Leveraging Latent Causal Relationships Among Web Services for Traffic Prediction

Chang Tian[a], Mingzhe Xing[b,1,*], Zenglin Shi[c], Matthew Blaschko[a], Yinliang Yue[b] and Marie-Francine Moens[a]

[a]KU Leuven
[b]Beijing Zhongguancun Laboratory
[c]Hefei University of Technology
chang.tian@kuleuven.be,xingmz@zgclab.edu.cn

Abstract. Predicting web service traffic is crucial for system operation tasks including dynamic resource scaling, anomaly detection, and fraud detection. Web service traffic is characterized by frequent and drastic fluctuations over time and are influenced by heterogeneous user behaviors, making accurate prediction a challenging task. Previous research has extensively explored statistical approaches, and neural networks to mine features from preceding service traffic time series for prediction. However, these methods have largely overlooked the latent causal relationships between services. Drawing inspiration from causality in ecological systems, we empirically recognize the causal relationships between web services. To leverage these relationships for improved traffic prediction, we propose an effective neural network module, CCMPlus, designed to extract causal relationship features across services. This module can be seamlessly integrated with existing time series models to consistently enhance the performance of traffic predictions. We theoretically justify that the causal correlation matrix generated by the CCMPlus module captures causal relationships among services. Empirical results on real-world datasets from Microsoft Azure, Alibaba Group, and Ant Group confirm that our method surpasses state-of-the-art approaches in Mean Squared Error and Mean Absolute Error for predicting service traffic time series. These findings highlight the efficacy of feature representations from the CCMPlus module.

1 Introduction

User-oriented web services continue to grow exponentially, which has significantly accelerated the development of a diverse range of customized web applications [20, 34, 36, 35, 37]. These services attract a substantial user base and play a pivotal role in enabling various social and practical activities. For instance, YouTube has amassed 2.7 billion users[2], powered by its cutting-edge recommendation algorithms. Accurately predicting web service traffic carries significant social and practical value, with applications spanning dynamic resource scaling [25, 48], load balancing [26], anomaly detection [23], service-level agreement compliance [21], fraud detection [2], and so on. These capabilities not only enhance system performance but also improve the overall user experience [19].

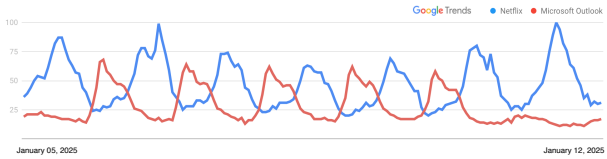


Figure 1: This figure illustrates the 7-day normalized search interest data for Netflix and Microsoft Outlook from Google Trends in the UK. The two services exhibit a causal relationship: an increase in Netflix usage corresponds to a decrease in the use of the email service.

Co-located, long-running web services often experience diverse workload patterns [48]. Web service traffic is characterized by frequent and significant fluctuations over time, driven by heterogeneous user behaviors. These factors collectively make predicting web service traffic a highly challenging task [31, 25]. Previous works formulates this task as a typical time series forecasting task. These approaches can broadly be categorized into statistical [17, 11, 18], machine learning [14, 7, 12], and deep learning [29, 9, 25, 48] methods. While statistical methods struggle to handle multi-dimensional and non-linear traffic data, machine learning methods address these limitations but fail to achieve the same level of accuracy as deep learning approaches [48, 3]. The recent advancements in Transformer [39] architecture have demonstrated superior performance in sequential prediction tasks. Consequently, Transformer-based methods [27] have emerged, achieving promising results in web service traffic prediction. Aside from these ad-hoc models, general state-of-the-art (SOTA) Transformer-based time-series prediction methods [47, 22, 6, 42] also have the potential to be applied to the web traffic task. Recently emerged LLM-based methods [33, 38, 16] leverage advanced reasoning capabilities by processing time series data through specially designed prompts, offering a promising avenue for temporal modeling.

Drawing inspiration from ecological causality, for example, grass abundance and rabbit populations influencing one another iteratively [15], we identify analogous patterns in web service traffic as shown in Figure 1. Empirical observations of Google Trends data reveal a causal relationship between leisure websites, such as Netflix, and work-related software, like Outlook. Specifically, increased web traffic to Netflix corresponds to decreased traffic to Outlook, and vice versa. These empirical observations suggest the existence of la-

* Mingzhe Xing and Yinliang Yue are corresponding authors.
[1] Co-first author.
[2] https://www.globalmediainsight.com/blog/youtube-users-statistics/

tent causal relationships underlying human-driven web behaviors. By uncovering and leveraging these causal relationships, we can achieve more accurate modeling of service traffic patterns. Building on this motivation and the causality theory of Convergent Cross Mapping (CCM) [32], which originates from ecology, we introduce the CCMPlus module. This module extracts features from web service traffic time series while including causal relationships among services, thereby enhancing the accuracy of web service traffic prediction. Furthermore, the CCMPlus module could integrate easily with existing time series forecasting models, enriching them with more informative, causally-aware features.

Concretely, we extend the CCM theory to effectively merge with neural networks, resulting in the development of the CCMPlus module. Specifically, the traditional CCM theory relies on a set of expert designed hyper-parameters, which might introduce severe bias to the causality identification. To alleviate this, we propose a multi-manifold embedding space which is constructed by learnable parameters and present the causality relationship in diverse spaces. After obtaining the intial embedding, we employ the CCM procedure but in the multi-manifold space to calculate the causal correlation matrix, which is updated with a momentum update mechanism. By integrating the causal correlation matrix to the initial embedding, it generates a resulting feature representation that incorporates informative causal information. This enhanced feature representation can then be concatenated with the feature representations of web service traffic time series extracted by existing time series models, thereby improving prediction accuracy. Our main contributions[3] in this work can be summarized as follows:

- **Method:** The CCMPlus module enhances existing time series forecasting models by generating feature representations that incorporate latent causal relationships across web services, addressing a critical limitation of many previous methods. Additionally, the CCMPlus module is designed for seamless integration with existing time series forecasting models, further contributing to improved prediction accuracy.
- **Theory:** We justify that the causal correlation matrix generated by the CCMPlus module effectively captures causal relationships across web services, enabling the incorporation of these relationships into web traffic prediction methods.
- **Experiments:** Experiments conducted on three real-world web service traffic datasets (Alibaba Group, Microsoft Azure, and Ant Group) demonstrate that our method achieves superior performance in terms of Mean Squared Error (MSE) and Mean Absolute Error (MAE) compared to previous state-of-the-art methods, thereby validating the effectiveness of the CCMPlus module.

2 Related Work

2.1 Web Service Traffic Prediction

Web service traffic prediction is a task critical to enabling service autoscaling, load balancing, and anomaly detection. As web service traffic is often represented as time series data, existing approaches primarily frame traffic prediction as a time series prediction problem [25, 48, 3].

Early research focused on statistical prediction methods [17, 11, 18] such as Moving Average, Auto-Regression, and Autoregressive Integrated Moving Average. These methods are valued for their simplicity and interpretability but are constrained by strict stationarity requirements. Moreover, they struggle to extend to multi-dimensional or non-linear data, limiting their applicability in dynamic environments. To overcome these limitations, methods like HOPBLR [14], LLR [7] and TWRES [12] employed machine learning techniques, including Logistic Regression and Support Vector Regression, respectively, for traffic prediction. However, these approaches were hindered by the limited expressive capacity of their models, resulting in suboptimal prediction accuracy. More recent advancements have shifted towards deep learning methods. CrystalLP [29] and GRUWP [9] utilize Long Short-Term Memory networks and Gated Recurrent Units, respectively, to predict service workloads. MagicScaler [25] proposes a novel multi-scale attentive Gaussian process-based predictor, capable of accurately forecasting future demands by capturing scale-sensitive temporal dependencies. The Performer [27] integrates the Transformer architecture into an encoder-decoder paradigm for service workload prediction. It leverages the self-attention mechanism to model temporal correlations and learn both global and local representations effectively. OptScaler [48] advances this direction with a proactive prediction module comprising a long-term periodic block and a short-term local block to capture multi-scale temporal dependencies. While existing traffic prediction methods demonstrate high accuracy through advanced time series forecasting techniques, they largely neglect the underlying causal relationships within the web services. Exploring these hidden causalities holds significant potential for further improving prediction performance.

2.2 Time Series Forecasting

Besides approaches specifically tailored for web service traffic prediction, general time series forecasting methods are also applied to predict web traffic [3, 48, 8].

Traditional statistical methods such as Prophet [30], and Holt-Winters [13] assume that time series variations adhere to predefined patterns. However, the inherently complex fluctuations of web service traffic often exceed the scope of these predefined patterns, thereby limiting the practical applicability of such statistical methods [43].

Recent advancements in neural network architectures have significantly enhanced temporal modeling capabilities. Neural network approaches for time series forecasting can be categorized into five paradigms [40, 33]: RNN-based, CNN-based, Transformer-based, MLP-based, and large language model (LLM)-based methods. Empirical methods often integrate components from the aforementioned categories, utilizing specific designs to effectively capture critical temporal features [40]. These specific designs incorporate series decomposition, multi-periodicity analysis, and multi-scale mixing architectures. For series decomposition, Autoformer [44] introduces a decomposition block based on moving averages, enabling the separation of complex temporal variations into seasonal and trend components. Building on this foundation, DLinear [45] utilizes series decomposition as a preprocessing step prior to performing linear regression. Crossformer [46] segments time series data into subseries-level patches and employs a Two-Stage Attention layer to effectively model cross-time and cross-variable dependencies within each patch. iTransformer [22] leverages the global representation of entire series and applies attention mechanisms to these series-wise representations, facilitating the capture of multivariate correlations. TimeXer [42] integrates external information into the Transformer architecture through a carefully designed embedding strategy, al-

[3] The code, data, and other resources will be available at https://github.com/changtianluckyforever/CCMPlus.

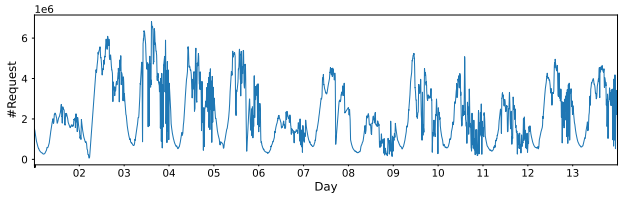


Figure 2: Web service traffic time series from Alibaba Cloud. Each point indicates the number of service requests at each time point.

lowing the inclusion of external information into patch-wise representations of endogenous series. In the context of multi-periodicity, NBEATS [24] employs multiple trigonometric basis functions to model time series, providing a robust framework for handling periodic patterns. Similarly, TimesNet [43] applies Fourier Transform to decompose time series into components of varying periodic lengths and utilizes a modular architecture to process these decomposed components effectively. With respect to multi-scale mixing architectures, Pathformer [6] adopts multi-scale patch representations and applies dual attention mechanisms across these patches to capture both global correlations and local details, thereby addressing temporal dependencies comprehensively. TimeMixer [40] captures temporal features by introducing a novel multi-scale mixing architecture, which comprises two key components: Past-Decomposable-Mixing, designed to leverage disentangled series for multi-scale representation learning, and Future-Multipredictor-Mixing, which ensembles complementary forecasting skills across multi-scale series to enhance prediction accuracy.

While general time series forecasting methods have been applied to web service traffic prediction [25, 48, 5], these methods often overlook causal relationships between services. In contrast, our CCMPlus module computes a causal correlation matrix used to generate temporal features incorporating causal relationships, significantly improving web service traffic prediction accuracy.

3 Background

In this section, we begin by giving a formal definition of web service traffic prediction task. Subsequently, we provide an overview of the causality theory, Convergent Cross Mapping (CCM), which serves as the theoretical foundation for our proposed CCMPlus module.

3.1 Web Service Traffic

Figure 2 presents an illustrative example of web service traffic time series from Alibaba Cloud. It exhibits significant fluctuations and frequent changes, primarily driven by human behavior and activity patterns. Accurately predicting web service traffic remains a well-recognized challenge due to the inherent complexity of traffic patterns, as highlighted by the research community [25, 48]. The web traffic prediction task can be formulated as follows:

$$y(t) = P(y(t-\alpha), y(t-2\alpha), \ldots, y(t-k\alpha)), \tag{1}$$

where $y(t)$ is the number of request at time t, α denotes the prediction granularity, and k is the historical sequence length.

3.2 Convergent Cross Mapping

Convergent Cross Mapping (CCM) theory published in Science [32] was originally proposed in the field of ecology and is designed to detect causal relationships between species. The original

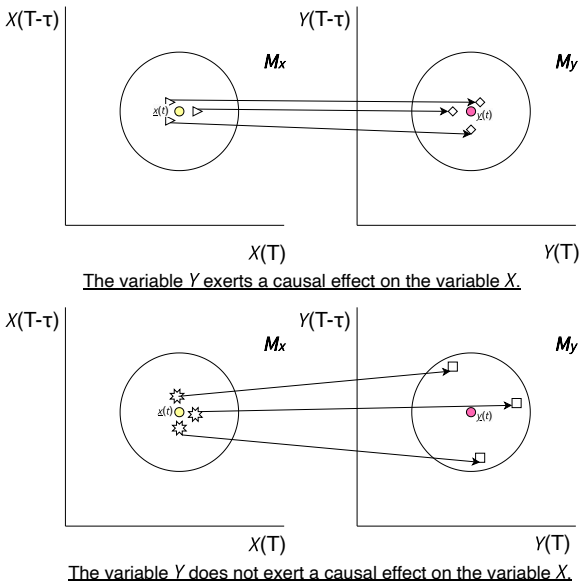


Figure 3: The point $\underline{y}(t)$ in the manifold M_y corresponds to the contemporaneous point in time $\underline{x}(t)$ in the manifold M_x.

work includes around 50 pages of supplementary material explaining the CCM theory. The exposition of CCM is often illustrated using the context of the Lorenz system. Its trajectory forms a manifold M in the state space, which consists of a collection of points that represent all possible states over time, with these points connected to create a structured geometric space.

The shadow manifolds, M_x or M_y, represents the projection of the original manifold M onto the system variables X or Y, respectively. Specifically, a lagged coordinate embedding utilizes E time-lagged values of $X(t)$ as coordinate axes to reconstruct the shadow manifold M_x. A point on M_x, denoted as $\underline{x}(t)$, is an E-dimensional vector expressed as:

$$\underline{x}(t) = [X(t), X(t-\tau), X(t-2\tau), \ldots, X(t-(E-1)\tau)],$$

where τ is a positive time lag, and E denotes the embedding dimension. Similarly, the same approach can be applied to points $\underline{y}(t)$ in the shadow manifold M_y.

Cross mapping refers to the process of identifying contemporaneous points in the manifold M_x of one variable X based on points in the manifold M_y of another variable Y. Specifically, given a point $\underline{y}(t)$ in the manifold M_y, the corresponding point in time from the manifold M_x is $\underline{x}(t)$. As illustrated in Figure 3, if Y exerts a causal effect on X, information from Y will be stored in X. Consequently, the neighbors of $\underline{x}(t)$ in M_x will correspond to points with the same time indices in M_y, and these corresponding points will also be neighbors of $\underline{y}(t)$. However, if Y has no causal effect on X, the information about Y in X will be incomplete [32]. As a result, the timely corresponding points in M_y will diverge and no longer be neighbors of $\underline{y}(t)$.

Convergence in CCM implies that if the variable Y has a causal effect on X, extending the observation period improves the ability to predict Y using points on the shadow manifold M_x. A longer observation period provides more trajectories to fill the gaps in the manifold, resulting in a more defined structure, which enhances the prediction of $\hat{Y}(t) \mid M_x$. Conversely, if two variables do not have a causal relationship, refining their manifolds will not lead to an improvement in predictive accuracy. More details of the CCM algo-

rithm can be found in Appendix Section A.

4 Methodology

Inspired by the concept of causal relationships in ecology, for example, the population dynamics of rabbits influence the abundance of grass, we analogically observe causal relationships in the web traffic patterns of different web services. As depicted in Figure 1, real-world data from Google Trends indicates that an increase in web traffic for Netflix corresponds to a decrease in traffic for the office software Outlook. To leverage the latent causality between services for traffic prediction, we extend the CCM theory by integrating it into a neural network module, referred to as CCMPlus (CCM+). It leverages causal relationships among web services to generate the CCMPlus representation (Section 4.1). This representation is used to enhance the representation produced by the Backbone Time Series Model (Section 4.2), boosting the accuracy of web service traffic time series prediction. Finally, we introduce the optimization process in Section 4.3.

4.1 CCMPlus Module

In this section, we aim to derive a feature representation that captures causal relationships across web services from raw time series data, thereby improving the precision of traffic prediction. This process follows the original CCM algorithm, and constructs the causal correlation matrix by estimating each time point of one web service time series using points from the shadow manifolds of another web service time series. After that, the casualty matrix is used to generate the CCMPlus representation.

① **Multi-Manifold Embedding** The first step is to derive the initial time series embedding. Given the raw time series $\hat{\mathbf{X}} \in \mathbb{R}^{N\times L}$ of length L for N web service traffic , we first extract date features, such as minutes, hours, and days, convert them into embeddings, and then combine these embeddings with the raw time series $\hat{\mathbf{X}}$ to form the C-dimensional time series embedding $\mathbf{X} \in \mathbb{R}^{N\times L\times C}$. Next, we follow the original CCM algorithm to transform the time series into shadow manifold space. Given that the time index of the time-evolving variable $X(t)$ ranges from 1 to L, the shadow manifold M_x is constructed by forming lagged coordinate vectors:

$$\underline{x}(t) = [X(t), X(t-\tau), X(t-2\tau), \ldots, X(t-(E-1)\tau)], \quad (2)$$

where $t \in [1+(E-1)\tau, L]$, τ represents the time lag, and E denotes the embedding dimension.

The traditional CCM algorithm relies on experts to set the value of τ, which might introduce human bias. However, since different web time series have different characteristics, determining τ based on expert knowledge is subjective, labor-intensive and challenging. To alleviate this issue, we extend it to a **multi-manifold space**, so as to learn the causality from **diverse shadow manifold spaces**. Specifically, we initialize two vectors of $\tau = [\tau_1, \ldots, \tau_i, \ldots, \tau_n]$ and the corresponding $\mathbf{E} = [E_1, \ldots, E_i, \ldots, E_n]$. For simplicity, we use the i-th shadow manifold, defined by the pair (τ_i, E_i), as an example to explain the CCMPlus procedures. The initialized time lag τ_i and the corresponding embedding dimension E_i are generated as follows:

$$E_i = \begin{cases} \left\lfloor \frac{\tau_w}{\tau_i} \right\rfloor - 1, & \text{if } \left\lfloor \frac{\tau_w}{\tau_i} \right\rfloor \bmod 2 = 0, \\ \left\lfloor \frac{\tau_w}{\tau_i} \right\rfloor, & \text{if } \left\lfloor \frac{\tau_w}{\tau_i} \right\rfloor \bmod 2 = 1, \end{cases}$$

where τ_w is set to 100 based on prior empirical observations for shadow manifolds as established in the literature [1]. Given the pair (τ_i, E_i), the corresponding calculation process of manifold embedding in Eq. 2 can be viewed as a convolution process, and can be effectively computed with the convolution neural network. Therefore, the convolution output can be derived as follows:

$$\mathbf{X}_{conv} = \text{Conv1D}\big(\mathbf{X};\ \text{kernel size} = E_i, \text{dilation} = \tau_i\big),$$

where $\mathbf{X}$ is reversed along the L dimension, the channels of input and output of the convolution network are C and D, respectively. The trajectory length of the i-th shadow manifold can be derived as $\bar{L} = L - \tau_i(E_i - 1)$.

After that, we can derive the shadow manifold embedding $\mathbf{X}_{ccm} \in \mathbb{R}^{N\times\bar{L}\times D}$ reshaped from $\mathbf{X}_{conv}$, where, for each time point in $\bar{L}$, the corresponding point in the shadow manifold is represented by coordinates in D dimensions. Specifically, $P = \mathbf{X}_{ccm}[m, :, :]$, with $P \in \mathbb{R}^{\bar{L}\times D}$, represents the points on the shadow manifold M_x, where the coordinates of the k-th point are given by $\underline{x}(k) = P[k, :]$. To ensure consistency with the trajectory length $\bar{L}$ of the shadow manifolds across different time series, we define the prediction target as $\mathbf{Y} = \hat{\mathbf{X}}[:, -\bar{L}:]$, where $\mathbf{Y} \in \mathbb{R}^{N\times\bar{L}}$. $\mathbf{Y}$ is reversed along the $\bar{L}$ dimension, the n-th target time series is $y(k) = \mathbf{Y}[n, :]$.

② **Estimate $y(k)$ within Multi-Manifold Space** We use the points in the shadow manifold M_x of one web service time series to predict the values $y(k)$ of another web service time series, denoted by $\hat{y}(k) \mid M_x$. This process can validate the existence of causality between the n-th and m-th time series of $\hat{\mathbf{X}}$.

We begin by locating the contemporaneous point $\underline{x}(k)$ in M_x, and find its $D+1$ nearest neighbors, denoting their time indices (from closest to farthest) by $t_{i_1}, ..., t_{i_{D+1}}$. Note that $D+1$ is the minimum number of points needed for a bounding simplex in an D-dimensional space. These neighbors are used to identify points in time series y to estimate $y(k)$ from a locally weighted mean of the $y(t_{i_s})$ values:

$$\hat{y}(k) \mid M_x = \sum_{s=1}^{D+1} w_{i_s} y(t_{i_s}),$$

where w_{i_s} represents the weight based on the distance between $\underline{x}(k)$ and its s-th nearest neighbor. The weights w_{i_s} are determined by:

$$w_{i_s} = \frac{u_{i_s}}{\sum_{j=1}^{D+1} u_{i_j}},$$
$$u_{i_s} = \exp\left\{ -\frac{d[\underline{x}(k), \underline{x}(t_{i_s})]}{d[\underline{x}(k), \underline{x}(t_{i_1})]} \right\},$$

where $d[\underline{x}(k), \underline{x}(t_{i_s})]$ denotes the **Euclidean distance** between the two points in the shadow manifold M_x.

③ **Momentum-Updated Correlation Coefficient Matrix** The predictions are then compared with the ground truth values of the target web service time series to compute the correlation coefficient for each pair. As a result, we can obtain the causal correlation matrix: $\mathbf{M} \in \mathbb{R}^{N\times N}$, where each element represents the causal relationship between a pair of web services.

To illustrate, we use a specific element of $\mathbf{M}$ indexed as (m, n) to demonstrate the process for quantifying causal relationships. $\mathbf{M}[m, n]$ quantifies the causal effect of the n-th time series of $\hat{\mathbf{X}}$ on the m-th time series of $\hat{\mathbf{X}}$ and can be calculated as follows:

$$\mathbf{M}[m, n] = \frac{\sum_{k=0}^{\bar{L}-1} \left(y(k) - \overline{y}(k)\right) \left(\hat{y}(k) - \overline{\hat{y}}(k)\right)}{\sqrt{\sum_{k=0}^{\bar{L}-1} \left(y(k) - \overline{y}(k)\right)^2 \sum_{k=0}^{\bar{L}-1} \left(\hat{y}(k) - \overline{\hat{y}}(k)\right)^2}}.$$

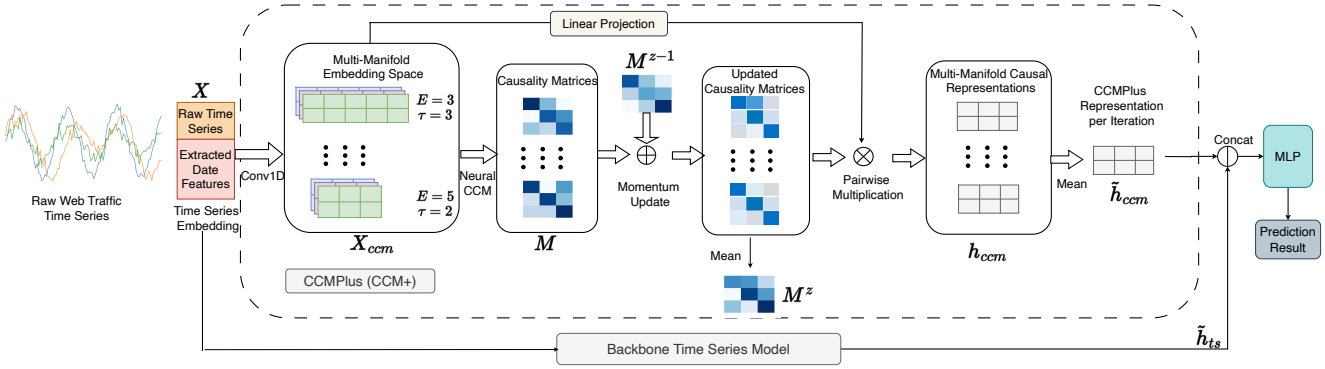


Figure 4: An overview of the proposed method, which consists of the CCMPlus module and the Backbone Time Series Model. The CCMPlus module identifies latent causal relationships among web services, generating a CCMPlus representation that enhances the predictive performance of the Backbone Time Series Model.

To ensure consistent and stable representations, the $\mathbf{M}$ matrix is updated using a momentum-based approach [10] with the causal correlation matrix $\widetilde{\mathbf{M}}^{z-1} \in \mathbb{R}^{N\times N}$ from the previous iteration z-1, followed by a softmax operation for normalization:

$$\mathbf{M}^z = (1-\lambda)\cdot \mathbf{M} + \lambda \cdot \widetilde{\mathbf{M}}^{z-1}, \tag{3}$$
$$\widehat{\mathbf{M}}^z(i) = \text{Softmax}(\mathbf{M}^z),$$

where z is the current iteration number, and λ is the momentum value in the range $(0, 1)$. $\widehat{\mathbf{M}}^z(i) \in \mathbb{R}^{N\times N}$ captures causal relationships among the N web services with the i-th shadow manifold of each web service time series.

④ **Causality Enhanced Time Series Representation** To incorporate causality among web services into the feature representation $\mathbf{h}_{ccm}(i)$ of time series, we use the causality score to weight the feature representation $\widehat{\mathbf{X}_{ccm}}$ of the shadow manifold embedding. Specifically, we first transpose and mapping $\mathbf{X}_{ccm}$ into $\widehat{\mathbf{X}_{ccm}} \in \mathbb{R}^{N\times D}$, and then computed the causality-weighted representation:

$$\mathbf{h}_{ccm}(i) = \widehat{\mathbf{M}}^z(i) \cdot \widehat{\mathbf{X}_{ccm}}, \tag{4}$$
$$\widehat{\mathbf{X}_{ccm}} = \text{Transpose}(\mathbf{X}_{ccm}) \cdot \mathbf{w} + b, \tag{5}$$

where $\mathbf{h}_{ccm}(i) \in \mathbb{R}^{N\times D}$ is the feature representation derived from the i-th shadow manifold. The CCMPlus representation $\widetilde{\mathbf{h}}_{\mathbf{ccm}}$ and causal correlation matrix $\widetilde{\mathbf{M}}^z$ are computed by averaging the representations and matrices from all shadow manifolds:

$$\widetilde{\mathbf{h}}_{\mathbf{ccm}} = \text{Mean}(\sum_{i=1}^{n} \mathbf{h}_{ccm}(i)), \tag{6}$$
$$\widetilde{\mathbf{M}}^z = \text{Mean}(\sum_{i=1}^{n} \widehat{\mathbf{M}}^z(i)),$$

where $\widetilde{\mathbf{M}}^z \in \mathbb{R}^{N\times N}$ and $\widetilde{\mathbf{h}}_{\mathbf{ccm}} \in \mathbb{R}^{N\times D}$ are the causal correlation matrix of the z-th iteration and the feature representation of the CCMPlus module respectively.

4.2 *Backbone Time Series Model*

The Backbone Time Series Model, denoted as BTSM, processes the input time series embedding $\mathbf{X} \in \mathbb{R}^{N\times L\times C}$ and outputs a feature representation $\widetilde{\mathbf{h}}_{\mathbf{ts}} \in \mathbb{R}^{N\times Q}$: $\widetilde{\mathbf{h}}_{\mathbf{ts}} = \text{BTSM}(\mathbf{X})$, which is concatenated with $\widetilde{\mathbf{h}}_{\mathbf{ccm}}$ in the training or testing mode, to jointly predict the web service traffic time series.

Based on the baseline evaluation performances across three real-world web service traffic datasets at multiple prediction granularities (Tables 1, 2 and Tables 5, 6 in Appendix) and insights from recent research [28, 41], TimesNet [43] and iTransformer [22] emerge as two strong-performing baselines among state-of-the-art time series models. TimesNet identifies and utilizes multi-periodicity, decomposing temporal variations into intraperiod and interperiod components. The core module, TimesBlock, adaptively discovers periodicities and extracts features using a parameter-efficient inception block. iTransformer repurposes the Transformer architecture for time series forecasting by applying attention and feed-forward networks on inverted dimensions. It embeds time points as variate tokens, allowing attention to capture multivariate correlations and the feed-forward network to learn nonlinear variate-specific representations.

To broadly verify the effectiveness of the CCMPlus (CCM+) module, we integrate it with these two Backbone Time Series Models, resulting in two variant models, *e.g.,* **CCM+iTransformer** and **CCM+TimesNet**.

4.3 *Optimization*

For convenient combination with the Backbone Time Series Model and to improve generalization, the CCMPlus representation $\widetilde{\mathbf{h}}_{\mathbf{ccm}}$ is concatenated with the time series model feature representation $\widetilde{\mathbf{h}}_{\mathbf{ts}}$. In general, the whole procedure can be formalized as: $\hat{x} = \text{MLP}\left(\widetilde{\mathbf{h}}_{\mathbf{ccm}} \mid \widetilde{\mathbf{h}}_{\mathbf{ts}}\right)$, where $\hat{x}$ is the predicted time series value, and MLP denotes the multilayer perception that projects the concatenated hidden feature to the final prediction. Following previous work [22, 43], we employ the mean square error as the loss function to optimize the model parameters. The overall training and inference algorithm can be found in Appendix Section B.

5 Experiments

5.1 *Datasets*

We conduct experiments on three publicly available real-world web service traffic datasets from Ant Group[4], Microsoft Azure[5], and Alibaba Group[6]. The Ant Group Traffic dataset includes 113 web services spanning a time range of 146 days. The Microsoft Azure Traffic dataset consists of 1,000 web services with a time range of 14 days. Similarly, the Alibaba Group Traffic dataset contains 1,000 web services, covering a total duration of 13 days.

[4] https://huggingface.co/datasets/kashif/App_Flow/
[5] https://github.com/Azure/AzurePublicDataset
[6] https://github.com/alibaba/clusterdata

Table 1: Prediction performances averaged over three runs with prediction granularity α set as 30 minutes. The best result is marked in bold. The t-test conducted on both metrics indicates that the improvement is statistical significant (p-value < 0.001).

30 Minutes	Alibaba Group Traffic		Microsoft Azure Traffic		Ant Group Traffic		Overall Mean	
Method	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MagicScaler [25]	3.4909	0.5362	42.0655	0.8441	1.5513	1.0743	15.7026	0.8182
OptScaler [48]	3.5708	0.6147	33.7485	0.9459	1.3017	0.9421	12.8737	0.8342
Llama3 [38]	7.0570	1.1545	43.8206	1.8958	3.4346	1.5931	18.1041	1.5478
TimeLLM [16]	3.4985	0.5339	17.2852	0.7088	1.5049	1.0560	7.4295	0.7662
TimeMixer [40]	3.1429	0.5455	15.1801	0.6759	1.4036	0.9950	6.5755	0.7388
iTransformer [22]	3.1247	0.5428	19.5574	0.7933	1.4116	1.0010	8.0312	0.7790
CCM+iTransformer (ours)	3.0773	**0.5098 ↓6.08%**	**14.3191 ↓26.8%**	**0.6482 ↓18.3%**	1.3162	0.9402	**6.2375 ↓22.33%**	**0.6994 ↓10.22%**
TimesNet [43]	3.1843	0.5406	16.6185	0.7177	1.4096	0.9989	7.0708	0.7524
CCM+TimesNet (ours)	**3.0206 ↓5.14%**	0.5200	14.9237	0.6791	**1.2897 ↓8.51%**	**0.9350 ↓6.40%**	6.4113	0.7114

Table 2: Prediction performances averaged over three runs with prediction granularity α set as 5 minutes. The best result is marked in bold. The t-test conducted on both metrics indicates that the improvement is statistical significant (p-value < 0.001).

5 Minutes	Alibaba Group Traffic		Microsoft Azure Traffic		Ant Group Traffic		Overall Mean	
Method	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MagicScaler [25]	3.2469	0.5176	9.3378	0.5473	1.5326	1.0640	4.7058	0.7096
OptScaler [48]	3.1070	0.4862	8.5807	0.5410	1.3214	0.9373	4.3364	0.6548
Llama3 [38]	7.3436	1.1474	11.5244	1.3587	3.3047	1.5606	7.3909	1.3556
TimeLLM [16]	3.2588	0.5071	6.5393	0.5063	1.4974	1.0459	3.7652	0.6864
TimeMixer [40]	2.7051	0.4818	3.0460	0.3890	1.3923	0.9819	2.3811	0.6176
iTransformer [22]	2.6458	0.4664	3.0367	0.3769	1.3939	0.9853	2.3588	0.6095
CCM+iTransformer (ours)	2.5657	0.4460	2.9780	0.3662	1.3025	0.9398	2.2821	0.5840
TimesNet [43]	2.1681	0.3925	2.8696	0.3527	1.3923	0.9822	2.1433	0.5758
CCM+TimesNet (ours)	**1.8103 ↓16.50%**	**0.3394 ↓13.53%**	**2.6347 ↓8.19%**	**0.3150 ↓10.69%**	**1.2871 ↓7.56%**	**0.9287 ↓5.45%**	**1.9107 ↓10.85%**	**0.5277 ↓8.35%**

5.2 Experiment Settings

5.2.1 Baselines

We evaluate our methods against the following baselines:

- **Large Language Models**: Llama3 [38] and TimeLLM [16].
- **Specialized Web Service Traffic Prediction Models**: MagicScaler [25] and OptScaler [48].
- **General Time Series Prediction Models**: TimesNet [43], TimeMixer [40], and iTransformer [22].

More details about baselines can be found in the related work (§2). These baselines represent the SOTA approaches in their respective categories. We apply the same experiment settings on these baselines and our methods to ensure fair comparison.

5.2.2 Evaluation Metrics

We evaluate the model performances using Mean Squared Error (MSE) and Mean Absolute Error (MAE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \quad \text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|,$$

where y_i and $\hat{y}_i$ denote the true and predicted values, respectively, and n is the number of samples in the test set.

5.2.3 Implementation Details

The Llama3 (8B) baseline is fine-tuned using LoRA (rank is set as 16). The τ_w is set as 100, and $\tau \in [1, 2, 3, 4]$. The momentum value λ is 0.5. The channels of input and output of the convolution network are C and D, C and D are set as 16 and 32, respectively. The batch size B is 8, and the Adam optimizer is configured with a learning rate of 0.000001. The input length L is 168. Training is performed on an H100 GPU. The model is trained for 15 epochs with an early stopping patience of 5 epochs based on validation performance. The prediction granularity parameter, α, corresponds to the prediction horizon, distinguishing between long-term and short-term forecasts. α has following values: 1, 5, 15 and 30.

5.3 Prediction Performance Analysis

Performance Comparisons. To evaluate the performance of CCM-Plus (CCM+), we compare two variants of our proposed framework, CCM+TimesNet and CCM+iTransformer, against the baselines. As shown in Table 1, we report the prediction results on three datasets, with the prediction granularity α (explained in Eq. 1) set to 30 minutes. Among all the baselines, Llama3 performs the worst. While large language models have demonstrated effectiveness in reasoning over sequential data, web traffic exhibits significantly higher volatility, posing substantial challenges for plain LLMs to accurately forecast future values. In contrast, TimeLLM inherently treats time series patches as tokens and employs task-specific prompt embeddings for forecasting. Although it captures patch correlations, it still underperforms on dynamic web traffic data compared to methods specifically tailored for time series analysis. Magicscaler and OptScaler, in particular, are designed for service workload prediction. While these models account for the volatility inherent in traffic time series, they fail to explicitly capture the underlying seasonal and trend components, which are crucial for predicting web service traffic driven by human behavior.

TimeMixer addresses this limitation by decomposing time series into seasonal and trend components across multiple periodicities, enabling it to learn decomposed temporal patterns ranging from fine-grained to macro-level perspectives. As a result, TimeMixer achieves superior prediction performance compared to both LLM-based methods and specialized workload prediction models. TimesNet leverages Fourier transforms to derive more adaptive periodicity terms, further reducing prediction errors. On the other hand, iTransformer treats independent time series as tokens, learning both temporal and cross-dimensional correlations by modeling token sequences, and achieves performance comparable to TimesNet. However, all the above methods focus solely on internal temporal patterns while neglecting external causal relationships across multiple time series. To address this limitation, we integrate CCMPlus with TimesNet and iTransformer—the two best-performing baseline models—to create two new variants. As observed, while TimesNet and iTransformer employ carefully designed architectures and achieve SOTA performance, integrating CCMPlus leads to further improvements. This highlights the importance of capturing the causality inherent in web service traffic to accurately predict traffic volume. Furthermore, CCMPlus

effectively models this causality, contributing to enhanced predictive performance.

Prediction Granularity Analysis. Prediction granularity is an important factor for traffic forecasting. To assess this, we vary the prediction granularity α in the time series in $\{1, 5, 15, 30\}$ minutes and compare the model performances in Table 6, 2, 5 and 1, respectively. (results for $\alpha = 1$ and $\alpha = 15$ are in Appendix §C.) Note that the more fine-grained time interval setting would generate more data samples. First, we observe that CCMPlus consistently improves the SOTA models (TimesNet and iTransformer), underscoring the importance of considering causality among service traffic patterns. Notably, CCMPlus demonstrates flexibility in integrating with various SOTA backbone models, highlighting its potential for performance enhancement. Second, the performance improvements diminish for the 1-minute prediction granularity setting. This is because service traffic is highly dynamic and volatile, and fine-grained time series data often lack a sufficient time duration to accumulate information necessary for capturing reliable causal relationships. Nevertheless, CCMPlus still achieves the best performance, further demonstrating its superiority in extracting hidden causal relationships and boosting prediction accuracy even under challenging conditions. We conduct a t-test [4] under both metrics shows that the improvement of our method over iTransformer and TimesNet is significant (p-value < 0.001).

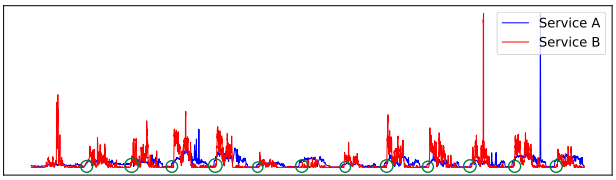
5.4 *Ablation Study*

The proposed model consists of the CCMPlus module (§4.1) and BTSM module (§4.2), where the CCMPlus encompasses of two key internal representations, i.e., the multi-manifold embedding and causality enhanced time-series representation (① and ④ introduced in §4.1). To assess the contribution of these modules and representations, we conduct a detailed ablation study using CCM+TimesNet on the most challenging dataset, Microsoft Azure Traffic. Table 4 presents the impact of model variants built on these modules and representations, yielding the following insights: (**1**) Excluding the Backbone Time Series Model (w/o BTSM) reduces performance. The deep learning-based backbone extracts essential seasonal and trend-related temporal features, which are crucial for accurate web service traffic forecasting. (**2**) By comparing the results when both Multi-Manifold Embedding (MME) and Causality Enhanced Representations (CER) are removed with those when only CER is removed, we can observe that MME provides diverse time-sensitive representations from multiple shadow manifold spaces formed by different time lags and embedding dimension values, which can lead to performance improvement for the advanced BTSM model; (**3**) Our complete model (i.e., BTSM+MME+CER) performs better than the variant without CER, which demonstrates that CER can generate feature representations that capture causal relationships among web services, enhancing traffic prediction accuracy.

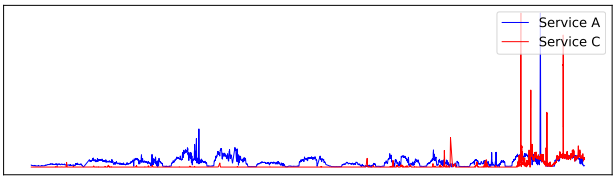
5.5 *Hyperparameter Tuning*

Table 3 presents the hyperparameter analysis, evaluating the impact of the following key parameters: (1) C defines the embedding dimension of the input time series $\hat{\mathbf{X}}$. As shown in Table 3, the model maintains stable performance across values, demonstrating robustness. Increasing this dimension enhances feature richness, but for efficiency, we set $C = 16$. (2) D defines the embedding dimension of shadow manifold points. A small value may miss features, while a large one can introduce noise. To balance representation quality and efficiency, we set $D = 32$. (3) The values in τ define the dilation parameter in Conv1D, controlling the number of shadow manifolds. Increasing this number enhances feature extraction but incurs higher computational costs. To balance accuracy and efficiency, we set $\tau = [1, 2, 3, 4]$.

(a) The causal relationship heatmap of web service A.



(b) Traffic time-series plots of web service A and service B.



(c) Traffic time-series plots of web service A and service C.

Figure 5: The heatmap of web service A's causal relationships and the plots of the service A, B and C with the highest and lowest causal relationship, respectively.

5.6 *Case Study*

Using the Alibaba Group Traffic dataset, we compute causal relationships between web services with the CCMPlus module. To visualize, we compare Service A with twenty randomly sampled services and plot the causal correlation heatmap (Figure 5a), where Service A exhibits a strong causal correlation with the third service, referred to as Service B, and the weakest causal correlation with the seventh service, referred to as Service C. To further analyze this relationship, we plot the traffic time series of Services A, B and C over the same period (Figure 5b and 5c). Figure 5b indicate the causality correlation between service A and B, where a increase of A prompt the lagged increase of B (as marked in green circle), which exhibits a high causality score. From Figure 5c, we can observe that A and C show no discernible correlation, and the traffic of service C is steady and independent with A, which is also aligned with the heatmap results.

6 Conclusion

We propose the CCMPlus module, which captures causal relationships between web services to improve web service traffic prediction. CCMPlus first constructs multi-manifold for each web service traffic time series. It then estimates target time series from the multi-manifold space of other web services. By comparing these estimations with the ground truth, a causal correlation matrix is computed and used to generate the CCMPlus representation, quantifying inter-service causal dependencies. This representation is con-

Table 3: Hyperparameter analysis of C, D, and τ on the Microsoft Azure Traffic dataset with 5-minute prediction granularity using CCM+TimesNet. The results are averaged over three experiments.

C	MSE	MAE	D	MSE	MAE	τ	MSE	MAE
12	2.6873	0.3178	28	2.8307	0.3149	[1, 2, 3]	2.7945	0.3157
14	2.8256	0.3385	30	2.6894	0.3223	[1, 2, 3, 4] (ours)	2.6347	0.3150
16 (ours)	2.6347	0.3150	32 (ours)	2.6347	0.3150	[1, 2, 3, 4, 5]	2.6531	0.3144
18	2.7078	0.3170	34	2.6357	0.3143			
20	2.6311	0.3117	36	2.7461	0.3273			

Table 4: Ablation study on the Microsoft Azure Traffic dataset with time granularity α set as 5 minutes.

Method	MSE	MAE
w/o BTSM	2.9981	0.3627
w/o MME&CER	2.8696	0.3527
w/o CER	2.7329	0.3320
CCM+TimesNet (ours)	2.6347	0.3150

catenated with the time series model's output, bridging the gap in causal inference and enhancing prediction performance. We evaluate CCMPlus-integrated models on real-world web service traffic datasets, which demonstrates that CCMPlus consistently improves web service traffic prediction.

References

[1] State space reconstruction parameters in the analysis of chaotic time series—the role of the time window length. *Physica D: Nonlinear Phenomena*, 95(1):13–28, 1996.

[2] K. Al-talak and O. Abbass. Detecting server-side request forgery (ssrf) attack by using deep learning techniques. *International Journal of Advanced Computer Science and Applications*, 12(12):12, 2021.

[3] S. Alharthi, A. Alshamsi, A. Alseiari, and A. Alwarafy. Auto-scaling techniques in cloud computing: Issues and research directions. *Sensors*, 24(17):5551, 2024.

[4] B. Bhattacharya and D. Habtzghi. Median of the p value under the alternative hypothesis. *The American Statistician*, 56(3):202–206, 2002.

[5] M. Catillo, U. Villano, and M. Rak. A survey on auto-scaling: how to exploit cloud elasticity. *International Journal of Grid and Utility Computing*, 14(1):37–50, 2023.

[6] P. Chen, Y. ZHANG, Y. Cheng, Y. Shu, Y. Wang, Q. Wen, B. Yang, and C. Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *The Twelfth International Conference on Learning Representations*.

[7] M. Daraghmeh, S. B. Melhem, A. Agarwal, N. Goel, and M. Zaman. Linear and logistic regression based monitoring for resource management in cloud networks. In *2018 IEEE 6th international conference on future internet of things and cloud (FiCloud)*, pages 259–266. IEEE, 2018.

[8] J. Deng, X. Chen, R. Jiang, D. Yin, Y. Yang, X. Song, and I. W. Tsang. Disentangling structured components: Towards adaptive, interpretable and scalable time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 36(8):3783–3800, 2024.

[9] Y. Guo and W. Yao. Applying gated recurrent units pproaches for workload prediction. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6. IEEE, 2018.

[10] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[11] Y. Hu, B. Deng, and F. Peng. Autoscaling prediction models for cloud resource provisioning. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pages 1364–1369. IEEE, 2016.

[12] Z. Hu, H. Kang, and M. Zheng. Stream data load prediction for resource scaling using online support vector regression. *Algorithms*, 12(2):37, 2019.

[13] R. Hyndman. *Forecasting: principles and practice*. OTexts, 2018.

[14] M. B. Issa, M. Daraghmeh, Y. Jararweh, M. Al-Ayyoub, M. Alsmirat, and E. Benkhelifa. Using logistic regression to improve virtual machines management in cloud computing systems. In *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 628–635. IEEE, 2017.

[15] L. Ji and J. Xiong. Superprocesses for the population of rabbits on grassland. *Proceedings of the Steklov Institute of Mathematics*, 316(1): 195–208, 2022.

[16] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations*.

[17] D. Kapgate. Weighted moving average forecast model based prediction service broker algorithm for cloud computing. *International Journal of Computer Science and Mobile Computing*, 3(2):71–79, 2014.

[18] A. S. Kumar and S. Mazumdar. Forecasting hpc workload using arma models and ssa. In *2016 International conference on information technology (ICIT)*, pages 294–297. IEEE, 2016.

[19] S. Kumar, S. Chattopadhyay, and C. Adak. Tpmcf: Temporal qos prediction using multi-source collaborative features. *IEEE Transactions on Network and Service Management*, 2024.

[20] D. Li, S. Wang, J. Zou, C. Tian, E. Nieuwburg, F. Sun, and E. Kanoulas. Paint4poem: A dataset for artistic visualization of classical chinese poems. *arXiv preprint arXiv:2109.11682*, 2021.

[21] H. Liao, T.-y. Liu, J. Guo, B. Huang, D. Yang, and J. Ding. Retrospecting available cpu resources: Smt-aware scheduling to prevent sla violations in data centers. *IEEE Transactions on Parallel & Distributed Systems*, (01):1–17, 2024.

[22] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*.

[23] K. Mitropoulou, P. Kokkinos, P. Soumplis, and E. Varvarigos. Anomaly detection in cloud computing using knowledge graph embedding and machine learning mechanisms. *Journal of Grid Computing*, 22(1):6, 2024.

[24] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[25] Z. Pan, Y. Wang, Y. Zhang, S. B. Yang, Y. Cheng, P. Chen, C. Guo, Q. Wen, X. Tian, Y. Dou, et al. Magicscaler: Uncertainty-aware, predictive autoscaling. *Proceedings of the VLDB Endowment*, 16(12):3808–3821, 2023.

[26] A. Pavlenko, J. Cahoon, Y. Zhu, B. Kroth, M. Nelson, A. Carter, D. Liao, T. Wright, J. Camacho-Rodríguez, and K. Saur. Vertically autoscaling monolithic applications with caasper: Scalable c ontainer-a s-a-s ervice p erformance e nhanced r esizing algorithm for the cloud. In *Companion of the 2024 International Conference on Management of Data*, pages 241–254, 2024.

[27] W. Qi, J. Yao, J. Li, and W. Wu. Performer: A resource demand forecasting method for data centers. In *International Conference on Green, Pervasive, and Cloud Computing*, pages 204–214. Springer, 2022.

[28] X. Qiu, J. Hu, L. Zhou, X. Wu, J. Du, B. Zhang, C. Guo, A. Zhou, C. S. Jensen, Z. Sheng, and B. Yang. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. *Proc. VLDB Endow.*, 17(9):2363–2377, 2024.

[29] L. Ruan, Y. Bai, S. Li, S. He, and L. Xiao. Workload time series prediction in storage systems: a deep learning based approach. *Cluster Computing*, pages 1–11, 2023.

[30] J. Sean and J. Taylor. Forecasting at scale. *Am. Stat*, 72(1):37–45, 2018.

[31] M. Straesser, S. Geissler, S. Lange, L. K. Schumann, T. Hossfeld, and S. Kounev. Trust your local scaler: A continuous, decentralized approach to autoscaling. *Performance Evaluation*, 167:102452, 2025.

[32] G. Sugihara, R. May, H. Ye, C.-h. Hsieh, E. Deyle, M. Fogarty, and S. Munch. Detecting causality in complex ecosystems. *science*, 338 (6106):496–500, 2012.

[33] M. Tan, M. A. Merrill, V. Gupta, T. Althoff, and T. Hartvigsen. Are language models actually useful for time series forecasting? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[34] C. Tian, W. Yin, and M. F. Moens. Anti-overestimation dialogue policy learning for task-completion dialogue system. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 565–577, 2022.
[35] C. Tian, M. Blaschko, W. Yin, M. Xing, Y. Yue, and M. F. Moens. A generic method for fine-grained category discovery in natural language texts. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3548–3566, 2024.
[36] C. Tian, W. Yin, D. Li, and M.-F. Moens. Fighting against the repetitive training and sample dependency problem in few-shot named entity recognition. *Ieee Access*, 12:37600–37614, 2024.
[37] C. Tian, M. B. Blaschko, M. Xing, X. Li, Y. Yue, and M.-F. Moens. Large language models reasoning abilities under non-ideal conditions after rl-fine-tuning. *arXiv preprint arXiv:2508.04848*, 2025.
[38] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
[40] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J. Y. Zhang, and J. ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*.
[41] Y. Wang, H. Wu, J. Dong, Y. Liu, M. Long, and J. Wang. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*, 2024.
[42] Y. Wang, H. Wu, J. Dong, G. Qin, H. Zhang, Y. Liu, Y. Qiu, J. Wang, and M. Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. *arXiv preprint arXiv:2402.19072*, 2024.
[43] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*.
[44] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.
[45] A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.
[46] Y. Zhang and J. Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2023.
[47] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022.
[48] D. Zou, W. Lu, Z. Zhu, X. Lu, J. Zhou, X. Wang, K. Liu, K. Wang, R. Sun, and H. Wang. Optscaler: A collaborative framework for robust autoscaling in the cloud. *Proceedings of the VLDB Endowment*, 17(12): 4090–4103, 2024.

Appendix

We offer some technical details and reproducibility-related information as supplementary materials to help readers understand and reproduce our model.

A Cross Convergence Mapping

Convergent Cross Mapping (CCM) theory is a classical algorithm proposed to detect causal relationships between species. Inspired by this algorithm and preliminary experiments where web services show latent causality, we leverage CCM to enhance traffic prediction. To help readers better understand the CCM, we provide more details in this section.

The exposition of CCM is often illustrated using the context of the Lorenz system. As depicted in Figure 6, the trajectory of the Lorenz system forms a manifold M in the state space. This manifold M consists of a collection of points that represent all possible states of the Lorenz system over time, with these points connected to create a structured geometric space. The manifold, also referred to as the attractor, encompasses all trajectories and potential states $\underline{m}(t)$ of the system. Each state $\underline{m}(t)$ corresponds to a point in M, represented by the coordinate vector $\underline{m}(t) = [X(t), Y(t), Z(t)]$.

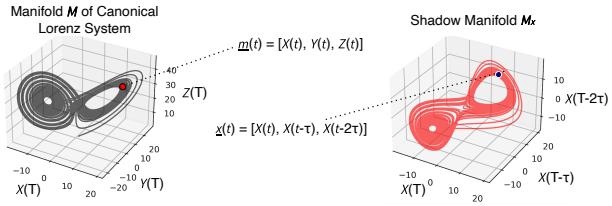


Figure 6: The CCM theory is explained using the canonical Lorenz system. This figure exemplifies the manifold M and its corresponding shadow manifold M_x.

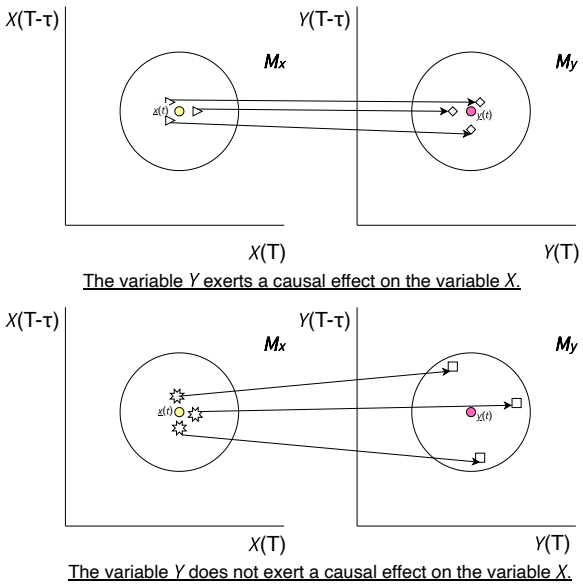


Figure 7: The point $\underline{y}(t)$ in the manifold M_y corresponds to the contemporaneous point in time $\underline{x}(t)$ in the manifold M_x.

The CCM procedure for detecting whether variable Y has causal effects on X consists of four key steps:

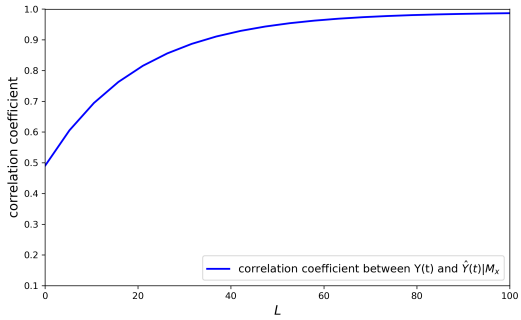


Figure 8: Convergent predictability as the time series length increases, assuming Y has a causal effect on X.

- **Step 1: Construct the shadow manifold M_x.** Consider two time-evolving variables $X(k)$ and $Y(k)$ of length L, where the time index k ranges from 1 to L. The shadow manifold M_x is constructed by forming lagged coordinate vectors:

$$\underline{x}(t) = [X(t), X(t-\tau), X(t-2\tau), \ldots, X(t-(E-1)\tau)],$$

  for $t = 1 + (E-1)\tau$ to $t = L$. Here, τ represents the time lag, and E denotes the embedding dimension.
- **Step 2: Identify nearest neighbors in M_x.** To estimate $Y(t)$ for a specific t in the range $1+(E-1)\tau$ to L, use the shadow manifold M_x. Denote the estimated value as $\hat{Y}(t) \mid M_x$. Begin by locating the contemporaneous lagged coordinate vector $\underline{x}(t)$ in M_x, and find its $E+1$ nearest neighbors. Note that $E+1$ is the minimum number of points needed for a bounding simplex in an E-dimensional space. Let the time indices of these neighbors (ranked by proximity) be $t_1, t_2, \ldots, t_{E+1}$. The nearest neighbors of $\underline{x}(t)$ in M_x are therefore denoted by $\underline{x}(t_i)$, where $i = 1, \ldots, E+1$.
- **Step 3: Estimate $Y(t)$ using locally weighted means.** The time indices $t_1, t_2, \ldots, t_{E+1}$ corresponding to the nearest neighbors of $\underline{x}(t)$ are used to identify points in the variable $Y(k)$. These points are then used to estimate $Y(t)$ through a locally weighted mean of the $E+1$ values $Y(t_i)$:

$$\hat{Y}(t) \mid M_X = \sum_{i=1}^{E+1} w_i Y(t_i),$$

  where w_i represents the weight based on the distance between $\underline{x}(t)$ and its i-th nearest neighbor in M_x, and $Y(t_i)$ are the contemporaneous values of variable $Y(k)$. The weights w_i are determined by:

$$w_i = \frac{u_i}{\sum_{j=1}^{E+1} u_j},$$

  where

$$u_i = \exp\left\{-\frac{d[\underline{x}(t), \underline{x}(t_i)]}{d[\underline{x}(t), \underline{x}(t_1)]}\right\}.$$

  Here, $d[\underline{x}(t), \underline{x}(t_i)]$ denotes the Euclidean distance between the two vectors.
- **Step 4: Calculate the correlation coefficient r.** Finally, calculate the correlation coefficient r between $\hat{Y}(t)$ and $Y(t)$, where t ranges from $1 + (E-1)\tau$ to L:

$$r = \frac{\sum_{t=1+(E-1)\tau}^{L} \left(Y(t) - \overline{Y}(t)\right)\left(\hat{Y}(t) - \overline{\hat{Y}}(t)\right)}{\sqrt{\sum_{t=1+(E-1)\tau}^{L} \left(Y(t) - \overline{Y}(t)\right)^2 \sum_{t=1+(E-1)\tau}^{L} \left(\hat{Y}(t) - \overline{\hat{Y}}(t)\right)^2}}.$$

  If variable Y has causal effects on X, $\hat{Y}(t)$ will converge to $Y(t)$ as the observation period increases. In ideal cases, the correlation coefficient r will approach 1.

B Overall Training Algorithm

Algorithm 1 presents the training algorithm for our CCMPlus module corresponding to the procedure introduced in §4.1, which produce the causality enhanced representation $\widetilde{\mathbf{h}}_{\mathbf{ccm}}$ and causal correlation matrix $\widetilde{\mathbf{M}}^z$, respectively.

C Additional Performance Comparison Results

In this part, we provide additional results of the prediction granularity experiments ($\alpha = 1$ and $\alpha = 15$). It show that with our proposed CCMPlus module, the SOTA models can achieve further improvements on various prediction granularity setting across all the datasets. The t-test also demonstrates the improvements are statistical significant.

Algorithm 1 Algorithmic Procedure for the CCMPlus Module

Require: $\tau = [\tau_1, \ldots, \tau_n]$: list of time lagged parameters, τ_w : time window length, L: input time series length, N: the number of web service within the input, C: the number of input channels for Conv1D, D: the number of output channels of the Conv1D, $\hat{\mathbf{X}} \in \mathbb{R}^{N \times L}$: input time series, $\mathbf{X} \in \mathbb{R}^{N \times L \times C}$: input time series embedding, $\mathbf{M} \in \mathbb{R}^{N \times N}$: causal correlation matrix, m: the momentum value used for updating the causal correlation matrix, constrained within the range $(0, 1)$, training_flag $\in \{\text{True}, \text{False}\}$: training or testing mode.

Ensure: **Output** $\widetilde{\mathbf{h}}_{\mathbf{ccm}}$ and $\widetilde{\mathbf{M}}^z$

Initialize $\mathbf{E}$ as an empty list.
for each τ_i in τ **do**
  $E_i \leftarrow \left\lfloor \frac{\tau_w}{\tau_i} \right\rfloor$
  if E_i is even **then**
    $E_i \leftarrow E_i - 1$
  end if
  Append E_i to $\mathbf{E}$
end for
for $z \leftarrow 0, \ldots, Z$ **do**
  $\mathcal{H} = [], \mathcal{C} = []$
  for $i \leftarrow 1 \ldots n$ **do**
    $\tau_i \leftarrow \tau[i-1], \quad E_i \leftarrow \mathbf{E}[i-1]$
    if $(L - \tau_i \times (E_i - 1)) < 0$ **then**
      continue
    end if
    Reverse the order of $\mathbf{X}$ along the L dimension. Then reshape $\mathbf{X}$ to $\bar{\mathbf{X}} \in \mathbb{R}^{N \times C \times L}$
    $\mathbf{X}_{conv} = \text{Conv1D}\big(\bar{\mathbf{X}};\ \text{kernel\_size} = E_i, \text{dilation} = \tau_i, \big), \mathbf{X}_{conv} \in \mathbb{R}^{N \times D \times \bar{L}}$
    if training_flag = True **then**
      Reshape $\mathbf{X}_{conv}$ to $\mathbf{X}_{ccm} \in \mathbb{R}^{N \times \bar{L} \times D}$, $\mathbf{Y} \leftarrow \hat{\mathbf{X}}[:, -\bar{L}:]$, $\mathbf{Y}$ is reversed along the $\bar{L}$ dimension
      Compute the distance matrix $\mathcal{D} \leftarrow \text{PairwiseDistances}(\mathbf{X}_{ccm}\mathbf{X}_{ccm})$
      Find the nearest neighbor's indices of each web service $\mathcal{I} \leftarrow \text{argsort}(\mathcal{D}, \text{dim} = -1)[:, :, 1 : (D+2)]$
      $d \leftarrow \text{Gather}(\mathcal{D}, \mathcal{I})$
      $u_z \leftarrow \exp\Big(-\frac{d}{d[:,:,0{:}1] + \epsilon}\Big), w_z \leftarrow \frac{u_z}{\sum(u_z) + \epsilon}$
      $Y_{nearest} \leftarrow \text{Gather}(\mathbf{Y}, \mathcal{I}), \hat{Y} \leftarrow \sum(w_z \times Y_{nearest})$
      $corr \leftarrow \frac{\text{Cov}(\hat{Y},\ \mathbf{Y})}{\sigma(\hat{Y})\,\sigma(\mathbf{Y}) + \epsilon}, \mathbf{M} \leftarrow corr^\top$
      if z >0 **then**
        $\widehat{\mathbf{M}}^z(i) \leftarrow \lambda * \widetilde{\mathbf{M}}^{z-1} + (1 - \lambda) * \mathbf{M}$
      else
        $\widehat{\mathbf{M}}^z(i) \leftarrow \mathbf{M}$
      end if
    else
      $\widehat{\mathbf{M}}^z(i) \leftarrow \widetilde{\mathbf{M}}^{z-1}$
    end if
    $\widehat{\mathbf{X}_{ccm}} \leftarrow \text{LinearProjection}(\text{Transpose}(\mathbf{X}_{ccm}))$
    $\mathbf{h}_{ccm}(i) = \widehat{\mathbf{M}}^z(i) \cdot \widehat{\mathbf{X}_{ccm}}$
    Append $h_{ccm}(i)$ to $\mathcal{H}$; Append $\widehat{\mathbf{M}}^z(i)$ to $\mathcal{C}$
  end for
  $\widetilde{\mathbf{h}}_{\mathbf{ccm}} \leftarrow \text{Mean}(\text{Stack}(\mathcal{H}), \text{dim} = 0)$
  $\widetilde{\mathbf{M}}^z \leftarrow \text{Mean}(\text{Stack}(\mathcal{C}), \text{dim} = 0)$
end for

Table 5: Prediction performances averaged over three runs with prediction granularity α set as 15 minutes. The best result is marked in bold. The t-test conducted on both metrics indicates that the improvement is statistical significant (p-value < 0.001).

15 Minutes	Alibaba Group Traffic		Microsoft Azure Traffic		Ant Group Traffic		Overall Mean	
Method	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MagicScaler [25]	3.3695	0.5262	19.2405	0.6764	1.5044	1.0544	8.0381	0.7523
OptScaler [48]	3.4188	0.5950	17.0180	0.7560	1.3069	0.9503	7.2479	0.7671
Llama3 [38]	7.4170	1.1452	12.6471	1.5288	3.3408	1.5700	7.8016	1.4147
TimeLLM [16]	3.3954	0.5242	6.7723	0.6217	1.5176	1.0579	3.8951	0.7346
TimeMixer [40]	2.8915	0.5004	5.8507	0.5522	1.3962	0.9900	3.3795	0.6809
iTransformer [22]	2.8854	0.5118	5.7501	0.5310	1.4017	0.9938	3.3457	0.6789
CCM+iTransformer (ours)	2.8374	0.4932	5.0156	0.4582	1.3127	0.9368	3.0552	0.6294
TimesNet [43]	2.8635	0.4904	5.0550	0.4641	1.3962	0.9876	3.1049	0.6474
CCM+TimesNet (ours)	**2.7151 ↓5.18%**	**0.4823 ↓1.65%**	**4.7434 ↓6.16%**	**0.4422 ↓4.72%**	**1.2951 ↓7.24%**	**0.9199 ↓6.85%**	**2.9179 ↓6.02%**	**0.6148 ↓5.04%**

Table 6: Prediction performances averaged over three runs with prediction granularity α set as 1 minute. The best result is marked in bold. The t-test conducted on both metrics indicates that the improvement is statistical significant (p-value < 0.001).

1 Minute	Alibaba Group Traffic		Microsoft Azure Traffic		Ant Group Traffic		Overall Mean	
Method	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MagicScaler [25]	2.7799	0.4783	6.7159	0.4748	1.9403	1.1674	3.8120	0.7068
OptScaler [48]	3.2586	0.4912	6.5052	0.4693	1.3629	0.9598	3.7089	0.6401
Llama3 [38]	7.6830	1.1101	7.8901	0.6657	3.2022	1.5345	6.2584	1.1034
TimeLLM [16]	3.1173	0.4423	5.3406	0.4513	1.4188	1.0059	3.2922	0.6332
TimeMixer [40]	2.6458	0.4664	2.4406	0.3253	1.3918	0.9800	2.1594	0.5906
iTransformer [22]	2.3571	0.2800	2.3565	0.2875	1.3918	0.9804	2.0351	0.5160
CCM+iTransformer (ours)	2.3677	0.2776	2.3206	0.2782	1.3608	0.9618	2.0164	0.5059
TimesNet [43]	2.2003	**0.2609**	**2.2419**	0.2602	1.3917	0.9791	1.9446	0.5001
CCM+TimesNet (ours)	**2.1979 ↓0.11%**	0.2612 ↑0.01%	2.2503 ↑0.04%	**0.2573 ↓1.09%**	**1.3344 ↓4.12%**	**0.9520 ↓2.77%**	**1.9275 ↓0.88%**	**0.4902 ↓2.00%**