant of the auto-encoder structure by fusing the auto-encoder with the Bayesian learning structure [17].
- **EncDec-AD:** A reconstruction based anomaly detection model using LSTM auto-encoder [26].
- **GANomaly:** GANomaly is also based on the auto-encoder structure. It incorporates the generative adversarial network (GAN) to enhance accurate reconstruction [2].
- **LSTM-NDT:** A prediction based anomaly detection algorithm using the LSTM network [15].
- **DAGMM:** A reconstruction based anomaly detection model derived from VAE using GMM prior [41].
- **LSTM-VAE:** A reconstruction based detector combining VAE and LSTM networks [10].
- **BeatGAN:** Like GANomaly, BeatGAN is also uses the GAN structure to enhance the reconstruction ability of auto-encoders [39].
- **OmniAnomaly:** A variant of the VAE model for reconstruction based anomaly detection. It uses the planar normalizing flows technique to create more randomized prior [31].
- **DAEMON:** A reconstruction based anomaly detector with two GAN discriminators to regulate the hidden space and the output space [6].

5.2.3 Implementation Details. As shown in Table 6, the input sub-blocks in the temporal module has four MLP layers and both the forecasting and cascade sub-blocks has two MLP layers. The intermediate length for spatial correlation learning is set to 24.

On training parameters, we use the Adam optimizer with MSE loss function and learning rate is set to 0.005. The input historical sequence length is 100 and the batch size is 128. For each dataset, we save the model with the highest validation MSE as the best model. The normalization weight is $\alpha = 0.1$.

5.2.4 Main Results. The full comparison is shown in Table 7 with the best result in bold font and the second best underlined. It is clearly observed that in F1 score and Precision, MARINA outperforms all the baselines with obvious margin on all four datasets. Only on the MSL dataset, the Recall of MARINA is slightly behind that of DAEMON with a tiny margin of 0.007.

Also, as defined in Eq. (11) and Eq. (12), the Precision is related to false alarms while the Recall metric is related to missed detections. On most baselines there is a clear bias to either false alarm or missed detection. For example, in the SMD dataset, DAGMM shows a good Recall rate of 0.917, however, its false alarm rate is exceedingly high with the Precision being only 0.789. Even for the second best performing detector DAEMON, the largest Precision-Recall gap is as high as 9%. Such a bias is not appreciated as, for example, in AIOps scenarios, too many false alarms will overwhelm the operators while too many missed detections will cover the underlying problems. In MARINA, the largest Precision-Recall gap is only 4% which is significantly smaller than that of all the baselines.

5.3 Ablation Study

5.3.1 Normalization Method. In this study, we investigate the benefit brought by dynamic normalization. The testing scenario is long sequence forecast on ETTh1 and ETTh2 datasets. The results are listed in Table. 8. On the two preprocessing methods, i.e., static normalization and dynamic normalization, it is observed that dynamic normalization outperforms static normalization on all metrics. Within ETTh1 and ETTh2 datasets, the value range mismatch problem is obvious, therefore, the lead margin of dynamic normalization is prominent and the margin grows larger with higher horizon. The results demonstrates that, first, our proposed dynamic normalization method is effective against the concept drift problem; second, the negative effect of the concept drift problem on static normalization grows more prominent as the forecasting horizon grows larger.

5.3.2 Correlation Learning Modules. In this study, we investigate the effects of different temporal module structures and spatial module structures. The testing scenario is long sequence forecast with horizon 48 and 168 on the Electricity dataset.

For temporal modules, the baselines are the widely adopted NLP temporal modules: LSTM, GRU, self-attention in temporal dimension and TCN. As all these baseline modules do not alter the temporal length of the input, to ensure the output length of the the temporal modules equals to the forecasting length η, we take the last η steps of the temporal module's output as the input to the spatial module.

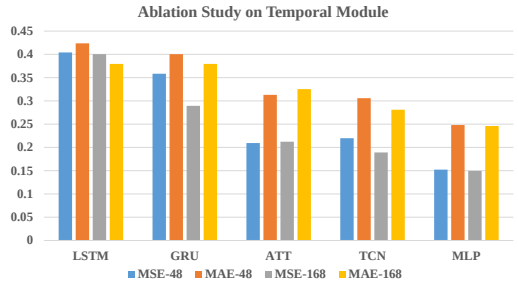

Figure 5: Ablation study: effect of different temporal modules

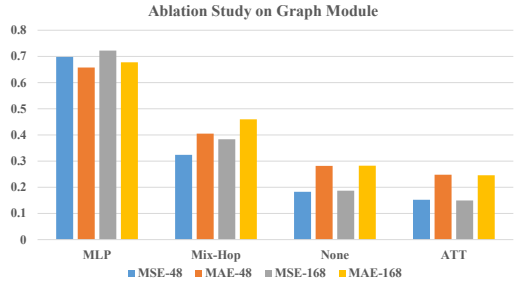

Figure 6: Ablation study: effect of different graph modules

As illustrated in Fig. 5, the LSTM module has the worst forecasting performance. As another variant in the RNN family, GRU performs slightly better than LSTM. The attention module and the TCN module are generally on par with each other and they outperform their recursive counterparts with obvious margin. One interesting observation is that using the self-attention module for temporal correlation learning results in better performance than the NLP seq2seq structure employed in Informer as illustrated in Table 4. Overall, the MLP structure with residual connections achieves the best performance in the long sequence forecasting tasks.

With respect to the spatial modules, the baselines are MLP, mix-hop, None and, as used in MARINA, self-attention. The MLP graph module means that MLPs are directly used for message passing among different dimensions. The mix-hop module is a sparse graph convolution network proposed in [36] with learnable parameters. None means that the spatial module of MARINA is removed.

As illustrated in Fig. 6, it is surprising to observe that the MLP module and the mix-hop module actually perform worse than the no spatial module case. The mix-hop uses learnable node embedding functions to explicitly learn the underlying graph of the time-series. However, whether such a learning scheme is viable is in doubt as reflected by the experimental results. With the attention spatial module, performance improvement can be observed compared with no graph learning. Therefore, the effectiveness of attention based graph learning is proven.

5.4 Efficiency Study

In efficiency study, we also study the long sequence forecasting tasks. We use the ETTh1 dataset with the input length set as 96, 192 and the forecasting length set as 48, 168. Six models are tested,

Table 7: Experimental results on anomaly detection. The best results are highlighted in bold and the second best results are underlined.

Dataset	SMD			SMAP			MSL			SWaT		
Metrics	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall
VAE	0.916	0.945	0.89	0.868	0.804	0.943	0.827	0.721	0.971	0.82	0.953	0.719
EncDec-AD	0.809	0.862	0.762	0.752	0.779	0.728	0.922	0.941	0.903	0.837	0.961	0.742
GANomaly	0.5	0.38	0.724	0.805	0.704	0.939	0.733	0.6	0.941	0.655	0.548	0.813
LSTM-NDT	0.649	0.557	0.778	0.911	0.845	0.989	0.764	0.63	0.969	0.723	0.677	0.774
DAGMM	0.848	0.789	0.917	0.781	0.699	0.884	0.709	0.723	0.696	0.806	0.895	0.734
LSTM-VAE	0.903	0.951	0.859	0.77	0.726	0.821	0.797	0.835	0.763	0.855	0.946	0.78
BeatGAN	0.921	0.901	0.942	0.87	0.771	**0.999**	0.901	0.844	0.967	0.818	0.949	0.719
OmniAnomaly	0.856	0.764	0.974	0.784	0.647	0.995	0.86	0.772	0.971	0.82	0.963	0.745
DAEMON	0.963	0.963	0.962	0.91	0.929	0.892	0.953	0.91	**1.0**	0.947	0.966	0.929
MARINA	**0.982**	**0.987**	**0.977**	**0.981**	**0.964**	**0.999**	**0.973**	**0.954**	0.993	**0.958**	**0.978**	**0.938**

Table 8: Ablation study on normalization methods

Dataset		ETTh1					ETTh2				
static	MSE	0.305	0.345	0.441	0.474	0.561	0.182	0.243	0.371	0.425	0.715
	MAE	0.356	0.381	0.445	0.473	0.546	0.275	0.321	0.408	0.450	0.610
dynamic	MSE	**0.300**	**0.339**	**0.432**	**0.453**	**0.541**	**0.174**	**0.236**	**0.345**	**0.385**	**0.422**
	MAE	**0.353**	**0.376**	**0.432**	**0.450**	**0.514**	**0.265**	**0.305**	**0.383**	**0.414**	**0.450**

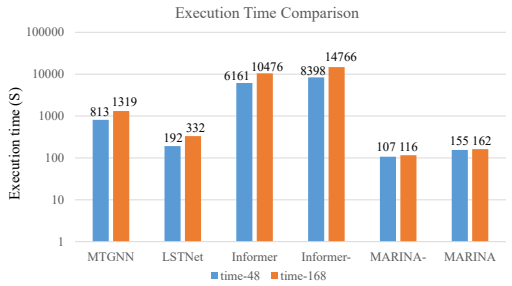

Figure 7: 30 epoch training plus test execution time

MTGNN, LSTNet-Skip, Informer, Informer-, MARINA- (MARINA without spatial module) and MARINA. We first run 30 training epochs, then conduct forecasting with the whole test set and at last record the whole execution time. The results are shown in Fig. 7.

It is observed that, MARINA and MARINA- is able to complete the tasks with minimum time consumption. Without the attention module, the execution time of MARINA- is within two minutes. When the number of dimensions of the input data is small and different dimensions do not show similar patterns, the attention module can be removed to save time. Even with the attention module, MARINA is still able to complete training and testing within three minutes. The execution time of all baseline models is significantly longer than that of MARINA. Among them, MTGNN's execution time, with its complex multi-layer graph learning module, is 8.1 times longer in the 168 step forecasting task compared with MARINA. Informer views each time point as a word vector. When the input time length is great, in the attention process, the matrix multiplication becomes extremely time consuming and the execution time of Informer and Informer- is unacceptably high.

One interesting observation is that, while other models show significant (at least 60%) execution time growth when the input length and the forecasting length becomes longer, MARINA- (time growth: 9 seconds, 8% higher) and MARINA (time growth: 7 seconds, 4.5% higher) shows minimum time growth. This property of MARINA makes it suitable for long sequence forecasting.

6 CONCLUSION

In this paper, we propose MARINA for multivariate time-series analysis. MARINA is equipped with an MLP temporal correlation module, a self-attention spatial correlation module and an output reshaping module to respectively learn the correlation between history and future points, the correlation among different dimensions as well as to reshape the output sequence to its desired length. Through extensive experiments, MARINA is proven to reach the SOTA performance in long sequence forecasting and anomaly detection tasks. MARINA's versatility and efficiency makes it suitable for industrial level applications.

7 ACKNOWLEDGMENTS

This work is partially supported by NSFC (No. 61972069, 61836007 and 61832017), and Shenzhen Municipal Science and Technology R&D Funding Basic Research Program (JCYJ20210324133607021).

REFERENCES

[1] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. 2017. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* 262 (Nov. 2017), 134–147.
[2] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. 2018. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian Conference on Computer Vision*. Springer, 622–637.
[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]* (May 2016). http://arxiv.org/abs/1409.0473 arXiv: 1409.0473.
[4] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv:1803.01271* (2018).
[5] C. Chatfield. 1978. The Holt-Winters Forecasting Procedure. *Journal of the Royal Statistical Society* 27, 3 (1978), 264–279.
[6] Xuanhao Chen, Liwei Deng, Feiteng Huang, Chengwei Zhang, Zongquan Zhang, Yan Zhao, and Kai Zheng. 2021. DAEMON: Unsupervised Anomaly Detection and Interpretation for Multivariate Time Series. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE.
[7] Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. 1990. STL: A seasonal-trend decomposition. *Journal of Official Statistics* 6, 1 (1990), 3–73.

[8] Yue Cui, Jiandong Xie, and Kai Zheng. 2021. Historical Inertia: An Ignored but Powerful Baseline for Long Sequence Time-series Forecasting. *arXiv:2103.16349 [cs]* (March 2021). http://arxiv.org/abs/2103.16349 arXiv: 2103.16349.

[9] E. S. Gardner. 1985. Exponential smoothing: The state of the art. *Journal of Forecasting* 4, 1 (1985), 1–28.

[10] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2018. A Multimodal Anomaly Detector for Robot-Assisted Feeding Using An Lstm-Based Variational Autoencoder. *IEEE Robotics and Automation Letters* 3, 3 (July 2018), 1544–1551.

[11] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI 2019*, Vol. 33. 922–929.

[12] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (July 2019), 922–929.

[13] E. J. Hannan and L. Kavalieris. 2010. REGRESSION, AUTOREGRESSION MODELS. *Journal of Time* 7, 1 (2010), 27–49.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Las Vegas, NV, USA, 770–778.

[15] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, London United Kingdom, 387–395.

[16] K. Kalpakis, D. Gada, and V. Puttagunta. 2002. Distance measures for effective clustering of ARIMA time-series. In *Proceedings 2001 IEEE International Conference on Data Mining*.

[17] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]* (May 2014). http://arxiv.org/abs/1312.6114 arXiv: 1312.6114.

[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]* (Feb. 2017). http://arxiv.org/abs/1609.02907 arXiv: 1609.02907.

[19] Nikita Kitaev, ?ukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. *arXiv:2001.04451 [cs, stat]* (Feb. 2020). http://arxiv.org/abs/2001.04451 arXiv: 2001.04451.

[20] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, Ann Arbor MI USA, 95–104.

[21] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. 2015. Generic and Scalable Framework for Automated Time-series Anomaly Detection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Sydney NSW Australia, 1939–1947.

[22] Dan Li, Dacheng Chen, Lei Shi, Baihong Jin, Jonathan Goh, and See-Kiong Ng. 2019. MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks. *arXiv:1901.04997 [cs, stat]* (Jan. 2019).

[23] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2020. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. *arXiv:1907.00235 [cs, stat]* (Jan. 2020). http://arxiv.org/abs/1907.00235

[24] Dapeng Liu, Youjian Zhao, Haowen Xu, Yongqian Sun, Dan Pei, Jiao Luo, Xiaowei Jing, and Mei Feng. 2015. Opprentice: Towards Practical and Automatic Anomaly Detection Through Machine Learning. In *Proceedings of the 2015 Internet Measurement Conference*. ACM, Tokyo Japan, 211–224.

[25] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2020. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting* 36, 1 (Jan. 2020), 54–74.

[26] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. *arXiv:1607.00148 [cs, stat]* (July 2016). http://arxiv.org/abs/1607.00148 arXiv: 1607.00148.

[27] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv:1905.10437 [cs, stat]* (Feb. 2020). http://arxiv.org/abs/1905.10437 arXiv: 1905.10437.

[28] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20, 1 (Jan. 2009), 61–80.

[29] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. 2020. Spatial-Temporal Synchronous Graph Convolutional Networks: A New Framework for Spatial-Temporal Network Data Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (April 2020), 914–921.

[30] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. 2016. Unsupervised Learning of Video Representations using LSTMs. *arXiv:1502.04681 [cs]* (Jan. 2016). http://arxiv.org/abs/1502.04681 arXiv: 1502.04681.

[31] ya su, youjian zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, Anchorage Alaska USA, 2828–2837.

[32] Sean Taylor and Benjamin Letham. 2018. Forecasting at scale. *The American Statistician* 72, 1 (2018), 37–74.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]* (Dec. 2017). http://arxiv.org/abs/1706.03762 arXiv: 1706.03762.

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *arXiv:1710.10903 [cs, stat]* (Feb. 2018). http://arxiv.org/abs/1710.10903 arXiv: 1710.10903.

[35] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2022. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *arXiv:2106.13008 [cs.LG]* (Jan. 2022). https://arxiv.org/abs/2106.13008 arXiv:2106.13008.

[36] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. 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*. ACM, Virtual Event CA USA, 753–763.

[37] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. 2018. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. In *Proceedings of the 2018 WWW*. 187–196.

[38] X. Zhang, C. Huang, Y. Xu, and L. Xia. 2020. Spatial-Temporal Convolutional Graph Attention Networks for Citywide Traffic Flow Forecasting. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management*.

[39] Bin Zhou, Shenghua Liu, Bryan Hooi, Xueqi Cheng, and Jing Ye. 2019. BeatGAN: Anomalous Rhythm Detection using Adversarially Generated Time Series. In *Proceedings of IJCAI*. Macao, China, 4433–4439.

[40] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*. AAAI Press, online.

[41] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. (2018), 1–19.