

Probabilistic Hypergraph Recurrent Neural Networks for Time-series Forecasting

Hongjie Chen
Dolby Laboratories
Atlanta, GA, USA
hongjie.chen@dolby.com

Ryan A. Rossi
Adobe Research
San Jose, CA, USA
ryrossi@adobe.com

Sungchul Kim
Adobe Research
San Jose, CA, USA
sukim@adobe.com

Kanak Mahadik
Adobe Research
San Jose, CA, USA
mahadik@adobe.com

Hoda Eldardiry
Virginia Tech
Blacksburg, VA, USA
hdardiry@vt.edu

Abstract

Leveraging graph structures for time-series forecasting has garnered significant attention due to their effective relationship modeling between nodes and their associated time-series. However, in scenarios entities communicate in a broadcasting manner, graph models fall short of pairwise modeling. Hypergraph models address this by capturing beyond-pairwise interactions among node time-series. Nevertheless, most hypergraph models overlook the dynamics between nodes and their incident hyperedges, assuming constant node-hyperedge connections. In this paper, we introduce a novel model, **Probabilistic Hypergraph Recurrent Neural Networks (PHRNN)**, which leverages node-hyperedge dynamics for accurate time-series forecasting. PHRNN associates each time-series with a node and models node interactions on a hypergraph, capturing beyond-pairwise interactions. Moreover, PHRNN learns a probabilistic hypergraph in which node-hyperedge relations are modeled as probabilistic distributions instead of fixed values, capturing dynamic node-hyperedge relations. PHRNN further integrates a prior knowledge KNN hypergraph as regularization when learning the probabilistic hypergraph structure. To the best of our knowledge, PHRNN is the first time-series forecasting model that incorporates hypergraph modeling and probabilistic relationship modeling. Forecasting results from extensive experiments show that PHRNN outperforms state-of-the-art graph and hypergraph baselines on real-world datasets.

CCS Concepts

• Information systems → Data mining.

Keywords

Time-series Forecasting, Probabilistic Modeling, Hypergraph Neural Networks

ACM Reference Format:

Hongjie Chen, Ryan A. Rossi, Sungchul Kim, Kanak Mahadik, and Hoda Eldardiry. 2025. Probabilistic Hypergraph Recurrent Neural Networks for

This work is licensed under a Creative Commons Attribution International 4.0 License.

KDD '25, August 3–7, 2025, Toronto, ON, Canada
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1245-6/25/08
<https://doi.org/10.1145/3690624.3709202>

Time-series Forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25)*, August 3–7, 2025, Toronto, ON, Canada. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3690624.3709202>

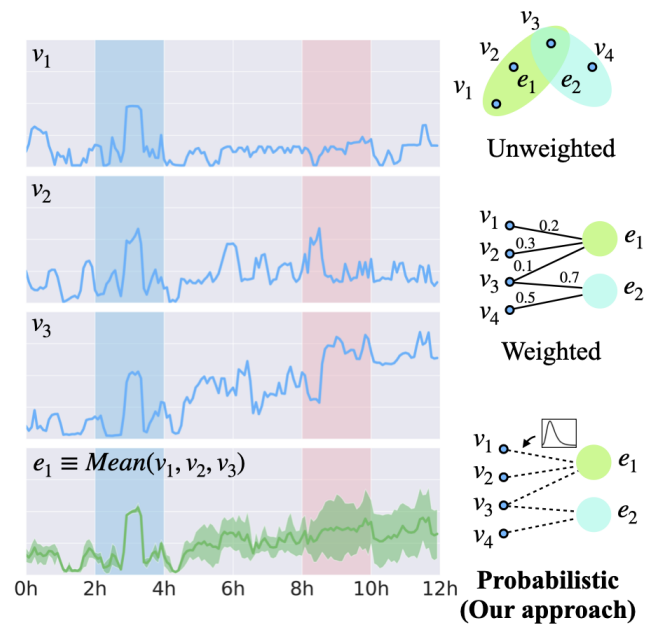


Figure 1: Dynamics of node-hyperedge connections. The time-series of nodes (v_1, v_2, v_3) and their hyperedge e_1 are depicted on the left. The green area around the hyperedge time-series describes the standard deviation. Node time-series exhibit higher correlations with hyperedge time-series in the blue (left shadow) area than in the red (right shadow) area, indicating fluctuations in the connection strength. Hence, probabilistic relationship modeling better captures interaction dynamics than deterministic relationship modeling, whether unweighted or weighted.

1 Introduction

Time-series forecasting is crucial for various applications such as traffic forecasting, website visit estimates, and workload prediction [4, 5, 14, 15, 19]. Early forecasting models are mainly statistical, such as ARIMA and exponential smoothing, among many others [8, 22, 30]. These models focus on small groups of time-series and often overlook inter-series relationships. With the increasing availability of datasets, research directions have shifted toward DL-based models for their superior performance in both univariate and multivariate time-series datasets [1, 3, 12, 46]. Many DL-based methods leverage inter-relationships between time-series. For example, recent approaches employ graph structures, among which Graph Neural Networks (GNN) models have garnered significant attention for their ability to model complex relationships [6, 16, 29, 38]. In a typical GNN time-series model, each node is associated with an individual time-series, and each edge represents a connection between time-series. The mutual relationships between nodes are captured and leveraged through graph convolution or diffusion.

Despite their successes, two challenges remain in existing GNN models. The first challenge lies in modeling beyond-pairwise relationships among nodes. Conventional GNN models are limited to modeling pairwise relationships [10, 20]. However, beyond-pairwise connections are ubiquitous. For instance, beyond-pairwise connections exist among computing nodes in cloud systems, where multiple computing nodes work in cooperation [18]. When nodes collaborate, messages are typically broadcast among them in simultaneous communications instead of one-on-one interactions. GNN models oversimplify broadcasting into pairwise connections. In contrast, hypergraph modeling naturally represents the beyond-pairwise relationships among nodes [32]. The second challenge is related to the deterministic structure adopted by existing models. These models assume that the leveraged graph or hypergraph structure is statically fixed, and connection weights among nodes remain constant. However, the underlying inter-node or node-hyperedge relationships may follow a distribution, with dynamic connection weights sampled from the distribution. To address this challenge, probabilistic **graph** models have been proposed, which model connections between nodes as probabilistic distributions, resulting in significant performance improvement [24, 28]. Following this thread, we introduce a novel probabilistic **hypergraph** approach. An example of three time-series and their dynamics with a hyperedge is presented in Fig. 1. For illustration purposes, we define hyperedge time-series as the mean time-series of its nodes. Notably, the node-hyperedge relationships are dynamic, as nodes exhibit varying correlations with the hyperedge over time (e.g., the blue area versus the red area). Our probabilistic modeling approach aligns with the nature of node interactions and intuitively captures these dynamics, compared to existing hypergraph modeling approaches (either unweighted or weighted), as they assume deterministic node-hyperedge weights.

Our model incorporates both probabilistic modeling and hypergraph modeling, and earns the following advantages: (1) PHRNN is a hypergraph-based model, enabling it to capture beyond-pairwise interactions among nodes. (2) PHRNN does not require an explicit hypergraph structure but learns a hypergraph structure from time-series features. (3) PHRNN models node-hyperedge interactions as

probabilistic distributions, allowing it to exploit the dynamics of node-hyperedge relationships. Our key contributions are,

- We introduce a novel time-series forecasting model, Probabilistic Hypergraph Recurrent Neural Networks (PHRNN), which leverages the inter-relationships among time-series by learning a probabilistic hypergraph. The probabilistic hypergraph approach benefits our model with the advantages of both probabilistic modeling and hypergraph modeling. To the best of our knowledge, PHRNN is the first probabilistic hypergraph model proposed for time-series forecasting. The novel probabilistic hypergraph modeling within our proposed PHRNN serves as an adaptable spatial component.
- We conduct extensive experiments with closely related baselines on multiple datasets. Our experimental results show that PHRNN outperforms the baselines in terms of forecasting accuracy. We investigate the effectiveness of PHRNN on different hyperparameters, including prior knowledge hypergraph constructions and regularization coefficients.

2 Related Work

GNN-based time-series forecasting models have recently surged in popularity [19, 35, 43, 45, 47]. While the majority of existing graph models leverage explicit graph structures that are derived from geographical information [19, 43, 47], there is a growing trend in graph structure learning, which frees models from relying on connectivity information. Graph structure learning is useful when an explicit graph structure is missing or is challenging to define. Models along this line either learn graph structures from time-series features or from embeddings [28, 34, 35]. However, most of them learn a deterministic graph where a static weight is learned for each edge [25, 37]. This assumes that the weights between nodes are fixed, and may not accurately reflect real-world scenarios, as weights between node time-series fluctuate and evolve over time. In order to better reflect these dynamics, some models learn a probabilistic graph structure where each edge represents a distribution, and edge weights are sampled from these distributions [28, 44]. To leverage this advantage, our proposed PHRNN embraces a probabilistic approach by modeling interactions among nodes as distributions.

In addition to the graph time-series forecasting models, efforts have also been made in hypergraph time-series modeling [39, 40, 42, 48]. A hypergraph comprises nodes and hyperedges, where each hyperedge represents a set of nodes and describes connections among nodes that extend beyond pairwise relationships. Compared to graph structures, hypergraphs can represent more complex relationships among nodes [7, 32]. Many existing hypergraph models aim to capture inherent connections among groups or communities of nodes [21, 41, 49]. For example, Wu et al. [33] introduced a hypergraph time-series forecasting model that leverages relational modeling with the aid of hyperedge categories. In contrast, our proposed PHRNN does not assume hyperedge categories and is free from such constraints. Zhao et al. [49] captured the traffic network isomorphism by constructing hyperedges based on actual network connections. Unlike these models that fall short of requiring explicit hypergraph structures, our proposed PHRNN benefits from learning a probabilistic hypergraph structure, which is advantageous when connectivity information is unreliable or missing.

Table 1: A Summary of Notations

Notations	Descriptions	Notations	Descriptions
$\mathbf{X} \in \mathbb{R}^{N \times T}$	time-series data	T	length of time-series
\mathbf{x}^t	time values at t	\mathbf{x}_v	time values of v
$H(V, E, C)$	hypergraph	$N = V $	number of nodes
V	node set	$M = E $	number of edges
E	hyperedge set	$E(v)$	edges that v incidence to
C	incidence matrix	λ_{reg}	regularization coefficient
l	lag of time-series	τ	prediction horizon
θ	parameters	g	Gumbel distribution
s	temperature	\oplus	concatenation operator
\mathbf{z}_v	node embedding	\mathbf{y}_e	hyperedge embedding
FC (\cdot)	a dense layer	Conv (\cdot)	a convolutional layer

3 Problem Formulation

We target a multivariate time-series forecasting problem, where our objective is to predict future values for a set of N time-series. Let $\mathbf{X} \in \mathbb{R}^{N \times T}$ denote the matrix comprising N time-series, each with a length of T . With these time-series values, we initially learn a **probabilistic hypergraph**, denoted by $H(V, E, C)$.

$$\mathbf{X}^{1:T} \rightarrow H(V, E, C) \quad (1)$$

The letters V and E denote a node set and a hyperedge set, respectively. Each node $v \in V$ is associated with a time-series $\mathbf{x}_v \in \mathbb{R}^T$, resulting in $N = |V|$. Furthermore, we let $\mathbf{x}^t \in \mathbb{R}^N$ denote the time-series values for all nodes at time t . Each hyperedge $e \in E$ is incident to a subset of the node set V . We let $M = |E|$ denote the number of hyperedges. The letter $C \in \mathbb{R}^{N \times M}$ denotes the probabilistic incidence matrix for the hypergraph H , where each column $C_{:,e} \in \mathbb{R}^N$ describes the incident information of a hyperedge e , signifying node-hyperedge connections between all nodes and hyperedge e . We aim to learn a probabilistic distribution for each element $C_{v,e}$ in the incidence matrix, which subsequently represents the node-hyperedge dynamics. Once a probabilistic hypergraph H is learned, our primary objective is to determine a function f that predicts future time-series values $\hat{\mathbf{X}}^{T+1:T+\tau}$ of a horizon (e.g., τ steps ahead). The prediction is based on the historical time-series observations $\mathbf{X}^{1:T}$ in conjunction with the learned hypergraph H :

$$\left[\mathbf{X}^{1:T}, H \right] \xrightarrow{f(\cdot)} \hat{\mathbf{X}}^{T+1:T+\tau} \quad (2)$$

A table of notations is provided in Table 1.

4 Probabilistic Hypergraph Recurrent Neural Networks

Our proposed Probabilistic Hypergraph Recurrent Neural Networks (PHRNN) adopts an encoder-decoder structure, where the encoder recursively digests input sequence values to learn encoded states for nodes, while the decoder utilizes these states to recursively make predictions. In contrast to previous graph-based methods, PHRNN does not require an explicit graph to model the inter-relations between nodes. Instead, we introduce a probabilistic hypergraph learning component that automatically learns the underlying hypergraph structure from time-series. Once the hypergraph is learned, it is used to update node time-series embeddings through a node-to-hyperedge aggregation and a hyperedge-to-node aggregation.

Subsequently, these updated node embeddings are used by a predictor to generate final predictions.

4.1 Probabilistic Hypergraph Learning

To overcome the limitations of existing hypergraph methods that they rely on explicit hypergraphs with fixed incidence matrix values [39, 49], PHRNN learns a probabilistic hypergraph structure that parameterizes the hypergraph representation. We parameterize C through the parameters $\theta \in [0, 1]^{N \times M}$ using the Gumbel reparameterization trick, which allows efficient differentiability [11, 28]. Hence, the distribution of an element $C_{v,e}$ can be described by

$$C_{v,e} = \sigma \left(\frac{\log \left(\frac{\theta_{v,e}}{1-\theta_{v,e}} + (g_{v,e}^1 - g_{v,e}^2) \right)}{s} \right) \quad (3)$$

where $\sigma(\cdot)$ denotes a sigmoid function. $g_{v,e}^1$ and $g_{v,e}^2$ denote two standard Gumbel distributions $Gumbel(\theta, 1)$, and s is a temperature hyperparameter that controls the sparsity of the incidence matrix. During the hypergraph structure optimization, we independently sample $g_{v,e}^1$ and $g_{v,e}^2$ from the standard Gumbel distribution, which consequently instantiates the node-hyperedge weights $C_{v,e}$ for node v and hyperedge e . For brevity, we let $C_{v,e}$ denote both the distribution and the instantiation for the incidence matrix, as context will clarify any ambiguity.

Calculation of parameters θ . In Gumbel reparameterization described in Eq. 3, the parameter θ depends on pairs of node-hyperedges. More precisely, the value of $\theta_{v,e}$ depends on node v and hyperedge e . Therefore, we propose calculating θ from node embeddings and hyperedge embeddings. We derive node embeddings from time-series values. Let \mathbf{z}_v denote the node embedding for node $v \in V$. We compute \mathbf{z}_v through a fully connected layer FC (\cdot) and a convolutional layer Conv (\cdot) over time-series \mathbf{x}_v :

$$\text{Node embeddings: } \mathbf{z}_v = \text{FC}(\text{Conv}(\mathbf{x}_v)) \in \mathbb{R}^{d_z} \quad (4)$$

The use of the convolutional layer over the entire time span of time-series allows node embeddings to capture the temporal relations across a wide range of time steps.

Hyperedge embeddings, on the other hand, cannot be directly derived from time-series due to the missing hypergraph structure information. To address this, we leverage a K-Nearest-Neighbors (KNN) prior knowledge hypergraph to initialize hyperedge embeddings. The KNN-based hypergraph is constructed by iterating over each node, where at each iteration a hyperedge is formed by including the iterated ego-node and its $K-1$ closest neighbors in terms of time-series similarity. Redundant hyperedges are further removed, resulting in $M \leq N$ hyperedges. An example of 3 hyperedges is presented in Fig. 2 (a). Let $H^K(V, E^K)$ denote the KNN-based prior knowledge hypergraph, we define the hyperedge embedding \mathbf{y}_e as the aggregation of node embeddings for nodes incident to e :

$$\text{Hyperedge embeddings: } \mathbf{y}_e = \frac{1}{|e|} \sum_{v \in e} \mathbf{z}_v, \quad e \in E^K \quad (5)$$

As depicted in Fig. 2 (b), we employ mean aggregation, but other aggregation methods such as attention are also plausible [9].

Given node embedding \mathbf{z}_v and hyperedge embedding \mathbf{y}_e , a node-hyperedge incidence predictor is needed to derive a real value for

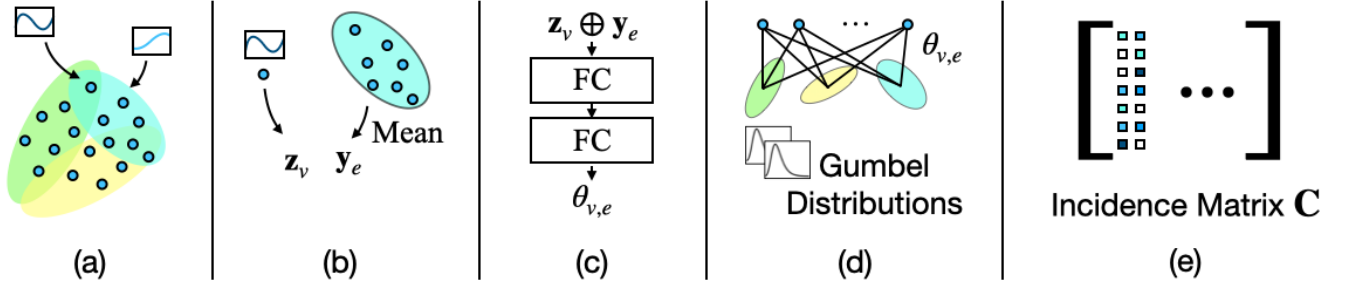


Figure 2: Probabilistic hypergraph structure learning. (a) PHRNN utilizes a prior knowledge KNN hypergraph. A hyperedge is derived by taking each node and its closest $K - 1$ nodes in terms of time-series similarity. (b) A node embedding z_v is calculated for each node v using Eq. 4. A hyperedge embedding is calculated by averaging all node embeddings the hyperedge is incident to. (c) Using the node embeddings and hyperedge embeddings, the hypergraph parameter $\theta_{v,e}$ is derived through a two-layer neural module. (d) $\theta_{v,e}$ is computed for each node-hyperedge pair. Gumbel distributions are used with $\theta_{v,e}$ to form a probabilistic distribution for each node-hyperedge weight. (e) The learned hypergraph structure is represented in the format of an incidence matrix, where each element represents the sampled node-hyperedge weight from a distribution.

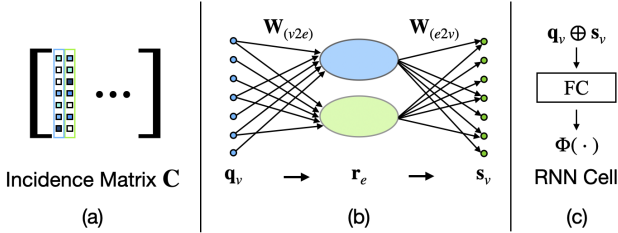


Figure 3: The hypergraph-based RNN cell. (a) The sampled hypergraph is used to determine hyperedges. (b) Two hyperedges are highlighted for the node-hyperedge-node aggregations. Each hyperedge is first aggregated from its incident nodes, as described in Eq. 7. Each node is then aggregated from its incident hyperedges, as described in Eq. 8. (c) The cell function $\Phi(\cdot)$ outputs a hidden state for each node by applying a fully connected layer to the original node vector q and the transformed node vector s_v .

$\theta_{v,e}$. We propose using a two-layer module, as depicted in Fig. 2 (c):

$$\theta_{v,e} = \text{FC}(\text{FC}(z_v \oplus y_e)) \quad (6)$$

Our incidence predictor has a simple yet effective structure but other readout functions are also viable [34, 36]. Subsequently, the parameter $\theta_{v,e}$ is computed for each pair of node v and hyperedge e , as shown in Fig. 2 (d). In conjunction with the Gumbel distributions from Eq. 3, the parameter θ is used to derive the probabilistic hypergraph structure C , as shown in Fig. 2 (e).

4.2 Hypergraph-based RNN Cells

With the learned probabilistic hypergraph structure from Eq. 3, we design a hypergraph-based RNN cell that serves as the basic unit for our encoder-decoder framework. Inspired by HGC-RNN [39], our cell takes node time-series and hidden states for nodes as input, and outputs updated hidden states, through a node-hyperedge-node update route. An example of two hyperedges are depicted in

Fig. 3 (a). For the clarity of writing, we let $q_v \in \mathbb{R}^{d_q}$ denote the input time-series vector for node v to our RNN cell. Each RNN cell consists of two stages, a node-to-hyperedge aggregation stage and a hyperedge-to-node aggregation stage, as depicted in Fig. 3 (b). In the node-to-hyperedge aggregation stage, we aggregate each node time-series vector to its incident hyperedges. Let $r_e \in \mathbb{R}^{d_r}$ denote the vector for hyperedge $e \in E$ through a mean aggregator.

$$\text{v-to-e aggregation: } r_e = \frac{1}{|e|} \cdot \sigma \left(\sum_{v \in e} W_{(v2e)}^T \cdot q_v \right), \forall e \in E \quad (7)$$

where $W_{(v2e)}^T \in \mathbb{R}^{d_q \times d_r}$ denotes a weight layer that transforms node vectors into hyperedge vectors. In the hyperedge-to-node stage, the updated hyperedge vectors are aggregated to produce a new node vector, denoted as $s_v \in \mathbb{R}^{d_s}$ for each node v :

$$\text{e-to-v aggregation: } s_v = \frac{1}{|E(v)|} \cdot \sigma \left(\sum_{e \in E(v)} W_{(e2v)}^T \cdot r_{E(v)} \right) \quad (8)$$

where $W_{(e2v)}^T \in \mathbb{R}^{d_r \times d_s}$ denotes a weight layer that transforms hyperedge vectors into node vectors. $E(v)$ denotes hyperedges that are incident to node v . We adopt a residual structure and apply a fully connected layer to the concatenation of the original node vectors and the aggregated node vectors, as depicted in Fig. 3 (c). Let $\Phi(\cdot)$ denote the RNN cell in our PHRNN model:

$$\Phi(q_v) = \text{FC}(q_v \oplus s_v) \in \mathbb{R}^{d_\Phi} \quad (9)$$

4.3 A Hypergraph Encoder-decoder Framework

Inspired by previous work [19, 28, 36, 39], our PHRNN utilizes an encoder-decoder structure. The encoder recursively consumes historical observations to learn a hidden state, which is subsequently used by the decoder for recursive forward predictions. Our hypergraph encoder-decoder framework is depicted in Fig. 4, where both the encoder and the decoder are built with the hypergraph-based RNN cell described in Sec. 4.2. Let $H^t \in \mathbb{R}^{N \times d}$ denote the hidden state at time step t , where d denotes the dimension of the hidden

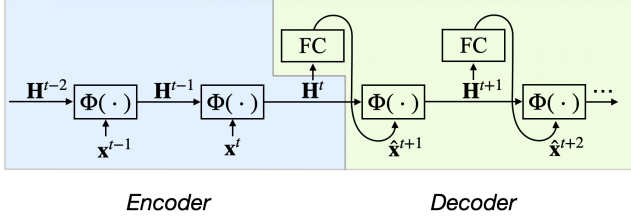


Figure 4: The hypergraph encoder-decoder framework. The encoder comprises hypergraph RNN cells $\Phi(\cdot)$ described by Eq. 9 that recursively consume historical time-series values. Given the encoded states, the decoder utilizes hypergraph RNN cells to yield forecasts for all nodes through a fully connected layer.

Algorithm 1 PHRNN Model Training

- 1: Initialize model parameters $\text{FC}_*(\cdot)$, $\text{Conv}_*(\cdot)$, \mathbf{W}_* , etc.
- 2: **for** each node v and each hyperedge e in the prior knowledge graph H^K **do**
- 3: With the time-series values \mathbf{x}_v
- 4: Calculate the node embedding \mathbf{z}_v in Eq. 4
- 5: Calculate the hyperedge embedding \mathbf{z}'_e in Eq. 5
- 6: Calculate the parameter $\theta_{v,e}$ in Eq. 6
- 7: Sample a value for the incidence matrix $\mathbf{C}_{v,e}$ in Eq. 3
- 8: **end for**
- 9: **for** all time-series \mathbf{x}^t at timestamp t in a batch **do**
- 10: With current model parameter estimates
- 11: Encode \mathbf{x}^t with Hypergraph-based RNN cells in Eq. 10
- 12: Decode from the encoded hidden state \mathbf{H}^t in Eq. 11 to give time-series predictions
- 13: In the current batch, calculate the time-series loss in Eq. 12 and the regularization loss in Eq. 13. Perform stochastic gradient descent to update the trainable parameters accordingly.
- 14: **end for**

state features. The hidden state \mathbf{H}^t is computed from the time-series values \mathbf{x}^t and the previous hidden state \mathbf{H}^{t-1} . Hence, the input variable \mathbf{q}_v in Eq. 9 is substituted accordingly, as $\mathbf{q}_v \equiv \mathbf{x}^t \oplus \mathbf{H}^{t-1}$,

$$\mathbf{H}^t = \Phi(\mathbf{x}^t \oplus \mathbf{H}^{t-1}) \in \mathbb{R}^{d_\Phi} \quad (10)$$

Prediction is obtained by decoding the final hidden state \mathbf{H}^t using a projection layer. We employ a fully connected layer as the projection layer, but other structures are viable [29]:

$$\hat{\mathbf{x}}^{t+1} = \text{FC}(\mathbf{H}^t) \quad (11)$$

Prediction for future time steps is computed by recursively feeding the predicted output as input to the model.

PHRNN takes an end-to-end supervised training approach that aims to minimize the loss between the ground truth time-series values and the predictions. We use the Mean Absolute Error (MAE) as the training loss during the model optimization:

$$\text{Loss}_{ts} = \frac{1}{|\Omega_{tr}|} \frac{1}{N} \frac{1}{\tau} \sum_{t \in \Omega_{tr}} \sum_{i=1}^N \sum_{j=1}^{\tau} |x_i^{t+j} - \hat{x}_i^{t+j}| \quad (12)$$

Table 2: Statistics of Experimental Datasets

Data	$ V $	$ E $	Density	Min. Deg.	Mdn. Deg.	Ave. Deg.	Max. Deg.	Time-scale
Adobe	1,135	332	0.88%	1	1	2.92	38	30 min
Google	845	838	1.18%	1	7	9.89	57	5 min
Traffic	862	828	1.02%	1	7	8.45	31	1 hour

where Ω_{tr} denotes the set of training samples. Additionally, we integrate a regularization loss on the learned hypergraph. Previous work has shown that incorporating a prior knowledge (hyper)graph potentially improves forecasting performance [28, 44]. The prior knowledge graph can be derived from external connectivity information, such as geographic distance, or from time-series patterns. In PHRNN, we reuse the KNN hypergraph in hyperedge embeddings, described in Sec. 4.1, for the purpose of regularization. Each hyperedge in the KNN hypergraph is constructed based on a node and its $K-1$ closest nodes in terms of time-series similarity, which is aligned with the assumption that time-series with similar patterns are more related to each other than other time-series [5]. We use the element-wise cross entropy between the learned hypergraph and the KNN hypergraph as the regularization loss:

$$\text{Loss}_C = -\frac{1}{N} \frac{1}{M} \sum_{v \in V} \sum_{e \in E} \left(C_{v,e} \log(C_{v,e}^K) + (1 - C_{v,e}) \log(1 - C_{v,e}^K) \right) \quad (13)$$

Our final loss function is a combination of the time-series loss and the hypergraph loss using a regularization coefficient λ_{reg} as a weighting factor:

$$\text{Loss} = \text{Loss}_{ts} + \lambda_{reg} \text{Loss}_C \quad (14)$$

The workflow of PHRNN is described in Algorithm 1.

5 Experimental Results

We conduct extensive experiments on real-world datasets to evaluate the forecasting accuracy of our PHRNN model. Furthermore, we perform a hyperparameter analysis to investigate the effectiveness of prior knowledge hypergraph.

5.1 Datasets

In our experiments, we select three datasets that cover applications of cloud computing and traffic networks. *Adobe Cloud Trace*. This dataset contains 1,135 time-series of CPU utilization rates recorded from October to December in 2018, with a time interval of half an hour. *Google Cloud Trace*. The Google trace dataset records the CPU usage of tens of thousands of computing nodes in May of 2011 [26, 27]. We select a subset of highly active nodes, i.e., those with utilization rates exceeding 25% for more than 70% of their lifetime, resulting in 845 time-series of CPU utilization rates. *Traffic Occupancy*. This dataset contains 862 time-series of road occupancy rates recorded in San Francisco from 2015 to 2016, with a time interval of one hour [17]. We also include results for a larger dataset in Sec. A due to space limit. All time-series values of these datasets fall within the range of $[0, 1]$. To evaluate the robustness of our model on various time-series patterns, we divide each dataset into 10 folds. For each fold, we randomly select a period to be split into a training set, a validation set, and a testing set in a 2 : 1 : 1

Table 3: Results for One-step ahead Forecasting (MAE and RMSE)

Metrics	DATA	HGC-RNN	DGSL	RGSL	MTGNN	STGCN	StemGNN	MegaCRN	ESG	PHRNN (Ours)
MAE	ADOBE	0.052 ± 0.021	0.071 ± 0.019	0.064 ± 0.023	0.052 ± 0.020	0.061 ± 0.022	0.059 ± 0.020	0.041 ± 0.026	0.044 ± 0.017	0.039 ± 0.021
	GOOGLE	0.064 ± 0.024	0.063 ± 0.012	0.062 ± 0.015	0.065 ± 0.016	0.071 ± 0.022	0.077 ± 0.020	0.062 ± 0.010	0.061 ± 0.013	0.060 ± 0.022
	TRAFFIC	0.026 ± 0.015	0.060 ± 0.021	0.030 ± 0.015	0.025 ± 0.021	0.029 ± 0.016	0.038 ± 0.019	0.021 ± 0.010	0.027 ± 0.011	0.013 ± 0.011
RMSE	ADOBE	0.101 ± 0.037	0.128 ± 0.032	0.119 ± 0.041	0.100 ± 0.036	0.117 ± 0.039	0.111 ± 0.036	0.085 ± 0.042	0.087 ± 0.029	0.075 ± 0.034
	GOOGLE	0.082 ± 0.028	0.082 ± 0.013	0.080 ± 0.018	0.081 ± 0.018	0.092 ± 0.025	0.096 ± 0.022	0.079 ± 0.011	0.078 ± 0.015	0.077 ± 0.025
	TRAFFIC	0.040 ± 0.020	0.076 ± 0.026	0.043 ± 0.023	0.038 ± 0.028	0.044 ± 0.024	0.053 ± 0.025	0.034 ± 0.019	0.043 ± 0.018	0.019 ± 0.020

Table 4: Results for Multi-step ahead Forecasting (MAE and RMSE)

Metrics	DATA	H	HGC-RNN	DGSL	RGSL	MTGNN	STGCN	StemGNN	MegaCRN	ESG	PHRNN (Ours)	
MAE	A	3	0.059 ± 0.020	0.071 ± 0.019	0.062 ± 0.022	0.052 ± 0.021	0.061 ± 0.021	0.059 ± 0.021	0.049 ± 0.025	0.047 ± 0.018	0.049 ± 0.023	
		6	0.059 ± 0.021	0.069 ± 0.018	0.064 ± 0.023	0.054 ± 0.022	0.061 ± 0.021	0.058 ± 0.021	0.050 ± 0.024	0.052 ± 0.019	0.051 ± 0.023	
		12	0.058 ± 0.021	0.068 ± 0.018	0.060 ± 0.023	0.055 ± 0.019	0.060 ± 0.024	0.058 ± 0.021	0.054 ± 0.021	0.054 ± 0.019	0.053 ± 0.022	
	G	3	0.066 ± 0.012	0.062 ± 0.010	0.063 ± 0.012	0.067 ± 0.022	0.068 ± 0.016	0.068 ± 0.014	0.063 ± 0.016	0.062 ± 0.013	0.061 ± 0.014	
		6	0.069 ± 0.010	0.065 ± 0.008	0.063 ± 0.011	0.080 ± 0.039	0.064 ± 0.010	0.068 ± 0.009	0.063 ± 0.014	0.063 ± 0.013	0.062 ± 0.010	
		12	0.072 ± 0.016	0.068 ± 0.016	0.073 ± 0.012	0.081 ± 0.034	0.068 ± 0.013	0.069 ± 0.012	0.068 ± 0.020	0.064 ± 0.015	0.067 ± 0.016	
	T	3	0.036 ± 0.012	0.061 ± 0.017	0.031 ± 0.009	0.038 ± 0.023	0.033 ± 0.015	0.036 ± 0.016	0.030 ± 0.010	0.029 ± 0.010	0.029 ± 0.009	
		6	0.037 ± 0.010	0.065 ± 0.017	0.037 ± 0.019	0.041 ± 0.017	0.034 ± 0.013	0.042 ± 0.014	0.036 ± 0.011	0.034 ± 0.011	0.034 ± 0.008	
		12	0.033 ± 0.012	0.068 ± 0.019	0.033 ± 0.010	0.036 ± 0.017	0.035 ± 0.014	0.035 ± 0.015	0.035 ± 0.011	0.030 ± 0.008	0.033 ± 0.012	
	RMSE	A	3	0.110 ± 0.035	0.128 ± 0.031	0.115 ± 0.039	0.101 ± 0.035	0.118 ± 0.036	0.112 ± 0.036	0.092 ± 0.040	0.092 ± 0.030	0.092 ± 0.038
			6	0.111 ± 0.037	0.126 ± 0.031	0.119 ± 0.039	0.103 ± 0.035	0.117 ± 0.037	0.111 ± 0.036	0.094 ± 0.037	0.098 ± 0.030	0.097 ± 0.038
			12	0.110 ± 0.037	0.125 ± 0.031	0.113 ± 0.039	0.106 ± 0.033	0.125 ± 0.057	0.110 ± 0.036	0.101 ± 0.035	0.103 ± 0.033	0.100 ± 0.036
G		3	0.080 ± 0.014	0.081 ± 0.011	0.081 ± 0.015	0.085 ± 0.024	0.089 ± 0.018	0.086 ± 0.016	0.080 ± 0.018	0.079 ± 0.013	0.078 ± 0.015	
		6	0.084 ± 0.012	0.080 ± 0.008	0.083 ± 0.013	0.101 ± 0.043	0.083 ± 0.011	0.085 ± 0.010	0.080 ± 0.015	0.080 ± 0.014	0.079 ± 0.011	
		12	0.086 ± 0.018	0.087 ± 0.018	0.088 ± 0.015	0.101 ± 0.040	0.089 ± 0.015	0.088 ± 0.013	0.086 ± 0.024	0.083 ± 0.018	0.085 ± 0.019	
T		3	0.051 ± 0.019	0.077 ± 0.022	0.040 ± 0.013	0.053 ± 0.029	0.050 ± 0.024	0.052 ± 0.024	0.047 ± 0.020	0.047 ± 0.020	0.045 ± 0.018	
		6	0.053 ± 0.018	0.081 ± 0.023	0.051 ± 0.027	0.057 ± 0.024	0.050 ± 0.022	0.059 ± 0.020	0.053 ± 0.018	0.051 ± 0.019	0.050 ± 0.015	
		12	0.048 ± 0.017	0.083 ± 0.021	0.046 ± 0.015	0.049 ± 0.022	0.045 ± 0.018	0.048 ± 0.019	0.049 ± 0.016	0.043 ± 0.015	0.047 ± 0.017	

ratio. Following commonly adopted configurations [28, 34], we construct data points by setting a time lag $l = 12$ and a prediction horizon $\tau = 12$. For each period, we generate a prior knowledge hypergraph with $K = 10$ from time-series values, and then train a model instance. Consequently, we train 10 model instances for our PHRNN and for each baseline. A summary of the prior knowledge hypergraph statistics is provided for each dataset in Table 2.

5.2 Baselines

We select highly related graph and hypergraph models as baselines.

- *HGC-RNN* [39] utilizes an explicit hypergraph for time-series forecasting. HGC-RNN also learns from a node-to-hyperedge aggregation and a hyperedge-to-node aggregation to predict the node-level time-series.

We also include probabilistic graph structure learning models as baselines. These methods model the pairwise interactions between nodes as probabilistic distributions.

- *DGSL* [28] exploits the pairwise information among time-series by optimizing the mean performance over the graph distribution. The probabilistic graph structure is learned based on the historical values of time-series.

- *RGSL* [44] fuses both implicit and explicit graphs, where the implicit graph is learned from node embeddings, and the explicit graph is rendered from domain knowledge. We derive the explicit graph from the KNN hypergraph by connecting all nodes within each hyperedge.

Other non-probabilistic graph-based models are also included:

- *MTGNN* [34] is a relational learning model for time-series forecasting that learns the uni-directed relations among nodes through node embeddings.
- *STGCN* [43] consists of a spatial layer that extracts relationships between nodes and a temporal layer that learns temporal dependencies.
- *StemGNN* [2] predicts time-series values by combining graph Fourier transform and discrete Fourier transform to learn the intra-series correlations and inter-series correlations in the spectral domains.
- *ESG* [38] is a graph structural learning method that builds a hierarchical graph for multi-scale temporal correlations.
- *MegaCRN* [13] is another graph structural learning approach that learns a mega-graph to capture the interactions between time-series.

For hyperparameters settings, we adopt the recommended hyperparameters from the papers of baselines. Readers may refer to papers of baselines for detailed configurations. As for PHRNN, we conducted a grid search and configure hyperparameters with the following values: $d = d_z = d_s = d_\phi = 16$, $d_r = 32$. We select $K = 10$ and Euclidean distance when generating prior knowledge hypergraphs. We adopt $s = 0.25$ for the temperature and $\lambda_{reg} = 0.02$ for the weight factor.

5.3 Forecasting Performance

We conduct experiments to evaluate the forecasting accuracy regarding one-step ahead prediction and multi-step ahead prediction of our model. For PHRNN and each baseline, we report the mean values and standard deviation of selected loss metrics, including the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) from their 10 model instances. Specifically, we report MAE and RMSE for a prediction step H (e.g., $H = 1$ for one-step ahead prediction):

$$\text{MAE}^H = \frac{1}{|\Omega_{te}|} \frac{1}{N} \sum_{t \in \Omega_{te}} \sum_{i=1}^N |x_i^{t+H} - \hat{x}_i^{t+H}| \quad (15)$$

$$\text{RMSE}^H = \frac{1}{|\Omega_{te}|} \frac{1}{N} \sum_{t \in \Omega_{te}} \sum_{i=1}^N \sqrt{(x_i^{t+H} - \hat{x}_i^{t+H})^2} \quad (16)$$

where Ω_{te} denotes the set of testing samples. For one-step ahead prediction, our proposed PHRNN outperforms all other baselines by a significant margin on the selected datasets, as shown in Table 3. In addition to PHRNN, the only hypergraph baseline model HGC-RNN also exhibits comparatively better performance than most other baselines, underscoring the positive impact of hypergraph modeling on forecasting accuracy. In contrast, the probabilistic graph model DGSL [28] achieves the least favorable results among the compared models in the Adobe and Traffic dataset, suggesting that only using probabilistic graph modeling may not precisely capture the interactions among node time-series. For multi-step ahead prediction, we assess prediction performance for horizons of 3, 6 and 12, as presented in Table 4. Notably, PHRNN achieves superior performance across most datasets and horizons. Despite PHRNN exhibits slightly subpar RMSE results for prediction horizons $H = 3$ and $H = 12$ on the traffic dataset compared to the baseline, this observation may be attributed to the presence of more outlier points in the traffic dataset, which contribute to larger RMSE losses for PHRNN [31]. Additionally, we observe that some models predict more accurately on larger horizons than smaller ones, a phenomenon also found in previous research [50]. For instance, our model has lower prediction losses on $H = 12$ than $H = 6$ for the traffic dataset. This behavior is rooted in the randomness of our data selection process, resulting in more challenging values to predict for $H = 6$ in the testing set. We further select the runner-up models MTGNN and RGSL and conduct extensive experiments to confirm that PHRNN still outperforms them, as shown in Sec. A.

5.4 Hypergraph Sensitivity Analysis

We investigate how the prior knowledge graph impacts PHRNN with respects to two hyperparameters, hyperedge size K and regularization coefficient λ_{reg} . Hyperedge size K determines the density

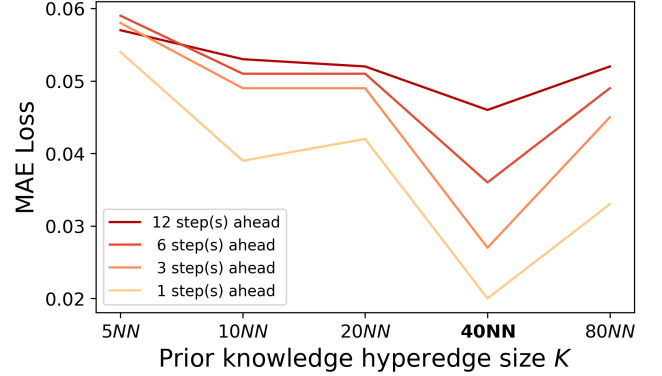


Figure 5: Effectiveness of PHRNN on K . MAE losses of 1, 3, 6, and 12 steps ahead prediction are reported on different hyperedge size $K \in [5, 10, 20, 40, 80]$. A large K means that more nodes are included in each hyperedge of the prior knowledge hypergraph. Our PHRNN reaches the best performance when $K = 40$ (highlighted) for the Adobe dataset.

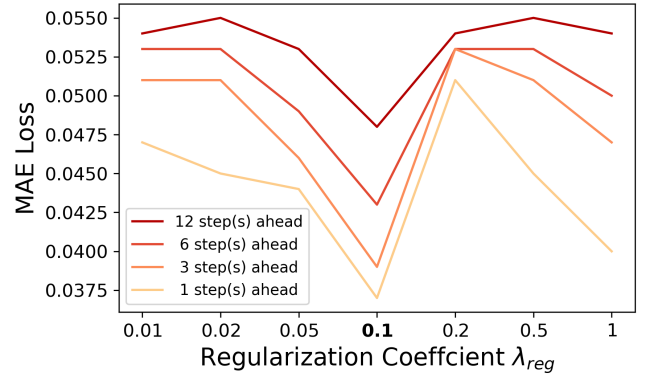


Figure 6: Effectiveness of PHRNN on λ_{reg} . MAE losses of 1, 3, 6, and 12 steps ahead prediction are reported on different regularization coefficients $\lambda_{reg} \in [0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1]$. A larger λ_{reg} indicates the model weights more on complying with the prior knowledge hypergraph. Our PHRNN reaches the best performance when $\lambda_{reg} = 0.1$ (highlighted) for the Adobe dataset.

of the resulting incidence matrix and consequently affects the characteristics of the hypergraph, while regularization coefficient λ determines how closely the learned hypergraph should adhere to the prior knowledge hypergraph.

Variation of Hyperedge Size K . To investigate the effectiveness of hyperedge size K in the prior knowledge graph, we vary $K \in [5, 10, 20, 40, 80]$ and run our PHRNN using the Adobe dataset. For each selected K , we report the MAE losses of 1, 3, 6, and 12 steps ahead predictions. As shown in Fig. 5, our PHRNN exhibits improved performance as K increases from 5 to 40. This improvement may be attributed to the inclusion of more nodes within each

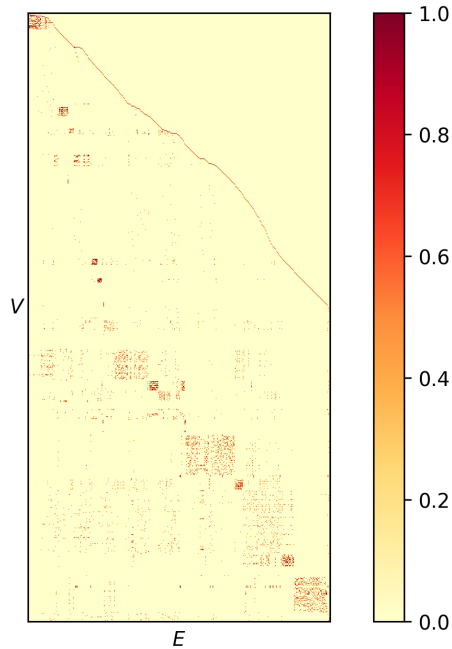


Figure 7: The incidence matrix of one (10NN) prior knowledge hypergraph for the Adobe dataset. Each column indicates a hyperedge.

hyperedge in the prior knowledge hypergraph, leading to the incorporation of extra information. However, this information gain is not unlimited, as the performance deteriorate when K continues to increase to 80.

Variation of Regularization Coefficient λ_{reg} . To investigate the effectiveness of regularization coefficient λ_{reg} , we vary $\lambda_{reg} \in [0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1]$ and run PHRNN on the Adobe dataset. The MAE losses for predictions at 1, 3, 6, and 12 steps ahead are reported for each selected λ_{reg} . As illustrated in Fig. 6, we observe that PHRNN benefits from assigning more weights to the prior knowledge hypergraph as λ_{reg} increases from 0.01 to 0.1, and PHRNN achieves the best performance when $\lambda_{reg} = 0.1$. However, the performance deteriorates when λ_{reg} continues to increase, as a larger regularization coefficient restricts PHRNN from effectively learning the hypergraph structure. More details are included in Sec. A.

6 Case Study: Hypergraph Comparison

We conduct a case study to compare the prior knowledge hypergraph with the learned hypergraph from PHRNN, aiming to understand what node connections PHRNN has learned and how these connections contributes to forecasting accuracy. Fig. 7 illustrates the incidence matrix of a prior knowledge hypergraph for the Adobe dataset, where each row indicate a node and each column indicates a hyperedge. We have reordered the hyperedges by the indices of the nodes they are incident to. Evidently, some hyperedges are incident to similar sets of nodes, as visually observed by the ‘submatrices’ or ‘blocks’ in Fig. 7. These hyperedges convey nearly identical connections, which may lead to redundant information

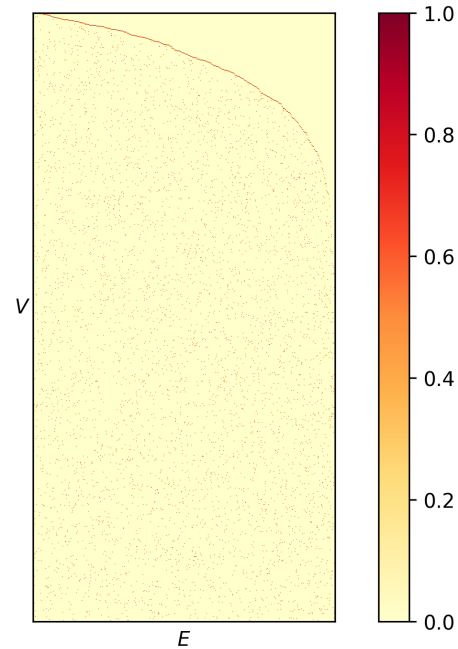


Figure 8: The incidence matrix of one sampled hypergraph for the Adobe dataset. Each column indicates a hyperedge.

in the hypergraph. In contrast, we illustrate a sampling from a learned probabilistic hypergraph of PHRNN, as shown in Fig. 8. To facilitate comparison, we choose to exhibit the top 10 nodes with the highest weights to each hyperedge. We observe that the values of elements in the sampling hypergraph are more evenly distributed across hyperedges, indicating that PHRNN captures and incorporates underlying connections that may be missing from the prior knowledge hypergraphs.

7 Conclusion

In this paper, we introduce a novel model for time-series forecasting, namely, Probabilistic Hypergraph Recurrent Neural Networks (PHRNN). Our proposed PHRNN captures the nature of node interactions from two distinct perspectives: (1) PHRNN learns a hypergraph which allows nodes to engage with hyperedges, simulating broadcasting interactions in groups or communities, and (2) PHRNN models node-hyperedge interactions in a probabilistic manner, providing a more precise representation of the underlying dynamics between individual node time-series and their associated hyperedge. In addition, our extensive experiments on multiple datasets show that PHRNN consistently outperforms state-of-the-art baselines. In the appendix, we include detailed experimental results to show the effectiveness of PHRNN from various perspectives, as described in Sec. A. We also analyze the time complexity of PHRNN, as well as possible ways to boost its computation in Sec. B. In the future, we plan to further extend our investigation of hypergraph models and study how they perform on previous graph models, with an aim of better capturing the interaction manners between entities.

References

- [1] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Bernie Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, et al. 2020. Neural forecasting: Introduction and literature overview. *arXiv preprint arXiv:2004.10240* (2020).
- [2] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Conguri Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. 2021. Spectral temporal graph neural network for multivariate time-series forecasting. *arXiv preprint arXiv:2103.07719* 33 (2021), 17766–17778.
- [3] Hongjie Chen and Hoda Eldardiry. 2023. Graph Time-series Modeling in Deep Learning: A Survey. *ACM Transactions on Knowledge Discovery from Data* (2023).
- [4] Hongjie Chen, Ryan A Rossi, Kanak Mahadik, and Hoda Eldardiry. 2021. Context Integrated Relational Spatio-Temporal Resource Forecasting. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 1359–1368.
- [5] Hongjie Chen, Ryan A Rossi, Kanak Mahadik, Sungchul Kim, and Hoda Eldardiry. 2023. Graph Deep Factors for Probabilistic Time-series Forecasting. *ACM Transactions on Knowledge Discovery from Data* 17, 2 (2023), 1–30.
- [6] Wenchao Chen, Long Tian, Bo Chen, Liang Dai, Zhibin Duan, and Mingyuan Zhou. 2022. Deep variational graph convolutional recurrent network for multivariate time series anomaly detection. In *International Conference on Machine Learning*. PMLR, 3621–3633.
- [7] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3558–3565.
- [8] Everette S. Gardner. 1985. Exponential smoothing: The state of the art. *Journal of Forecasting* 4 (1985), 1–28.
- [9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).
- [10] Yixuan He, Quan Gan, David Wipf, Gesine D Reinert, Junchi Yan, and Mihai Cucuringu. 2022. GNNRank: Learning global rankings from pairwise comparisons via directed graph neural networks. In *International Conference on Machine Learning*. PMLR, 8581–8612.
- [11] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).
- [12] Sheo Yon Jhin, Jaehoon Lee, Minju Jo, Seungji Kook, Jinsung Jeon, Jihyeon Hyeong, Jayoung Kim, and Noseong Park. 2022. Exit: Extrapolation and interpolation-based neural controlled differential equations for time-series classification and forecasting. In *Proceedings of the ACM Web Conference 2022*. 3102–3112.
- [13] Renhe Jiang, Zhaonan Wang, Jiawei Yong, Puneet Jeph, Quanjun Chen, Yasumasa Kobayashi, Xuan Song, Shintaro Fukushima, and Toyotaro Suzumura. 2023. Spatio-temporal meta-graph learning for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 8078–8086.
- [14] Weiwei Jiang and Jiayun Luo. 2022. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications* 207 (2022), 117921.
- [15] Ming Jin, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I Webb, Irwin King, and Shirui Pan. 2023. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *arXiv preprint arXiv:2307.03759* (2023).
- [16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings*. OpenReview.net. <https://openreview.net/forum?id=SJU4ayYgl>
- [17] 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*. 95–104.
- [18] Chonho Lee, Ping Wang, and Dusit Niyato. 2013. A real-time group auction system for efficient allocation of cloud internet applications. *IEEE Transactions on Services Computing* 8, 2 (2013), 251–268.
- [19] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=SjHXGWAZ>
- [20] Zhenyu Liu, Baotian Hu, Zhenran Xu, and Min Zhang. 2023. PPAT: Progressive Graph Pairwise Attention Network for Event Causality Identification. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, Edith Elkind (Ed.). International Joint Conferences on Artificial Intelligence Organization, 5150–5158. <https://doi.org/10.24963/ijcai.2023/572> Main Track.
- [21] Xiaoyi Luo, Jiaheng Peng, and Jun Liang. 2022. Directed hypergraph attention network for traffic forecasting. *IET Intelligent Transport Systems* 16, 1 (2022), 85–98.
- [22] Spyros Makridakis and Michele Hibon. 1997. ARMA Models and the Box–Jenkins Methodology. *Journal of Forecasting* 16, 3 (1997).
- [23] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.
- [24] Domenico Mandaglio, Andrea Tagarelli, and Francesco Gullo. 2020. In and out: optimizing overall interaction in probabilistic graphs under clustering constraints. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1371–1381.
- [25] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=r1ecqn4YwB>
- [26] Md. Rasheduzzaman, Md. Amirul Islam, and Rashedur M. Rahman. 2014. Workload Prediction on Google Cluster Trace. *Int. J. Grid High Perform. Comput.* 6, 3 (July 2014), 34–52.
- [27] Charles Reiss, John Wilkes, and Joseph L Hellerstein. 2011. Google cluster-usage traces: format+ schema. *Google Inc., White Paper* (2011), 1–14.
- [28] Chao Shang, Jie Chen, and Jinbo Bi. 2020. Discrete Graph Structure Learning for Forecasting Multiple Time Series. In *International Conference on Learning Representations*.
- [29] Alasdair Tran, Alexander Mathews, Cheng Soon Ong, and Lexing Xie. 2021. Radflow: A recurrent, aggregated, and decomposable model for networks of time series. In *Proceedings of the Web Conference 2021*. 730–742.
- [30] Vladimir Vapnik, Steven Golowich, and Alex Smola. 1996. Support vector method for function approximation, regression estimation and signal processing. *Advances in neural information processing systems* 9 (1996).
- [31] Cort J Willmott and Kenji Matsuura. 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research* 30, 1 (2005), 79–82.
- [32] Hanrui Wu and Michael K Ng. 2022. Hypergraph convolution on nodes-hyperedges network for semi-supervised node classification. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16, 4 (2022), 1–19.
- [33] Jiming Wu, Qi Qi, Jingyu Wang, Haifeng Sun, Zhikang Wu, Zirui Zhuang, and Jianxin Liao. 2023. Not Only Pairwise Relationships: Fine-Grained Relational Modeling for Multivariate Time Series Forecasting. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th–25th August 2023, Macao, SAR, China*. ijcai.org, 4416–4423. <https://doi.org/10.24963/ijcai.2023/491>
- [34] 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*. 753–763.
- [35] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for deep spatial-temporal graph modeling. In *International Joint Conference on Artificial Intelligence 2019*. Association for the Advancement of Artificial Intelligence (AAAI), 1907–1913.
- [36] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. Association for the Advancement of Artificial Intelligence (AAAI), International Joint Conferences on Artificial Intelligence Organization, 1907–1913.
- [37] Sheng Xiang, Dawei Cheng, Chencheng Shang, Ying Zhang, and Yuqi Liang. 2022. Temporal and Heterogeneous Graph Neural Network for Financial Time Series Prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3584–3593.
- [38] Junchen Ye, Zihan Liu, Bowen Du, Leilei Sun, Weimiao Li, Yanjie Fu, and Hui Xiong. 2022. Learning the evolutionary and multi-scale graph structure for multivariate time series forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2296–2306.
- [39] Jaehyuk Yi and Jinkyoo Park. 2020. Hypergraph convolutional recurrent neural network. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3366–3376.
- [40] Nan Yin, Fuli Feng, Zhigang Luo, Xiang Zhang, Wenjie Wang, Xiao Luo, Chong Chen, and Xian-Sheng Hua. 2022. Dynamic hypergraph convolutional network. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 1621–1634.
- [41] Nan Yin, Fuli Feng, Zhigang Luo, Xiang Zhang, Wenjie Wang, Xiao Luo, Chong Chen, and Xian-Sheng Hua. 2022. Dynamic Hypergraph Convolutional Network. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 1621–1634. <https://doi.org/10.1109/ICDE53745.2022.00167>
- [42] Nan Yin, Li Shen, Huan Xiong, Bin Gu, Chong Chen, Xian-Sheng Hua, Siwei Liu, and Xiao Luo. 2023. Messages are Never Propagated Alone: Collaborative Hypergraph Neural Network for Time-Series Forecasting. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 01 (2023), 1–15.
- [43] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3634–3640.
- [44] Hongyuan Yu, Ting Li, Weichen Yu, Jianguo Li, Yan Huang, Liang Wang, and Alex Liu. 2022. Regularized Graph Structure Learning with Semantic Knowledge for Multi-variables Time-Series Forecasting. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. Lud De Raedt

- (Ed.). International Joint Conferences on Artificial Intelligence Organization, 2362–2368. <https://doi.org/10.24963/ijcai.2022/328> Main Track.
- [45] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit Yan Yeung. 2018. GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*.
- [46] Yunhao Zhang and Junchi Yan. 2022. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*.
- [47] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. 2019. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems* 21, 9 (2019), 3848–3858.
- [48] Yusheng Zhao, Xiao Luo, Wei Ju, Chong Chen, Xian-Sheng Hua, and Ming Zhang. 2023. Dynamic Hypergraph Structure Learning for Traffic Flow Forecasting. ICDE.
- [49] Zhenzhen Zhao, Guojiang Shen, Junjie Zhou, Junchen Jin, and Xiangjie Kong. 2023. Spatial-temporal hypergraph convolutional network for traffic forecasting. *PeerJ Computer Science* 9 (2023), e1450.
- [50] 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 *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 11106–11115.

Table 6: PHRNN MAE Results of Mean and Sum Aggregation

Data	v to e	Horizon			
		1	3	6	12
ADOBE	Mean	0.039 ± 0.021	0.049 ± 0.023	0.051 ± 0.023	0.053 ± 0.022
	Sum	0.049 ± 0.026	0.050 ± 0.024	0.052 ± 0.023	0.052 ± 0.022
GOOGLE	Mean	0.060 ± 0.022	0.061 ± 0.014	0.062 ± 0.010	0.067 ± 0.016
	Sum	0.061 ± 0.019	0.062 ± 0.011	0.062 ± 0.009	0.066 ± 0.016
TRAFFIC	Mean	0.013 ± 0.011	0.029 ± 0.009	0.034 ± 0.008	0.033 ± 0.012
	Sum	0.033 ± 0.014	0.037 ± 0.012	0.038 ± 0.013	0.034 ± 0.013

Table 7: PHRNN MAE Results of Various K on Adobe

K	Horizon			
	1	3	6	12
5	0.054 ± 0.020	0.058 ± 0.021	0.059 ± 0.021	0.057 ± 0.021
10	0.039 ± 0.021	0.049 ± 0.023	0.051 ± 0.023	0.053 ± 0.022
20	0.042 ± 0.023	0.049 ± 0.027	0.051 ± 0.026	0.052 ± 0.024
40	0.020 ± 0.007	0.027 ± 0.010	0.036 ± 0.013	0.046 ± 0.015
80	0.033 ± 0.017	0.045 ± 0.028	0.049 ± 0.027	0.052 ± 0.024

Table 8: PHRNN MAE Results of Various λ_{reg} on Adobe

λ_{reg}	Horizon			
	1	3	6	12
0.01	0.047 ± 0.030	0.051 ± 0.026	0.053 ± 0.025	0.054 ± 0.023
0.02	0.045 ± 0.030	0.051 ± 0.025	0.053 ± 0.024	0.055 ± 0.021
0.05	0.044 ± 0.029	0.046 ± 0.024	0.049 ± 0.022	0.053 ± 0.020
0.1	0.037 ± 0.020	0.039 ± 0.018	0.043 ± 0.017	0.048 ± 0.017
0.2	0.051 ± 0.025	0.053 ± 0.022	0.053 ± 0.022	0.054 ± 0.021
0.5	0.045 ± 0.029	0.051 ± 0.026	0.053 ± 0.024	0.055 ± 0.022
1	0.040 ± 0.026	0.047 ± 0.025	0.050 ± 0.024	0.054 ± 0.022

Table 5: Ablation Study with Constant θ

Metrics	DATA	H	PHRNN with constant θ	PHRNN
MAE	ADOBE	1	0.044 ± 0.021	0.039 ± 0.021
		3	0.055 ± 0.029	0.049 ± 0.023
		6	0.056 ± 0.028	0.051 ± 0.023
		12	0.059 ± 0.028	0.053 ± 0.022
	GOOGLE	1	0.065 ± 0.013	0.060 ± 0.022
		3	0.063 ± 0.007	0.061 ± 0.014
		6	0.064 ± 0.009	0.062 ± 0.010
		12	0.068 ± 0.014	0.067 ± 0.016
	TRAFFIC	1	0.029 ± 0.016	0.013 ± 0.011
		3	0.035 ± 0.013	0.029 ± 0.009
		6	0.036 ± 0.011	0.034 ± 0.008
		12	0.035 ± 0.012	0.033 ± 0.012
RMSE	ADOBE	1	0.087 ± 0.037	0.075 ± 0.034
		3	0.099 ± 0.043	0.092 ± 0.038
		6	0.102 ± 0.039	0.097 ± 0.038
		12	0.105 ± 0.039	0.100 ± 0.036
	GOOGLE	1	0.083 ± 0.015	0.077 ± 0.025
		3	0.080 ± 0.008	0.078 ± 0.015
		6	0.082 ± 0.010	0.079 ± 0.011
		12	0.086 ± 0.016	0.085 ± 0.019
	TRAFFIC	1	0.043 ± 0.024	0.019 ± 0.020
		3	0.050 ± 0.021	0.045 ± 0.018
		6	0.052 ± 0.020	0.050 ± 0.015
		12	0.049 ± 0.017	0.047 ± 0.017

A Complete Experimental Results

In this section, we add extra results from a larger traffic dataset. We also show detailed experimental results of hypergraph sensitivity analysis described in Sec. 5.4. Additionally, we include ablation study results on different ways of hyperedge aggregation.

Effectiveness of Hypergraph. We further conduct ablation studies to assess the effectiveness of θ . Specifically, we drop the node and hyperedge embeddings, and set $\theta = \frac{1}{NM}$ as a constant value for each node-hyperedge pair. As observed in Table 5, PHRNN achieves better results when learning the probabilistic hypergraph structure, showing the effectiveness of the hypergraph.

Ablation Study on Hyperedge Aggregation. We further investigate the impact of different hyperedge aggregation methods. We obtain results with sum aggregation in addition to the original results that are based on mean aggregation. According to Table 6, both aggregation methods have similar performance.

Detailed Results on Hypergraph Sensitivity. Table 7 and Table 8 include detailed results on various K and λ_{reg} in the form of mean±std. These tables serve as supplements to Fig. 5 and Fig. 6. We note that PHRNN not only has the best mean value when $K = 40$ and $\lambda_{reg} = 0.1$, but it also achieves the best standard deviation.

Extra Experiments with a Larger Dataset. To enhance our experiments, we also add results from a larger dataset of Tokyo expressways containing 1421 nodes [13], as shown in Table 9.

B Time Complexity Analysis

The time complexity of our proposed PHRNN comprises two primary components: (i) prior knowledge KNN hypergraph construction, which can be computed in $O(N \log N)$ with the aid of HNSW [23], where N is the number of nodes; and (ii) the computation of θ for each element in the incidence matrix $C \in \mathbb{R}^{N \times M}$, where M is the number of hyperedges. This part incurs a time complexity of $O(NM)$. Overall, our PHRNN has a time complexity of $O(N \log N + NM)$. Notably, the order of M does not scale with the order of N ; instead, it relies on the choice of the prior knowledge hypergraph by the model users. Researchers can opt for a small M when using a more advanced clustering algorithm, thereby reducing it to a constant factor in terms of time complexity.

Table 9: Results for Single-step ahead and Multi-step ahead Forecasting (MAE and RMSE) for the Tokyo Traffic Dataset

Metrics	H	HGC-RNN	DGSL	RGSL	MTGNN	STGCN	StemGNN	MegaCRN	ESG	PHRNN (Ours)
MAE	1	13.142 ± 2.220	11.481 ± 1.473	13.455 ± 0.256	8.674 ± 0.433	10.268 ± 0.997	14.495 ± 3.456	9.627 ± 2.410	11.344 ± 1.794	8.544 ± 0.487
	3	13.269 ± 2.677	11.468 ± 1.348	13.615 ± 0.389	9.512 ± 0.330	11.087 ± 1.661	14.411 ± 0.889	10.338 ± 2.592	13.984 ± 3.358	9.446 ± 1.649
	6	13.947 ± 2.908	11.804 ± 1.571	13.710 ± 1.269	9.485 ± 1.384	11.300 ± 1.905	39.457 ± 10.042	10.745 ± 3.256	13.270 ± 0.783	12.439 ± 3.617
	12	15.262 ± 4.406	13.251 ± 2.834	14.635 ± 2.437	10.556 ± 2.175	12.223 ± 3.409	16.883 ± 2.167	12.080 ± 4.471	14.963 ± 3.028	14.464 ± 5.406
RMSE	1	17.538 ± 3.047	15.637 ± 2.571	18.618 ± 1.129	12.616 ± 1.680	14.457 ± 1.853	19.955 ± 5.576	13.772 ± 4.115	15.693 ± 1.765	11.711 ± 0.899
	3	17.809 ± 3.853	16.004 ± 2.743	19.165 ± 1.618	13.470 ± 2.210	15.552 ± 2.687	20.153 ± 2.084	14.568 ± 4.322	18.361 ± 4.409	13.191 ± 1.991
	6	18.870 ± 4.409	16.680 ± 3.491	19.104 ± 2.883	14.002 ± 3.432	16.109 ± 3.835	47.543 ± 10.988	15.312 ± 5.419	18.513 ± 1.903	16.871 ± 5.277
	12	20.591 ± 6.601	18.623 ± 5.542	20.466 ± 4.785	15.562 ± 4.963	17.549 ± 6.171	23.076 ± 3.665	17.077 ± 7.270	20.944 ± 4.792	19.464 ± 7.828