Knowledge Discovery in Database

Hypergraph Convolutional Recurrent Neural Network

Jaehyuk Yi
KAIST
韩国科学技术院
Daejeon, South Korea
randrandrre@kaist.ac.kr

Jinkyoo Park*
KAIST
Daejeon, South Korea
jinkyoo.park@kaist.ac.kr

ABSTRACT

In this study, we present a hypergraph convolutional recurrent neural network (HGC-RNN), which is a prediction model for structured time-series sensor network data. Representing sensor networks in a graph structure is useful for expressing structural relationships among sensors. Conventional graph structure, however, has a limitation on representing complex structure in real world application, such as shared connections among multiple nodes. We use a hypergraph, which is capable of modeling complicated structures, for structural representation. HGC-RNN performs a hypergraph convolution operation on the input data represented in the hypergraph to extract hidden representations of the input, while considering the structural dependency of the data. HGC-RNN employs a recurrent neural network structure to learn temporal dependency from the data sequence. We conduct experiments to forecast taxi demand in NYC, traffic flow in the overhead hoist transfer system, and gas pressure in a gas regulator. We compare the performance of our method with those of other existing methods, and the result shows that HGC-RNN has strengths over baseline models.

空中走行搬运

CCS CONCEPTS

• **Computing methodologies** → **Temporal reasoning**; **Spatial and physical reasoning**; • **Information systems** → *Sensor networks.*

KEYWORDS

hypergraph, representation learning, graph neural networks, time-series prediction

ACM Reference Format:
Jaehyuk Yi and Jinkyoo Park. 2020. Hypergraph Convolutional Recurrent Neural Network. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403389

1 INTRODUCTION

With the advancements in sensor and network technologies, increasing numbers of sensors are being installed in various systems to acquire continuous data for various purposes. As sequential data acquired from these sensors record synchronous time-series from multiple locations, more sophisticated and precise analysis is available compared to data from a single sensor. Since a large amount of data does not necessarily yield better results, proper modeling techniques are required to extract meaningful information from the data. Particularly, when modeling general systems in which underlying rules are embedded for the relationships between each element, such knowledge can be incorporated in the construction of data-driven models for accurate description of the behaviors of the systems.

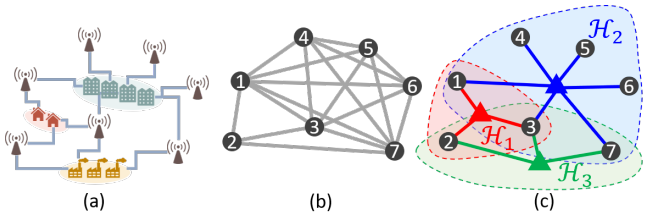


Figure 1: Comparison between conventional graph and hypergraph representation of complicated network. (a) Physical system with interconnected nodes. (b) Graph representation of the network. (c) Hypergraph representation of the network.

Graphs have been used as effective ways to represent structured data, capturing relationships among data entities. A graph neural network (GNN) is a type of neural network that receives data in graph representation as input, and effectively processes graph-structured features to conduct various end-tasks such as classification [3, 7, 10, 17], prediction [6, 27] and control [11, 18] tasks. Recent studies have integrated GNNs with various recurrent neural network models to capture the spatio-temporal evolution of data over networks [4, 8, 14, 19, 24].

Although various models have achieved successful results on several tasks using a GNN approach, this approach still has a limitation in expressing complex structures of data. Conventional graphs map individual data points to nodes, and connections or correlations between two points to edges. However, such pairwise connections are not always appropriate in actual application. In practical cases, there often exist more complicated relationships among multiple nodes than pairwise connections between two nodes. Figure 1 shows an example of complicated relationships of a sensor network that are difficult to represent with a conventional graph.

A hypergraph is a generalized concept that can express complex networks. In a conventional graph, the number of vertices on an edge is strictly defined as two. On the other hand, in a hypergraph, a flexible number of vertices are connected with hyperedges. Therefore, more complicated and higher level relationships among

*Jinkyoo Park is the corresponding author.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403389

nodes can be represented with hypergraphs, as shown in Figure 1(c). Some recent studies have introduced hypergraphs to deep learning models to resolve tasks in complex networks [2, 5, 15, 28].

In this study, we propose a hypergraph convolutional recurrent neural network (HGC-RNN), an integration of the hypergraph convolution to RNN, to model correlated multivariate sequential data from real-world sensor networks. The hypergraph convolution extracts the structural feature of the data, and the recurrent structure of the model extracts temporal dependencies between sequential time steps. Using the hypergraph concept, our model can overcome the limitations of conventional graph time-series models that cannot express complicated structures. Furthermore, with a flexible hyperedge setup, we can design the system to infer various external variables or missing data in stochastic conditions. In experiments, depending on the applications, we can attach several modules such as temporal embedding layers, coordinate embedding layers, or environmental feature embedding layers to improve models.

We evaluate the proposed method for taxi demand prediction, overhead hoist transfer (OHT) system traffic flow prediction, and city gas pressure regulator data forecasting tasks. For the taxi demand dataset, we employ the proposed model on the benchmark dataset, and compare the performance of our model to those of existing task-specified models. We show that our model has an accuracy comparable to those of the best task-specified baselines, though our model uses 1000-times fewer parameters. For the OHT system dataset, we employ the proposed model to predict the traffic flow of autonomous vehicles. Given networked data with missing sensor nodes, the proposed model uses as input a hypergraph constructed to define the relationships among observed sensor nodes and missing sensor nodes and predicts future traffic flow. The experimental results imply that HGC-RNN is capable of using hyperedges to infer the status of missing nodes, like an autoencoder. For the city gas pressure regulator data set, we employ the proposed model to simultaneously predict gas pressures of 195 gas regulators in actual operation in a city. The results show that our hypergraph-based model performs better than the conventional graph-based model.

In summary, our main contributions are as follows:

- We introduce the hypergraph concept for sequential data analysis. Compared to existing models using GNN for structured time-series data, HGC-RNN can perform more intuitive and sophisticated modeling of complex networks.
- With flexible hyperedge settings, the proposed model is shown to effectively infer information not given in the data, such as environmental states or unobservable nodes.
- We employ the proposed method to analyze real world application datasets and validate the effectiveness of the model

2 RELATED WORK

Multivariate Time-series Analysis. A number of recent studies have used deep learning to analyze multivariate time-series data. Neural network frameworks integrating convolutional and recurrent networks have been popular approaches to analyze multivariate time-series data. DeepSense [23] is a deep learning framework proposed for multivariate time-series sensor networks. It applies CNN on inputs to obtain unified features of data from multiple sensors; the obtained features are fed into RNN to capture temporal dependencies. MSCRED [25] detects anomalies in multivariate time series data, reconstructing the data with an encoder-decoder structured convolutional recurrent network model; it then uses the reconstruction error as the anomaly score. DMVST-Net [22] and STDN [21] predict taxi demand in certain regions of a city, applying CNN and RNN sequentially to capture spatial and temporal trends, respectively. DMVST-Net simultaneously leverages spatial, temporal, and semantic features. STDN, from a follow-up study of DMVST-Net, improved overall performance by attaching an attention module to identify long-term temporal dependencies.

Some of these studies obtained locality information through CNN [13, 21, 22], while others used CNN but did not consider actual locality [23, 25]. However, none of the methods were capable of considering non-adjacent structural connections.

Graph Time-series Analysis. Graph neural networks (GNN) have been used to represent structured data while conducting various tasks. Recently, GNNs have been employed with various recurrent neural network models to capture the temporal evolution of structured data. GNN models have been applied to the analysis of multivariate time-series data obtained from dynamic network systems.

Xu and Li[19] proposed a model using GNN and RNN to predict taxi demand. They define a graph with an Origin-Destination relation of grids, and perform graph convolution to capture the influence of non-adjacent areas. To forecast traffic flow, a diffusion convolutional recurrent neural network (DCRNN)[8] has been proposed that uses a directed graph to represent the status of the traffic network. DCRNN employs random walks on the graph to obtain spatial dependencies and an encoder-decoder with RNN structure to obtain temporal dependencies. Hsu [4] proposed an anomaly detection method that uses features extracted from a multivariate time series to identify faults of target system. System extracts time-series features using a graph laplacian transform under a framework based on VAE and RNN. ST-GCN [24] uses temporal gated convolution blocks and spatial graph convolution blocks to comprehensively obtain spatio-temporal features to predict the graph time-series. Graph-VRNN [14] combines GNN and VAE into the recurrent network and can predict graph time-series representing trajectories of multiple agents, such as sports players.

These models have achieved success in many applications; however, when utilizing graph convolution to capture non-adjacent structural connections, there still exists a limitation in modeling complicated networked systems with beyond pairwise connections (Figure 1(b)).

Hypergraph Learning. Because hyperedges can connect different numbers of nodes instead of connecting only two nodes, hypergraphs have been used as generalized representations of conventional graphs. Due to this flexibility of hypergraphs, several studies have used them as representation schemes to model more complex systems with higher order interaction.

[29] was the first to introduce hypergraphs to represent complicated relationships. It showed that hypergraph-based learning outperformed graph-based learning on several clustering, embedding, and classification tasks. In addition, hypergraph neural networks (HGNN)[2] introduce hypergraph-based representation learning

to a deep learning model, which has been shown to outperformed conventional graph neural networks on classification and object recognition tasks. HyperGCN [20], whose design was motivated by graph convolutional networks, suggested a more efficient and faster model for the same tasks to HGNN. Dynamic hypergraph neural networks [5] utilize k-NN and k-means clustering methods to build a hypergraph structure of data without an explicitly defined hypergraph structure and perform tasks similar to those performed by HGNN and HyperGCN. Deep hyper-network embedding (DHNE)[15] and Hyper-SAGNN [28] encode networks with hypergraphs to represent complicated structured data.

The majority of studies on hypergraphs have exclusively focused on conducting static tasks such as classification and clustering, utilizing the representation power of hypergraph-based embedding. In this study, we first extend the hypergraph-based convolution model to a dynamic setting to model the spatio-temporal evolution of structured sequential data.

3 PROBLEM DEFINITION

In many real-world applications, relationships between different nodes are not always one-to-one. When creating graphs in a complex network, it is difficult to clearly define edges, because edges can be defined only between exactly two nodes. If more than two nodes are correlated, their connections have to be separated into several different edges. In this section, we formally define a hypergraph that models high-level interactions between nodes and formulate the prediction task for the sequence of hypergraphs.

3.1 Definition of Hypergraph

Hypergraph is defined as: $G = (V, E)$, where $V = \{v_1, ..., v_N\}$ is the set of nodes, and $E = \{\mathcal{H}_1, ..., \mathcal{H}_M\}$ is the set of hyperedges representing connectivity among nodes. Hyperedge $\mathcal{H}_m$ is a subset of V, i.e., group of nodes that belong to hyperedge $\mathcal{H}_m$. In Figure 1(c), for illustration, $\mathcal{H}_1 = \{v_1, v_2, v_3\}, \mathcal{H}_2 = \{v_1, v_4, v_5, v_6, v_7\}$, and $\mathcal{H}_3 = \{v_2, v_3, v_7\}$.

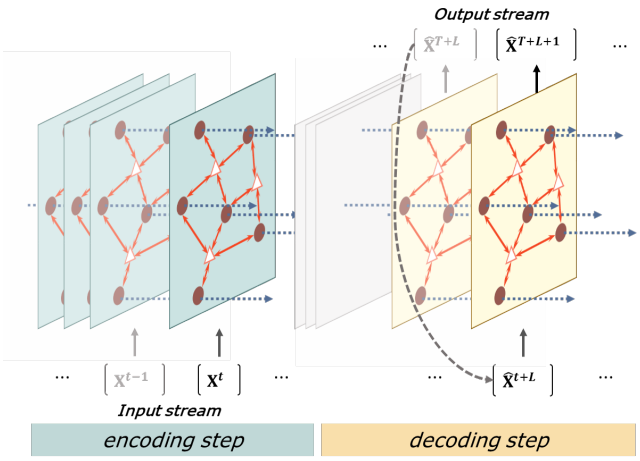


Figure 2: Sequence to sequence hypergraph prediction for spatio-temporal data modeling

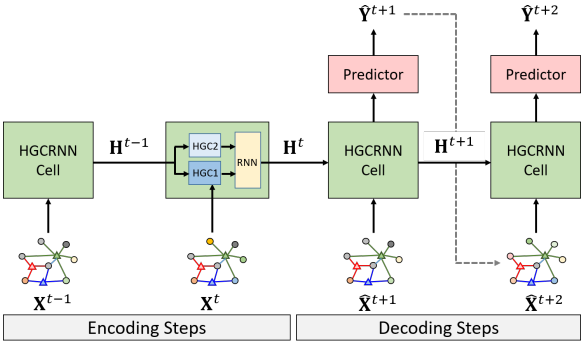


Figure 3: Overall architecture of HGC-RNN

3.2 Problem Definition

Our goal, given a past sequence of observations. is to predict a future sequence of complex structured data. Here we present a general definition of the problem we want to solve in this study, which is illustrated in Figure 2.

In this problem, we denote $\mathbf{x}_n^t$ as the feature vector obtained from node n at time t. Then, we can define $\mathbf{X}^t = (\mathbf{x}_1^t, ..., \mathbf{x}_N^t)$, a stacked node feature vector, as node feature matrix for all nodes at time t. Given the sequence of node feature matrix $\mathbf{X}^{t-R+1:t} = (\mathbf{X}^{t-R+1}, ..., \mathbf{X}^t)$ and the hypergraph G representing the relationships among nodes, our goal is to learn the prediction function f:

$$f(\mathbf{X}^{t-R+1:t}, G) = \hat{\mathbf{X}}^{t+1:t+L}$$ static

which predict the future L sequence of node feature matrices $\hat{\mathbf{X}}^{t+1:t+L}$.

4 METHODOLOGY

This section describes our proposed model HGC-RNN, a neural network framework designed to predict the stream of structured data.

4.1 Overall Structure of HGC-RNN

The overall architecture has a recurrent structure, as shown in Figure 3. Specifically, HGC-RNN is an encoder-decoder structured framework, estimating temporal states from the past graph sequences (encoding step) to predict the future sequence of the graph (decoding step). The HGC-RNN model is composed of two modules: an **HGC-RNN-cell** as the transition module that estimates the hidden states over sequence, and a **Predictor** as the decoder module that maps the hidden states onto the target outputs. All the inputs and outputs are node feature matrices with specified hypergraphs. That is, all modules accept the node feature matrix as an input and update the node feature vectors of all nodes simultaneously.

HGC-RNN-cell. The HGC-RNN-cell models the temporal evolution of the hidden feature matrix $\mathbf{H}^t$ over time. It receives the previous hidden feature matrix $\mathbf{H}^{t-1}$ and the current input node feature matrix $\mathbf{X}^t$ to compute the next hidden feature matrix $\mathbf{H}^t$ as:

$$\mathbf{H}^t = F_{\text{HGC-RNN-cell}}(\mathbf{X}^t, \mathbf{H}^{t-1}) = \Phi^{\text{RNN}}(\chi^t, \eta^t)$$

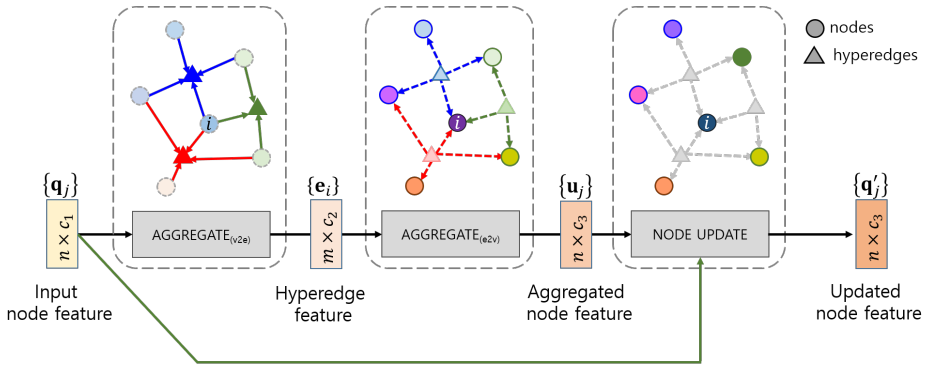


Figure 4: Two-step propagation of hypergraph convolution layer

where Φ^{RNN} is a RNN cell. Also, feature matrix χ^t contains structural and temporal features, while the other feature matrix η^t contains global states, which are computed as:

$$\chi^t = \Phi^{\text{HGC1}}(\mathbf{X}^t || \mathbf{H}^{t-1})$$
$$\eta^t = \Phi^{\text{HGC2}}(\mathbf{H}^{t-1})$$

where || denotes concatenation. In addition, Φ^{HGC1} and Φ^{HGC2}, hypergraph convolution operators, process node feature matrix into an updated hidden node feature matrix while considering the high-order structural relationships among nodes represented in the hypergraph. This operation is described in detail in section 4.2.

Predictor. The predictor maps the hidden vector h^t to the predicted values $\hat{y}_t$ as:

$$\hat{\mathbf{Y}}^t = \Phi^{\text{DEC}}(\mathbf{H}^t)$$

where Φ^{DEC} is a neural network working as a decoder.

During the decoding steps, the model can predict the future sequence of a graph of any arbitrary length by repeatedly using the output $\hat{\mathbf{Y}}^t$ of the previous cell as an input $\hat{\mathbf{X}}^{t+1}$ to the next cell.

4.2 Hypergraph Convolution (HGC) Layer

The most significant element of our model is the hypergraph convolution layer applied in each cell. The role of HGC is to use hypergraph G to update the node-wise feature matrix $\mathbf{Q}$. In this section, we focus on describing how a hidden feature vector $\mathbf{q}_j$ for a single node v_j is updated by a single operation of the HGC layer, given the neighbor node features with their structural relationships represented in the hypergraph G:

$$\mathbf{q}'_j = \Phi^{\text{HGC}}(\mathbf{q}_j)$$

where $\mathbf{q}'_j$ is the updated feature vector. The procedure of node feature update is composed of three steps, as shown in Figure 4; the details of each step are described below.

4.2.1 *Procedure of HGC.*

Node to edge aggregation. The first step of hypergraph convolution is the node-to-edge (*v-to-e*) aggregation to construct the edge feature vector $\mathbf{e}_i$ by aggregating the feature vectors $\{\mathbf{q}_k | v_k \in \mathcal{H}_i\}$ of the nodes that belong to $\mathcal{H}_i$:

$$\mathbf{e}_i = \text{AGGREGATE}_{(\text{v2e})}(\{\mathbf{q}_k | v_k \in \mathcal{H}_i\})$$

where $\mathbf{q}_k$ is the feature vector for node v_k. AGGREGATE denotes the aggregation function, which will be discussed in section 4.2.2. This aggregation is applied concurrently to all hyperedges $\mathcal{H}_i$ in E.

Edge to node aggregation. The second step is edge-to-node (*e-to-v*) aggregation in which each node aggregates the influences from the hyperedges containing the node. For each node v_j, feature vectors of hyperedges containing the node v_j (i.e., $\{\mathbf{e}_k | v_j \in \mathcal{H}_k\}$) are aggregated to compute the updated node feature vector $\mathbf{u}_j$ as:

$$\mathbf{u}_j = \text{AGGREGATE}_{(\text{e2v})}(\{\mathbf{e}_k | v_j \in \mathcal{H}_k\})$$

Single node updating procedures are conducted simultaneously for all the nodes in V.

Node update. The node update function receives the aggregated node-wise feature vector $\mathbf{u}_j$ and the original node-wise feature vector $\mathbf{q}_j$ as inputs to compute the updated node-wise feature $\mathbf{q}'_j$. For the update function, we have applied the linear model to the inputs as:

$$\mathbf{q}'_j = W^{\text{T}}_{(UPDATE)}(\mathbf{q}_j || \mathbf{u}_j) + \mathbf{b}$$

where W and $\mathbf{b}$ indicate learnable weight and bias, respectively.

The series of node updating procedure, which is composed of edge aggregation, node aggregation, and node updating, is similar to how a node feature is updated in a relation network [12]. Relation networks learn relationships between two nodes while HGC learns relationships between flexible numbers of nodes; HGC could be thought of as a generalization of the relation network.

4.2.2 *Variants of Aggregating Methods.*

Mean Aggregation. The first method used in the aggregation function is mean aggregation. The *v-to-e* aggregation function for edges computes an edge feature vector $\mathbf{e}_i$ for $\mathcal{H}_i$ by aggregating the transformed feature vectors of nodes that belong to $\mathcal{H}_i$. Similarly, the *e-to-v* aggregation function for nodes computes an updated node feature vector $\mathbf{u}_j$ for $\mathbf{v}_j$ by aggregating the transformed feature vectors of edges that contain $\mathbf{v}_j$. The updating operation can

be summarized as:

$$\mathbf{e}_i = \frac{1}{N_i} \sum_{k|v_k \in \mathcal{H}_i} W^{\mathrm{T}}_{(\mathrm{v2e})} \cdot \mathbf{q}_k$$

$$\mathbf{u}_j = \frac{1}{M_j} \sum_{k|v_j \in \mathcal{H}_k} W^{\mathrm{T}}_{(\mathrm{e2v})} \cdot \mathbf{e}_k$$

where N_i indicates the number of nodes connected to hyperedge $\mathcal{H}_i$, and M_j indicates the number of hyperedges connected to node v_j.

The strength of this method is its simplicity and lightness. By averaging the features of all connected hyperedges, the model can simply summarize the overall neighborhood features. But, in cases in which every hyperedge has a different importance, mean aggregation cannot consider the relative importance of the influence of neighborhoods.

Attentive Aggregation. The second method is attentive aggregation, which involves adoption of graph attention network(GAT)[17] in the hypergraph convolution. The *v-to-e* aggregation is same as the *v-to-e* aggregation of mean aggregation. The *e-to-v* aggregation function for nodes computes the weighted average of the transformed feature vectors of edges that contain $\mathbf{v}_j$. The weight is computed with an attention coefficient. These two operations are formulated as:

$$\mathbf{e}_i = \frac{1}{N_i} \sum_{k|v_k \in \mathcal{H}_i} W^{\mathrm{T}}_{(\mathrm{v2e})} \cdot \mathbf{q}_k$$

$$\mathbf{u}_j = \frac{1}{M_j} \sum_{k|v_j \in \mathcal{H}_k} \alpha_{kj} * (W^{\mathrm{T}}_{(\mathrm{e2v})} \cdot \mathbf{e}_k)$$

where α is the attention coefficient.

The attention coefficient is calculated with the original node feature and the aggregated hyperedge features, quantifying how important the corresponding hyperedge is to the target node. As in [17], we define the attention coefficient as the softmax of a single-layer feedforward neural network. Because the set of sources and the set of targets are the same in the GAT model, while in our case the set of sources is the hyperedge and the target is the node, we made a small modification to the computation of the attention coefficients. We used $W_{(\mathrm{e2v})}$ to encode the edge features, and defined a new weight matrix $W_{(\mathrm{v})}$ to encode the node features. To summarize, attention coefficient α is defined as follows:

$$\alpha_{kj} = \frac{\exp\left(\mathbf{LeakyReLU}\left(\vec{a}^{\mathrm{T}}[W^{\mathrm{T}}_{(\mathrm{v})}\mathbf{q}_j || W^{\mathrm{T}}_{(\mathrm{e2v})}\mathbf{e}_k]\right)\right)}{\sum_{l|j\in\mathcal{H}_l} \exp\left(\mathbf{LeakyReLU}\left(\vec{a}^{\mathrm{T}}[W^{\mathrm{T}}_{(\mathrm{v})}\mathbf{q}_j || W^{\mathrm{T}}_{(\mathrm{e2v})}\mathbf{e}_l]\right)\right)}$$

Using attentive aggregation, the model can consider different importances of connected hyperedges, without any external definition of edge weights. Attentive aggregation may be preferred in some specific cases in which the importance of hyperedges is a critical factor.

5 EXPERIMENT

In this section, we validate the proposed method using three datasets: (1) the taxi demand in NYC, (2) the inlet and outlet pressures of the gas regulators that are connected by the gas-pipeline network in a city, and (3) the traffic flow of overhead transportation system (OHT) in a semiconductor factory, and. All the experiments report the mean and standard deviation of the results over ten iterations. We note that HGC-RNN-M means HGC-RNN model using mean aggregation on hypergraph convolution, and HGC-RNN-A means HGC-RNN model using attentive aggregation on hypergraph convolution.

5.1 NYC-Taxi Demand

Objectives. Taxi demand varies depending on regions and times in a city. Accurately predicting taxi demand would be helpful in optimally dispatching, routing, and re-balancing taxis for maximizing the profits and the quality of service. The experiments are designed to show how the proposed model can effectively use a hypergraph to model the intricate taxi demand patterns with a more compact model with a significantly smaller number of model parameters than other state-of-art models.

Dataset Description. The NYC-Taxi dataset contains pick-up and drop-off records of taxis in NYC. Data are given in 10×20 grids, where each region covers a $1km \times 1km$ area. The dataset was collected from 01/01/2015 to 03/01/2015 during a total of two months, and the temporal interval between each time step is set as 30 minutes. In our experiments, we use the first 40 days of data as the training data and the remaining 20 days as the test data. The below explains in detail about the input and output of the HGC-RNN model at each time step t:

- $\mathbf{X}^t = \{\mathbf{x}^t_1, ..., \mathbf{x}^t_{200}\} \in \mathbb{R}^{200\times 1}$: The taxi demand data over all the regions where $\mathbf{x}^t_n \in \mathbb{R}$ is the taxi demand of node n at time t.
- $\mathbf{Y}^{t+1} \in \mathbb{R}^{200\times 1}$: The predicted taxi demand at time step $t+1$ (i.e., forecasting of taxi demand after 30 minutes).
- $G = (V, E)$: Hypergraph representing the city regions (nodes) and their relationships (hyperedges). Each node represents a single region of the city, and a hyperedge is constructed to group regions with a similar pattern of taxi demand. The number of nodes is $|\mathbf{V}| = 10 \times 20$, and the number of hyperedges is $|\mathbf{E}| = 42$. The detailed definition of the hypergraph is described in Appendix.

Model Specification and Baselines. In the sequence to sequence prediction framework, the model uses $R = 8$ past sequence of hypergraphs as an input and predicts the next $L = 1$ step ahead taxi demand over all the regions.

Along with node feature matrix $\mathbf{X}^t$ (demand records over all the regions), we used the following external features:

- $\mathbf{t}^t \in \mathbb{R}^{57}$: The concatenation of 4 one-hot vectors representing time-of-the-day (48), day-of-the-week (7), whether it is a holiday (1), whether it is the day before a holiday (1), respectively. The timestamp is encoded with 2-layer MLP, and the encoded hidden vector is used along with $\mathbf{X}^t$ as an input to the HGC-RNN cell.
- $\mathbf{C}^t \in \mathbb{R}^{200\times 2}$: The coordinates of the regions. The coordinates are encoded with CNN, and the encoded hidden vector is used with $\mathbf{X}^t$ as an input to the HGC-RNN cell.
- $\mathbf{D}^t \in \mathbb{R}^{200\times 15}$: The past 15-time step taxi drop off records of all the regions. Drop-off records are encoded with a separate 2-stacked hypergraph convolution (HGC) layers, and the

Table 1: Peformance comparison on predicting NYC-taxi demand

	RMSE	MAPE$_{(\times 10^2)}$	# of params
ARIMA	36.53	22.21	
XGBoost	26.07	19.35	
STResNet	26.23 ± 0.33	21.13 ± 0.63	4,884,353
DMVST-Net	25.74 ± 0.26	17.38 ± 0.46	1,499,021
STDN	**24.10 ± 0.25**	**16.30 ± 0.23**	9,446,274
GC-RNN	26.68 ± 0.03	17.06 ± 0.02	**6,071**
RN-RNN	26.99 ± 0.03	17.32 ± 0.03	**8,119**
HGC-RNN-M	25.45 ± 0.06	**16.30 ± 0.02**	**8,119**
HGC-RNN-A	25.97 ± 0.07	17.77 ± 0.02	**10,497**

encoded hidden node feature matrix along with the hidden node feature matrix $\mathbf{H}^t$ from HGC-RNN cell are used as an input to the Predictor to predict the next step taxi demand.

RNN layer Φ^{RNN} is defined as GRU, and the size of hidden states is 16. The state decoder Φ^{DEC} is defined as 2-layer MLP. HGC-RNN is trained with Adam optimizer using a learning rate of 0.001. While training, we first pretrain the model to minimize L2 loss, and retrain the pre-trained model to minimize L1 loss. When testing the models, we exclude the demand data with less than ten taxi-demand because they are of little interest in the practical application (the same setup is applied all the tests with other comparing models).

Several previous studies have developed task-specific models in taxi demand prediction. We compare our result to previous models: ARIMA, XGBoost[1], STResNet[26], DMVST-Net[22], and STDN[21].

Additionally, as ablation study, we compare the performances between HGC-RNN and its variants using conventional graph neural networks: GC-RNN using graph convolution method [9], graph neural network approach, to aggregate the neighboring node information, RN-RNN using relation network[12], a message-passing based graph neural network, in aggregating the neighboring node information. Hypergraph convolution can say to be a generalization of the relation network, in that the relation network learns the relationship between two nodes while the hypergraph convolution learns the relationship between a flexible number of nodes.

Results. Table 1 shows the RMSE and MAPE result on predicting single-step ahead NYC-Taxi demand. Because all the baseline methods are designed for a 1 step prediction task, we limit the experiment to predict 1 step ahead, even though our model is capable of predicting multi-step ahead of future demand. These performances are averaged ones for all the regions and the test period. The entire experiments were conducted ten times, and the averaged errors are provided with their standard errors (one standard deviation).

We see that the HGC-RNN-M model has similar or slightly higher errors comparing to those of STDN that have the lowest errors among the baselines. However, HGC-RNN-M model always outperforms other baselines like DMVST-Net or STResNet. Although the proposed model does not excel the state-of-art baseline models in terms of prediction accuracy, our model is constructed using more than 1000 times smaller number of model parameters, which is beneficial in reducing the training time, avoiding over-fitting for better generalization. Table 1 shows that the proposed model has a much smaller variance (standard deviation) than the baseline models. Note that baseline models like STDN[21], DMVST-Net[22], STResNet[26] are explicitly designed to predict single-step ahead taxi demand by considering various task-specific features, such as traffic flow data or weather data. However, our proposed model uses only the taxi demand data and the context information defined above, focusing on validating the generality of the model structure.

In comparing with conventional graph neural network-based modes (GC-RNN and RN-RNN), HGC-RNN-M performs the best, which validates the effectiveness of using a hypergraph instead of using a conventional pair-wise graph in this experiment.

5.2 Gas Pressure Regulator Network

Objectives. In a city, natural gas is supplied from the central provider to the end-users through a distributed gas pipeline network. A large number of gas pressure regulators concurrently controls the volume and pressure of the gas to maintain the gas flow stable. Typically, gas regulators measure the inlet and outlet gas flow continuously to monitor the operation of the gas distribution network. These inlet and outlet pressures of the entire gas regulators in the gas distribution network concurrently vary while affecting each other and, at the same time, are influenced by the external factors, such as whether and times. The accurate prediction of such gas pressures over the network and time would greatly facilitate the operation of the gas distribution network.

In this experiment, we employ the HGC-RNN model to predict the future sequence of gas pressure for the networked 195-regulators in a city. We mainly show that a hypergraph is indeed an effective representation for modeling a distributed and networked gas regulators, in that it can embed the mutual and higher-order-interaction among more than two regulators into the computation procedure of spatio-temporal modeling for gas pressure variation. We mainly investigate the superiority of HGC-RNN in the modeling of large-scale real-world network data.

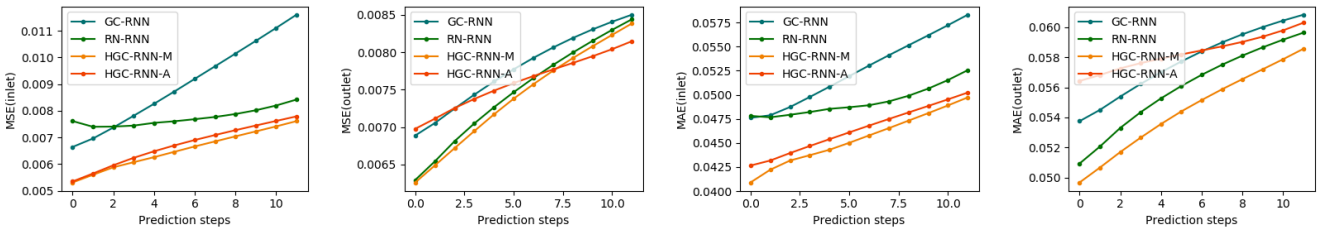
Dataset Description. The gas regulator pressure data was collected from 195 regulators in a city of South Korea between 01/01/2013 and 01/11/2014 with the 30-minutes time interval. We use the first one year of data as the training data and the rest ten months of data as the test data. The input, the output, and the environmental context data are defined as:

- $\mathbf{X}^t = \{\mathbf{x}_1^t, ..., \mathbf{x}_{195}^t\} \in \mathbb{R}^{195\times 2}$ where $\mathbf{x}_n^t \in \mathbb{R}^2$ is the inlet and outlet gas pressure at the n-th regulator.
- $\mathbf{Y}^t = \{\mathbf{y}_1^t, ..., \mathbf{y}_{195}^t\} \in \mathbb{R}^{195\times 2}$ where $\mathbf{y}_n^t \in \mathbb{R}^2$ is the predicted inlet and outlet gas pressure at the n-th regulator.
- $G = (V, E)$: Hypergraph representing the gas pressure regulators in the city and their interconnections. Each node denotes a regulator ($|\mathbf{V}| = 195$), and each hyperedge represents the set of regulators that are mutually connected by gas pipelines ($|\mathbf{E}| = 170$). The detailed definition of the hypergraph is in the Appendix.

Model Specification and Baselines. In sequence to sequence prediction setup, the input sequence length is $R = 24$ (12 hours), and the output sequence length is $L = 12$ (6 hours).

Table 2: Performance comparison on predicting gas pressure

		MSE($\times 10^3$)		MAE($\times 10^2$)	
		1 step	*12 step*	*1 step*	*12 step*
FC-RNN	*(Inlet)*	19.789	22.787	9.2258	9.5923
	(Outlet)	14.090	19.472	7.1924	9.4334
GC-RNN	*(Inlet)*	6.422 ± 0.004	11.100 ± 0.014	4.7603 ± 0.0013	5.8134 ± 0.0015
	(Outlet)	6.859 ± 0.002	8.517 ± 0.002	5.4067 ± 0.0009	6.1155 ± 0.0003
RN-RNN	*(Inlet)*	7.431 ± 0.004	8.490 ± 0.003	4.7677 ± 0.0004	5.2547 ± 0.0005
	(Outlet)	6.268 ± 0.002	8.394 ± 0.005	5.1210 ± 0.0007	5.9894 ± 0.0004
HGC-RNN-M	*(Inlet)*	**5.055 ± 0.007**	**7.647 ± 0.002**	**4.0631 ± 0.0011**	**4.9769 ± 0.0007**
	(Outlet)	**6.201 ± 0.003**	8.341 ± 0.003	**4.9883 ± 0.0008**	**5.8733 ± 0.0005**
HGC-RNN-A	*(Inlet)*	5.179 ± 0.003	7.876 ± 0.005	4.2549 ± 0.0010	5.0434 ± 0.0005
	(Outlet)	7.015 ± 0.001	**8.250 ± 0.002**	5.6807 ± 0.0004	6.0716 ± 0.0005



(a) (b) (c) (d)

Figure 5: Multi-step ahead prediction for the gas pressure

Along with node feature matrix $\mathbf{X}^t$ (demand records over all the regions), we used the following external features:

- $\mathbf{t}^t \in \mathbb{R}^{22}$: The concatenation of 1 scalar value representing time-of-the-day (1), and four one-hot vectors representing day-of-the-week (7), month (12), whether it is a holiday (1), whether it is the day before a holiday (1), respectively. The timestamp is encoded with 2-layer MLP, and the encoded hidden vector is used along with $\mathbf{X}^t$ as an input to the HGC-RNN cell.
- $\mathbf{w}^t \in \mathbb{R}^8$: The weather information including temperature, precipitation, wind speed, humidity, air pressure, amount of sunshine, amount of snowfall, and amount of clouds. The weather data is encoded with 2-layer MLP, and the encoded hidden vector is used along with $\mathbf{X}^t$ as an input to HGC-RNN cell.

RNN layer Φ^{RNN} is defined as 2-stacked GRU, whose size of hidden states is set as 256. The state decoder Φ^{DEC} is defined as 2-layer MLP. We use Adam optimizer with a learning rate of 1e-4 to minimize L2 loss. Because the scales of the inlet and outlet pressure are different, we have used normalized pressures values, which facilitates comparing the error metrics of the proposed and the baseline models.

We compare the performance of the proposed model with the three baseline models: FC-RNN, GC-RNN, and RN-RNN. FC-RNN runs on each node independently without considering the interaction among nodes, and the other two models have the same strictures with ones used in the previous taxi-demand prediction experiments.

Results. We compare the prediction errors for different models over different time steps. Table 2 summarizes the MSE and MAE error metrics for the proposed and other baseline models. It compares the averaged error metrics for 195 target nodes in predicting 1-step and 12-step ahead regulator's inlet and outlet pressures.

In addition, Figure 5 shows how the performance metrics for different models vary with the number of future prediction steps. In general, the level of errors generally increases as the prediction step increases. HGC-RNN-M shows that the proposed model outperforms other baseline models in terms of all the two error metrics.

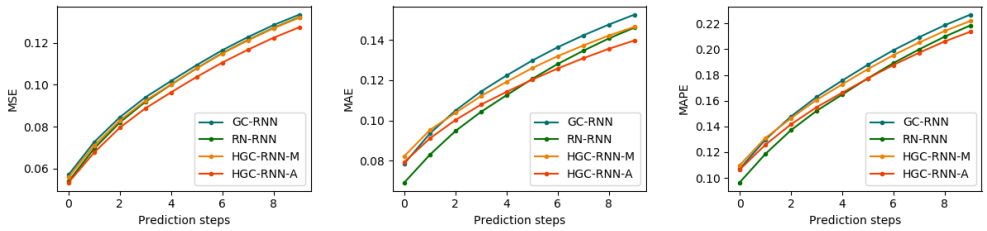
Because the gas network mutually connects more than two regulators, it is not adequate to use a pairwise edge between two nodes (regulators). Instead, we construct a hypergraph for nodes if a gas pipeline network connects the nodes. By employing a hypergraph, besides, we can reduce the number of edges from 299 to 170, which might have contributed to reducing the prediction errors.

5.3 OHT System Traffic Flow

Objectives. In a semiconductor factory, hundreds of autonomous vehicles are operating in the overhead hoist transportation (OHT) system, transporting parts and products from machines to machines. The optimal operation of the OHT system can maximize the production efficiency of semiconductors.

Table 3: Performance comparison on predicting OHT System Traffic Flow

	MSE($\times 10^2$)		MAE($\times 10^2$)		MAPE($\times 10^2$)	
	1 step	*10 step*	*1 step*	*10 step*	*1 step*	*10 step*
FC-RNN	6.14	13.84	8.74	15.83	12.28	23.89
GC-RNN	5.70 ± 0.02	13.36 ± 0.02	7.87 ± 0.02	15.25 ± 0.01	10.70 ± 0.03	22.69 ± 0.03
RN-RNN	5.38 ± 0.03	13.23 ± 0.03	**6.93 ± 0.02**	14.61 ± 0.02	**9.65 ± 0.04**	21.85 ± 0.03
HGC-RNN-M	5.59 ± 0.01	13.21 ± 0.03	8.24 ± 0.01	14.65 ± 0.01	10.99 ± 0.02	22.19 ± 0.04
HGC-RNN-A	**5.33 ± 0.02**	**12.75 ± 0.02**	7.94 ± 0.02	**13.97 ± 0.01**	10.65 ± 0.02	**21.36 ± 0.03**



(a) (b) (c)

Figure 6: Multi-step ahead prediction on OHT traffic flow

In this experiment, we particularly investigate how HGC-RNN can effectively predict the future traffic flows on OHT rail segments using partial traffic flow observations obtained from only a subset of rail segments chosen randomly from the entire rail segments.

Dataset Description. We constructed a 20-bay FAB simulator with *AutoMod*, software for simulation of production and logistics systems. The simulated system consists of 184 rail segments connected to each other, and the number of the vehicle used is 45, whose maximum speed is two *m/s*. We record the average speed of the vehicle on the entire rail segments. The time interval between data points is 5 seconds. The data is collected for 25 hours in total, and we use the first 20 hours data as the training data and remaining 4 hours data as the test data.

From the entire 184 segments, we randomly selected 73 of them (40%) and obtained observation from this subset of the entire rail segments. Then, the dependency among the data for the selected rail segments is prone to be lost due to the missing information of the rail segments excluded. We hypothesis that a hypergraph can be used to effectively infer the missing information in the hidden (unsampled) rail segments and predict the future traffic flow for the target (sampled) rail segments.

- $\mathbf{X}^t = \{\mathbf{x}_1^t, ..., \mathbf{x}_{73}^t\} \in \mathbb{R}^{73\times 1}$ where $\mathbf{x}_n^t \in \mathbb{R}$ is the average speed of node n (n-th rail segment) at time t
- $\mathbf{Y}^t = \{y_1^t, ..., y_{73}^t\} \in \mathbb{R}^{73\times 1}$ where y_n^t is the predicted average speed of vehicles at time step t on nth rail segment.
- $G = (V, E)$: Hypergraph representing the rail segments and their dependency. Each node represents the sampled rail segments ($|\mathbf{V}| = 73$), and each hyperedge denotes the rail segments that are connected by a designed path (route) of the vehicle ($|\mathbf{E}| = 70$). The detailed description of the construction of the hypergraph is described in the Appendix.

Model Specification and Baselines. In sequence to sequence prediction setup, the input sequence length is $R = 20$ (100 seconds), and the output sequence length is $L = 10$ (50 seconds). We do not use any external contextual features for this experiment.

RNN layer Φ^{RNN} is defined as 2-stacked GRU, whose size of hidden states is set as 16. The state decoder Φ^{DEC} is defined as 2-layer MLP. We use Adam optimizer with a learning rate of 0.001, to minimize L2 loss.

We test three comparable models FC-RNN, GC-RNN, and RN-RNN, which are the same as those used in the taxi demand prediction problem.

Results. Table 3 describes MSE, MAE, and MAPE results of each model, for the 1-step and the 10-steps prediction. For the 1-step ahead prediction, there is no distinct difference in the error metrics. However, for the 10-step ahead prediction, HGC-RNN-A has achieved consistently lower average error than other models. This is because the interaction among rail segments appears not instantly but gradually as the time step increases due to the time for vehicles to move over the rail network. This trend can be more clearly seen in Figure 6.

We have hypothesized that the hidden feature vectors of hyperedges are related to the actual average speed of missing rail segments. To validate this, we visualized the hidden feature vectors for all the hyperedges using t-SNE[16], as shown in Figure 7. The location of each point in the figure represents the reduced hidden representation, and the color indicates the true average speed of the missing rail segments that should have been contained by the hyperedge if it were not missed. We can see that the edges with higher speed are well-segmented from the rest of the points, implying that the hidden feature vector of each hyperedge embeds well the current state of the corresponding missing rail segments.

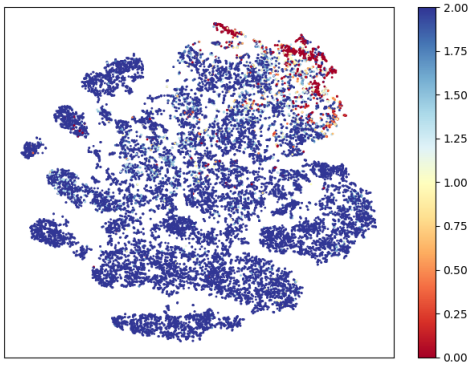


Figure 7: Result of t-SNE on hyperedge features in OHT experiment

6 CONCLUSION

In this study, we propose a hypergraph convolutional recurrent neural network(HGC-RNN) model to analyze the structured, sequential data. HGC-RNN employs a recurrent neural network structure to learn temporal dependency from the sequence of data while considering the spatial relationships in structured data by employing hypergraph convolution.

We present the evaluation of HGC-RNN on several experiments to verify its capability in various aspects. First, from the taxi demand prediction experiment, we show that HGC-RNN can accurately model the spatial and temporal evolution of data with a significantly smaller number of model parameters. In the experiment on gas regulator data, we show that a hypergraph is indeed an effective representation for modeling complex networked dynamic systems, especially for the case where the target system has a higher-order mutual interaction among nodes. In the experiments on the OHT dataset, we empirically show that the state of missing graph elements can be encoded into the hyperedge, through iterative information aggregation steps.

The strength of HGC-RNN is that it is capable of capturing not only the structural relationships but also the unknown factors affecting the behavior of the target system through the iterative aggregating procedure from the networked data. In the future, we plan to extend the model so that it can learn how to construct a time-varying hypergraph optimally from the given spatial-temporal data set.

ACKNOWLEDGMENTS

This research was supported by "Development of Big Data Platform and Centers for Environmental Business", Korea, funded by National Information Society Agency (N01200875).

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *KDD*. ACM, 785–794.
[2] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph Neural Networks. In *AAAI*. AAAI Press, 3558–3565.
[3] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*. ACM, 855–864.
[4] Daniel Hsu. 2017. Anomaly Detection on Graph Time Series. *CoRR* abs/1708.02975 (2017).
[5] Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. 2019. Dynamic Hypergraph Neural Networks. In *IJCAI*. ijcai.org, 2635–2641.
[6] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *CoRR* abs/1611.07308 (2016).
[7] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster)*. OpenReview.net.
[8] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *ICLR (Poster)*. OpenReview.net.
[9] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *AAAI*. AAAI Press, 4602–4609.
[10] Junyoung Park and Jinkyoo Park. 2019. Physics-induced graph neural network: An application to wind-farm power estimation. *Energy* 187 (08 2019), 115883. https://doi.org/10.1016/j.energy.2019.115883
[11] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin A. Riedmiller, Raia Hadsell, and Peter W. Battaglia. 2018. Graph Networks as Learnable Physics Engines for Inference and Control. In *ICML (Proceedings of Machine Learning Research)*, Vol. 80. PMLR, 4467–4476.
[12] Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *NIPS*. 4967–4976.
[13] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *NIPS*. 802–810.
[14] Chen Sun, Per Karlsson, Jiajun Wu, Joshua B. Tenenbaum, and Kevin Murphy. 2019. Stochastic Prediction of Multi-Agent Interactions from Partial Observations. In *ICLR (Poster)*. OpenReview.net.
[15] Ke Tu, Peng Cui, Xiao Wang, Fei Wang, and Wenwu Zhu. 2018. Structural Deep Embedding for Hyper-Networks. In *AAAI*. AAAI Press, 426–433.
[16] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605. http://www.jmlr.org/papers/v9/vandermaaten08a.html
[17] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR (Poster)*. OpenReview.net.
[18] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. 2018. NerveNet: Learning Structured Policy with Graph Neural Networks. In *ICLR (Poster)*. OpenReview.net.
[19] Ying Xu and Dongsheng Li. 2019. Incorporating Graph Attention and Recurrent Architectures for City-Wide Taxi Demand Prediction. *ISPRS Int. J. Geo-Information* 8, 9 (2019), 414.
[20] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha P. Talukdar. 2019. HyperGCN: A New Method For Training Graph Convolutional Networks on Hypergraphs. In *NeurIPS*. 1509–1520.
[21] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. 2019. Revisiting Spatial-Temporal Similarity: A Deep Learning Framework for Traffic Prediction. In *AAAI*. AAAI Press, 5668–5675.
[22] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. In *AAAI*. AAAI Press, 2588–2595.
[23] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek F. Abdelzaher. 2017. DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing. In *WWW*. ACM, 351–360.
[24] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *IJCAI*. ijcai.org, 3634–3640.
[25] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V. Chawla. 2019. A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data. In *AAAI*. AAAI Press, 1409–1416.
[26] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. In *AAAI*. AAAI Press, 1655–1661.
[27] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *NeurIPS*. 5171–5181.
[28] Ruochi Zhang, Yuesong Zou, and Jian Ma. 2019. Hyper-SAGNN: a self-attention based graph neural network for hypergraphs. *CoRR* abs/1911.02613 (2019).
[29] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with Hypergraphs: Clustering, Classification, and Embedding. In *NIPS*. MIT Press, 1601–1608.

A APPENDIX

A.1 Hypergraph Construction

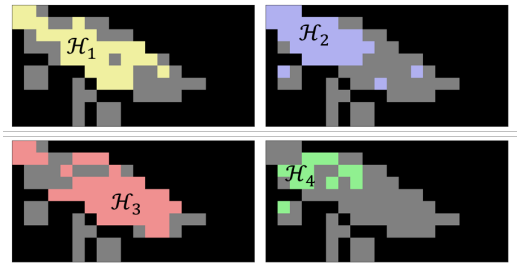


Figure 8: Four examples of hyperedges on NYC-Taxi data. The black-colored areas indicate regions with an average daily demand of less or equal to 10, and the gray-colored areas indicate regions with an average daily demand of more than ten (de facto target area), and the regions with different color indicate the four different hyperedges among 42 hyperedges constructed.

A.1.1 NYC-Taxi Demand. Taxi demand patterns vary depending on the region of the city and the time. We construct a hypergraph to effectively represent such spatial and temporal patterns of the taxi demand. One approach would be to construct a hypergraph over the regions with similar characteristics (business districts, residential areas, etc.). However, in this study, we do not have such semantic information, and we thus construct hypergraph considering the spatial and temporal demand patterns, as regions with similar characteristics will have similar patterns of demands. That is, we define a hyperedge as a set of regions with a similar taxi-demand quantity during a specific time domain.

The procedure of constructing a hypergraph for the NYC-taxi demand data is as follows:

(1) From the entire taxi demand data $\{\mathbf{X}^t | t = 1, ..., T\}$ We segment the data depending on a date and a time of day. That is, we segment $\{\mathbf{X}^t | t = 1, ..., T\}$ into $Q = 7 \times 24$ groups. The group of data matrix $\{\mathbf{X}^t | t \in q\}$ that is belong to temporal group q represents the taxi demand data over the entire regions at a specific date and a time (e.g.,q = 9 AM to 10 AM in Monday).
(2) For the taxi demand data at a specific temporal group q, we identify the regions $v_n | \mathbf{x}_n^t > (Md)^q and t \in q$, where $(Md)^q$ is the median of the $\mathbf{x}_n^t | t \in q$, and assign them as a hyperedge $\mathcal{H}^q$. We now have a total of Q hyperedges $\{\mathcal{H}^q | q = 1, ..., Q = 7 \times 24\}$.
(3) Given the identified hyperedge $\{\mathcal{H}^q | q = 1, ..., Q\}$, we merge some of hyperedges such that the similar spatial demand patterns are treated as a single hyperedge. To measure similarity between two hyperedges, $\mathcal{H}_i and \mathcal{H}_j$, we use Sørensen-Dice coefficient defined as

$$DSC = \frac{2|\mathcal{H}_i \cap \mathcal{H}_j|}{|\mathcal{H}_i| + |\mathcal{H}_j|}$$

where $|\mathcal{H}|$ is the size of the set $\mathcal{H}$.

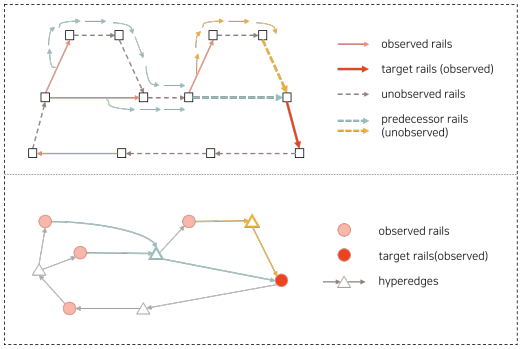


Figure 9: Construction of hyperedge for the OHT System

(4) Continue the merging process until the Sørensen-Dice coefficients for any pair of hyperedges become less than 0.8.

We finally obtain 42 distinct hyperedges in the NYC taxi data set. Figure 8 shows the four representative hyperedges constructed above procedure.

A.1.2 Gas regulator network data. Gas regulators are connected by a pipeline network. If we assign an edge between two regulators that are connected to each other, in using the conventional pairwise edge graph, we need a total of 299 edges, as shown in Figure 10(a).

Instead of using the pairwise edge construction, in this study, we construct a hypergraph to denote a set of regulators that are mutually connected. For example, a pipeline has thee branches to connect three regulators in the gas distribution network. As a result, we finally construct a total of 170 hyperedges, as shown in Figure 10(b).

A.1.3 OHT System Traffic Flow. The rail segments of an OHT system can be considered as a graph. Each rail segment is then considered as a node. Because the direction of the vehicle is predetermined in an OHT system, we can use this flow direction information to construct edges. Typically, a rail segment is connected by edges with the preceding and the following rail segments, respectively.

However, these well-defined graph concepts are prone to be lost when some of the graph elements are missing. Assuming that some of the original graph elements are missing, we have constructed the subset of the OHT rail networks using both the conventional graph and hypergraph representation, as shown in Figure 9. For the conventional graph, we construct an edge when the destination node is reachable from the origin node through successors of unobserved rails. In the case of a hypergraph, we construct a hyperedge over the set of nodes that are connected through the missing rail segments (unobserved rail segments). More specifically, the set of unobserved rail segments are replaced by a hyperedge that contains the rail segments that used to be connected with the unobserved rail segments. In the experiment, we only use 40% of the total rail segments, as shown in Figure 11(a). The full graph, including the entire rail segments, is shown in Figure 11(b). Using the subset of rail segments, we construct a conventional graph as in Figure 11(c) and a hypergraph as in Figure 11(c), respectively. The number of total edges is for the conventional graph is 164, and the total number of hyperedges is 70.

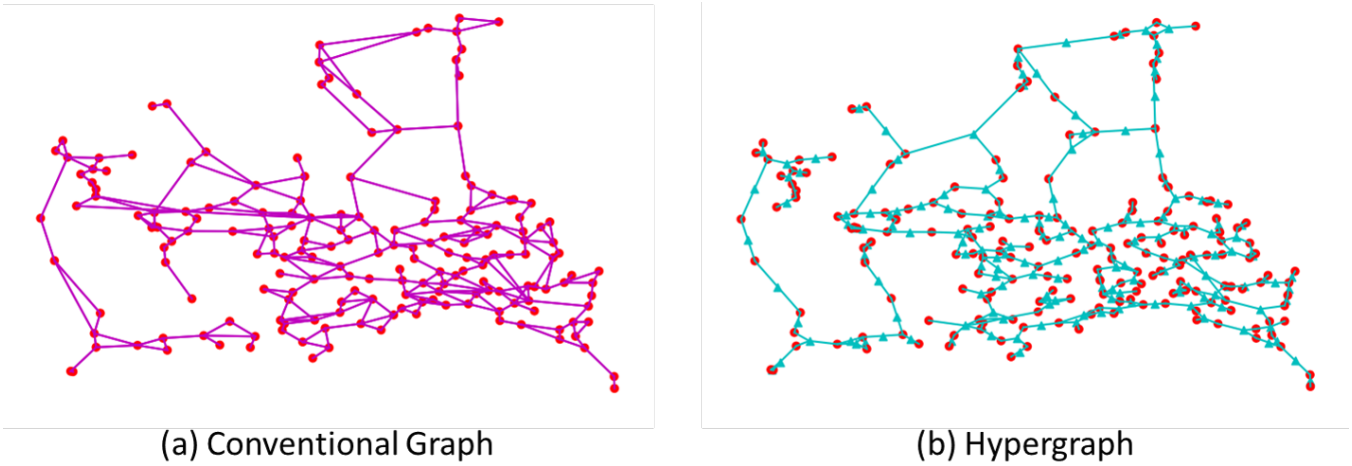


Figure 10: Construction of hyperedge for gas regulator network

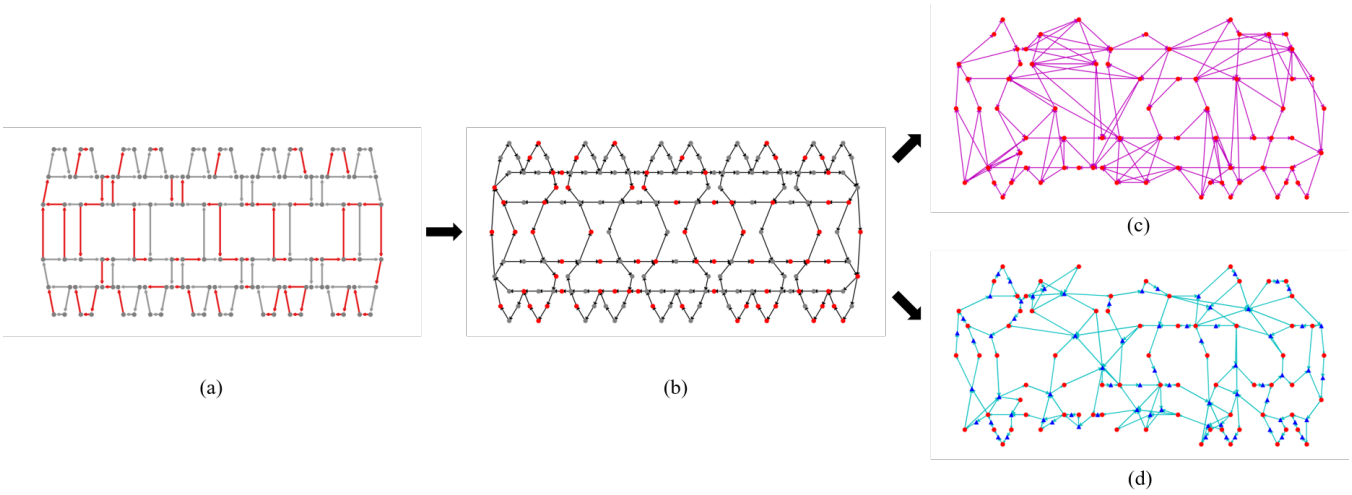


Figure 11: Generation of the conventional graph and hypergraph representation for OHT System. (a) Overall layout of rails in OHT system. (b) Directed graph representation of full OHT system. (c) Directed graph representation of target rails. (d) Directed hypergraph representation of target rails. Red rails in (a) and red nodes in (b) indicates target rails, while gray rails in (a) and gray nodes in (b) indicates missing rails