Pattern-Matching Dynamic Memory Network for Dual-Mode Traffic Prediction

Wenchao Weng, Mei Wu, Hanyu Jiang, Wanzeng Kong, Senior Member, IEEE, Xiangjie Kong, Senior Member, IEEE, and Feng Xia, Senior Member, IEEE

Abstract-In recent years, deep learning has increasingly gained attention in the field of traffic prediction. Existing traffic prediction models often rely on GCNs or attention mechanisms with $O(N^2)$ complexity to dynamically extract traffic node features, which lack efficiency and are not lightweight. Additionally, these models typically only utilize historical data for prediction, without considering the impact of the target information on the prediction. To address these issues, we propose a Pattern-Matching Dynamic Memory Network (PM-DMNet). PM-DMNet employs a novel dynamic memory network to capture traffic pattern features with only O(N) complexity, significantly reducing computational overhead while achieving excellent performance. The PM-DMNet also introduces two prediction methods: Recursive Multi-step Prediction (RMP) and Parallel Multi-step Prediction (PMP), which leverage the time features of the prediction targets to assist in the forecasting process. Furthermore, a transfer attention mechanism is integrated into PMP, transforming historical data features to better align with the predicted target states, thereby capturing trend changes more accurately and reducing errors. Extensive experiments demonstrate the superiority of the proposed model over existing benchmarks. The source codes are available at: https://github.com/wengwenchao123/PM-DMNet.

Index Terms—Traffic Prediction, Memory Network, Transfer Attention, Traffic Pattern, Time Embedding

I. INTRODUCTION

W ITH the development of society and technology, there has been a significant increase in vehicles within cities, as well as the growing popularity of services like shared bicycles and ride-hailing platforms such as Uber and Didi. This expansion has broadened the application of urban traffic management by governments and heightened public transportation demands. However, issues such as limited resources and inadequate scheduling systems have increasingly highlighted challenges in traffic management and the imbalance of transportation demand. As a result, accurate traffic

This work was supported in part by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang under Grant 2024C01214, and in part by the National Natural Science Foundation of China under Grant 62072409. (corresponding author: Xiangjie Kong.)

Wenchao Weng and Xiangjie Kong are with the College of Computer Science and Technology, Zhejiang Universityof Technology, Hangzhou 310023, China (e-mail: 111124120010@zjut.edu.cn; xjkong@ieee.org).

Mei Wu and Hanyu Jiang are with the Hangzhou Dianzi University ITMO Joint Institute, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: 222320007@hdu.edu.cn; 22320324@hdu.edu.cn).

Wanzeng Kong is with College of Computer Science, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: kongwanzeng@hdu.edu.cn).

Feng Xia is with School of Computing Technologies, RMIT University, Melbourne, VIC 3000, Australia (e-mail: f.xia@ieee.org). forecasting has become a crucial issue in fields such as traffic management, urban planning, and the sharing economy. Precise traffic prediction enables governments to better allocate social resources to maintain urban transportation operations. It also allows companies to distribute resources such as shared bicycles and taxis to areas with high demand, thereby avoiding their idle presence in low-demand areas. This approach can reduce energy consumption and passenger waiting times.

In recent years, researchers have conducted extensive studies in traffic prediction to promote the development of intelligent transportation systems. Early traffic prediction methods utilized statistical approaches for prediction. Auto-regressive (AR), Moving Average (MA), and Auto-Regressive Integrated Moving Average (ARIMA) models [1], as the most representative classical statistical methods, have been extensively employed in traffic prediction. Additionally, machine learning techniques represented by Support Vector Regression (SVR) [2] and Kalman filters [3] have also been applied to traffic prediction to achieve more accurate predictions and handle more complex sequences. However, these methods require data to exhibit stationarity to be effective, which limits their ability to capture the intricate non-linear spatio-temporal correlations present in traffic condition.

In recent years, the advancements of deep learning in domains such as Computer Vision and Natural Language Processing have motivated researchers to explore its application in traffic prediction for improved outcomes. Early deep learning prediction models conceptualized urban traffic as images and segmented them into grids. Convolutional Neural Networks (CNNs) [4] were employed to analyze spatial correlations within these grids, while Recurrent Neural Networks (RNNs) [5], [6], [7] or CNNs [8], [9] were utilized to capture temporal dependencies. However, the structure of the transportation network can be viewed as a topological graph, containing non-Euclidean attributes. CNNs only extract features from the surrounding nodes and cannot capture features from other locations across space. As Graph Convolutional Networks (GCNs) [10] are effective in handling non-Euclidean structures, it has been widely applied in the field of transportation [11], [8], [6]. Additionally, attention mechanisms [12], [13], [14] have been incorporated for spatio-temporal feature modeling.

However, current methods still possess the following limitations:

1) Lack of Effective Traffic Feature Extraction: Traffic data inherently exhibits complex spatio-temporal correlations. To capture these spatio-temporal correlations, researchers have employed GCN to capture spatial relationships between nodes,



(a) Graph Convolution Network (b) Dynamic Memory Network (GCN). (DMN).

Fig. 1: Comparison between GCN and DMN. As M is constant, the time complexity of GCN and DMN is $O(N^2)$ and O(N), respectively.

achieving significant success. As shown in Figure 1(a), current methods require evaluating the correlations between all pairs of nodes to dynamically generate the graph structure and then use GCN to extract spatio-temporal correlations [6], [11], resulting in an $O(N^2)$ computational complexity. However, in practical scenarios, the structure of transportation networks often exhibits sparsity, meaning that nodes are only correlated with a subset of other nodes, and most nodes do not have correlations with each other. As illustrated in Figure 2(a), Nodes A, B, and C exhibit evident correlations, representing a specific traffic pattern, while Nodes D and E signify another traffic pattern. Computing similarities between Nodes A, B, C, and Nodes D, E would be meaningless and resource-intensive. Recent studies [15], [16], [17] have focused on reducing computational complexity, but they each come with limitations. For instance, STWave [17] introduces an MS-ESGAT (Multi-Scale Edge-based Spatial Graph Attention) mechanism to achieve linear complexity. However, this method relies heavily on predefined graph structures, making it unsuitable for scenarios where no predefined graph is available.

2) Uncertainty in Predicting Trend Changes: Figure 2(b) illustrates two sets of historical data and their corresponding prediction targets, where the red segment represents historical data and the yellow segment represents the prediction target. As shown, the left side's historical data and corresponding prediction targets remain within a stable trend channel. However, on the right side, although the historical data is also within a stable trend channel, the corresponding prediction target shifts into a downward trend channel. This indicates that relying solely on historical data for prediction makes it challenging to capture such trend shifts. Although current studies [18], [5], [19] have proposed various methods to extract spatiotemporal features, they rely exclusively on historical data to model traffic conditions, leading to limitations in accurately capturing the trend changes of prediction targets.

To address the above issues, a novel Pattern-Matching Dynamic Memory Network (PM-DMNet) model for traffic prediction is proposed in this paper. For the first challenge, a Dynamic Memory Network (DMN) is designed to extract pattern features from nodes. Specifically, a learnable memory matrix is defined to learn representative traffic patterns within



Fig. 2: The findings about traffic data.

the traffic conditions. The traffic features input to the model are then used in conjunction with these embeddings to compute a pattern attention matrix, which facilitates the extraction of features from the most similar traffic patterns. Simultaneously, the DMN dynamically adjusts the representative traffic patterns at each time point by combining time embeddings with memory embeddings, thus avoiding issues related to traffic pattern homogenization. Moreover, as illustrated in Figure 1, compared to the high computational complexity of GCN, which is $O(N^2)$, this method reduces the computational complexity to O(N), significantly enhancing computational efficiency.

To address the second challenge, two prediction methods are designed: Recurrent Multi-step Prediction (RMP) and Parallel Multi-step Prediction (PMP). RMP uses the traditional recursive approach, where predictions are made during the decoding phase by recursively utilizing the time features and extracted hidden features for the target time points. PMP directly uses the time features for the target time points and the hidden features extracted from historical data for prediction. To mitigate the errors caused by discrepancies between historical data and prediction targets, a novel Transition Attention Mechanism is introduced in PMP. Specifically, this attention mechanism leverages the inherent periodicity in traffic data by integrating the input data, its time features, and the time features of the prediction targets. This transforms the hidden states to better align with the conditions of the target time points. This method enhances the adaptability of the extracted latent features to the prediction target states, improving accuracy. Furthermore, PMP reduces the required computation time compared to RMP, as it does not involve recursion, and it also enhances prediction performance.

In summary, the contributions of this paper can be summarized as follows:

 We present a new traffic prediction model, named Pattern Matching Dynamic Memory Network (PM-DMNet). This model can achieve both Parallel Multi-step Prediction (PMP) and Recurrent Multi-step Prediction (RMP) in the decoder stage depending on the requirements. Compared to RMP, PMP avoids the cyclic recursion process, thereby enhancing computational efficiency.

- We propose a novel Dynamic Memory Network (DMN) module designed to learn inherent representative traffic patterns within the data associated with each node. By employing a pattern matching approach, this module identifies and extracts traffic pattern features most similar to the input data while effectively reducing computational overhead.
- We introduce a new Transfer Attention Mechanism (TAM). TAM transforms the existing historical hidden states into latent states aligned with the prediction target features, mitigating the error caused by the discrepancy between historical data and prediction targets.
- Experimental results on ten authentic datasets substantiate that our proposed framework significantly outperforms state-of-the-art methods across all datasets.

II. RELATED WORK

A. Spatio-Temporal Prediction

As one of the most representative tasks in spatio-temporal prediction, researchers employed a myriad of methodologies to model the spatio-temporal characteristics within traffic condition. STGCN [20] leveraged GCN and predefined matrices to capture spatial correlations between nodes, employing Gate CNNs to model such spatial dependencies. DCRNN [7] integrated diffusion convolution with GRU to model the spatiotemporal relationships inherent in traffic condition. MTGNN [21] utilized adaptive embeddings to generate an adaptive graph structure, capturing spatial correlations among diverse nodes. CCRNN [22] introduced a novel graph convolutional structure termed as CGC and employed a hierarchical coupling mechanism, linking upper-layer graph structures with underlying ones to extract temporal-spatial features. GMAN [13] harnessed three distinct attention mechanisms to capture the spatio-temporal characteristics present in traffic condition. MPGCN [15] utilized GCN to identify mobility patterns at bus stops through clustering and employed GCN2Flow to predict passenger flow based on various mobility patterns. Building on the foundation of MPGCN, MPGNNFormer [16] designed a STGNNFormer to extract both temporal and spatial dependencies. Although these spatiotemporal prediction models have achieved notable success, the GCNs and attention mechanisms they use often require $O(N^2)$ or even higher complexity, resulting in substantial computational costs.

B. Neural Memory Network

The Memory Network [23] introduced an external memory mechanism, enabling it to better handle and utilize longterm information. Memory networks have found extensive applications in the domains of natural language processing and machine translation. MemN2N [24] introduced a novel endto-end memory network framework that facilitates its straightforward application in real-world environments. Kaiser et al.

[25] proposed memory networks with the capability to adapt to various zero-shot scenarios. Mem2seq [26] integrated multihop attention mechanisms with memory networks, enabling their deployment in dialog systems. MemAE [27] explored the application of memory networks in video anomaly detection tasks, subsequent studies [28] validating the feasibility of this approach. MTNet [29] endeavored to apply memory networks in multi-variate time series prediction, yielding promising results. In the most recent advancements, PM-MemNet [30] devised a novel Graph Convolutional Memory Network (GCMem) to model the spatio-temporal correlations inherent in given traffic condition. Additionally, MegaCRN [31], inspired by memory network principles, designed a Meta-Graph Learner to construct dynamic graphs, addressing temporal-spatial heterogeneities. Although memory networks have been applied in traffic prediction, they still require integration with other feature extraction methods (e.g., GCN) to perform effectively.

Unlike previous spatio-temporal prediction models, PM-DMNet uses a dynamic memory network to extract traffic pattern features, achieving superior performance while reducing complexity to O(N), which significantly lowers computational costs. Additionally, prior research overlooks the impact of time features corresponding to the prediction targets on the targets themselves. PM-DMNet fully considers this characteristic and designs two prediction methods to utilize these time features, leading to successful outcomes.

III. PRELIMINARIES

A. Temporal Indexing Function

TABLE I: Example of time index transformation

Time	d(t)	w(t)
Monday,00:05	0:05:00	Monday
Monday,01:00	1:00:00	Monday
Thursday,01:00	1:00:00	Thursday

Given that traffic condition is collected at regular time intervals, each set of traffic condition possesses unique and systematic temporal information. To harness these temporal characteristics effectively, we employ a temporal indexing function to extract time-related information. Let d(t) and w(t) represent the intra-daily and weekly indexing functions, respectively. These functions transform the temporal information of the traffic condition into corresponding intra-daily and weekly time-related attributes. For specific examples, refer to Table I.

B. Traffic Prediction

The objective of traffic prediction is to utilize historical traffic condition to forecast future traffic condition.

We represent the traffic condition $X_t \in \mathbb{R}^{N \times C}$ for N nodes in the road network at time t, where C is the dimensionality of traffic condition, signifying C types of traffic condition. We model the historical traffic condition $X = [X_1, X_2, ..., X_n] \in$ $\mathbb{R}^{n \times N \times C}$ over the past n time steps using the model f to predict the traffic condition $Y = [Y_{n+1}, Y_{n+2}, ..., Y_{n+m}] \in$



Fig. 3: Overview of PM-DMNet structure.

 $\mathbb{R}^{m \times N \times C}$ for the future *m* time steps, which can be expressed as:

$$[X_1, X_2, ..., X_n] \xrightarrow{J} [Y_{n+1}, Y_{n+2}, ..., Y_{n+m}]$$
(1)

In addition, The corresponding actual values are represented by $\hat{Y} = [\hat{Y}_{n+1}, \hat{Y}_{n+2}, ..., \hat{Y}_{n+m}] \in \mathbb{R}^{m \times N \times C}$,



Fig. 4: The construction of TE generator.

IV. MODEL ARCHITHECTURE

Figure 3 illustrates the comprehensive architecture of PM-DMNet, which comprises a Time Embedding Generator (TE Generator), Dynamic Pattern Matching Gated Recurrent Unit (DPMGRU), and Transfer Attention Mechanism (TAM). In the subsequent sections, we will provide a detailed exposition of each module.

A. Time Embedding Generator

Traffic condition is influenced by people's travel habits and lifestyles, exhibiting clear temporal such as rush hours during mornings and evenings. To fully leverage temporal features, we introduce two independent embedding pools $T^D \in \mathbb{R}^{N_d \times p}, T^W \in \mathbb{R}^{N_w \times p}$ to learn features for intradaily and weekly patterns. Here, N_d represents the number of time slots in a day, and $N_w = 7$ represents the number of days in a week. As depicted in Figure 4, based on the time information t, we derive the intra-daily index d(t) and the weekly index w(t). Utilizing d(t) and w(t), we obtain the intra-daily time feature embedding $T_{d(t)}^D$ and the weekly time feature embedding $T_{w(t)}^W$ corresponding to the specific time point. Ultimately, these $T_{d(t)}^D \in \mathbb{R}^p$ and $T_{w(t)}^W \in \mathbb{R}^p$ are integrated to yield a combined time embedding, which can be expressed as follows:

$$T_t = T_{d(t)}^D \odot T_{w(t)}^W \tag{2}$$

where \odot denotes the hadamard product.

B. Dynamic Memory Network

The memory module incorporates a learnable memory matrix $P = [P^1, P^2, ..., P^M] \in \mathbb{R}^{M \times p}$, where symbolizes a unique traffic pattern. To dynamically adjust the memory matrix, thereby avoiding pattern singularization and adapting to the prevailing traffic conditions at time t, we integrate the current time embedding T_t with P. This fusion can be represented as:

$$P_t = P \odot T_t \tag{3}$$

where $P_t \in \mathbb{R}^{M \times p}$ represents the memory network module at time t. Through training, P_t can learn the most representative traffic patterns at time t. By integrating the time embedding T_t dynamically, the model can adjust its memory P_t to better capture evolving traffic patterns and conditions over time.

As shown in Figure 5, we extract dynamic signals from the traffic condition, which can be represented as:

$$F_t^i = MLP(x_t^i) \tag{4}$$

where $F_t^i \in \mathbb{R}^p$ represents the dynamic signal extracted from the traffic condition x_t^i at node *i*. It is used to query the memory matrix for the traffic pattern most similar to x_t^i .

Afterwards, the similarity weight between F_t^i and the memory matrix P_t is computed through a similarity calculation:

$$w_t^i = softmax(F_t^i P_t^T) \tag{5}$$

where $w_t^i \in \mathbb{R}^M$ represents the similarity weight vector.



Fig. 5: Dynamic memory network.

Subsequently, P_t is linearly transformed to obtain the pattern features corresponding to various traffic patterns. It is then multiplied with the similarity weight vector w_t^i to extract the pattern features most similar to x_t^i , as follows:

$$h_t^i = w_t^i P_t \tag{6}$$

where $h_t \in \mathbb{R}^{M \times F_{out}}$ represents the pattern features in the memory matrix P_t , and h_t^i represents the extracted traffic pattern features.

Finally, the residual connection is employed to concatenate h_t^i and x_t^i for extracting hidden features:

$$H_t^i = (h_t^i || x_t^i) \Theta \tag{7}$$

where $\Theta \in \mathbb{R}^{F_{in} \times F_{out}}$ represents learnable parameters. All node hidden states H_t^i are aggregated into $H_t = (H_t^1, H_t^2, ..., H_t^N)$, serving as the final output of the dynamic memory network.

C. Node Adaptive Parameter Learning

To enable each node to learn its unique traffic pattern, enhancing the model's robustness and effectiveness, we utilize two parameter matrices to optimize the learnable parameters Θ . Specifically, we use the node embedding matrix $E \in \mathbb{R}^{N \times d}$ and the weight pool $W \in \mathbb{R}^{d \times F_{in} \times F_{out}}$ to generate $\Theta \in \mathbb{R}^{N \times F_{in} \times F_{out}}$, which can be expressed as:

$$\begin{aligned} H_t^i &= (h_t^i || x_t^i) \Theta \\ &= (h_t^i || x_t^i) E \cdot W \end{aligned} \tag{8}$$

where \cdot represents the multiplication of matrices in different dimensions. From the perspective of an individual node, E provides d independent traffic patterns, and the node adjusts W in a data-driven way to assign appropriate weights to each pattern. These weights are combined to create the node's unique traffic pattern.

D. Dynamic Pattern Matching Gated Recurrent Unit

To capture the spatio-temporal features inherent in traffic condition, we integrate the gated recurrent unit (GRU) with a dynamic memory network to construct a framework that encapsulates both temporal dynamics and spatial correlations. Specifically, we replace the MLP layer in the GRU with a dynamic memory network, resulting in the Dynamic Pattern Matching Gated Recurrent Unit (DPMGRU). Mathematically, DPMGRU can be formulated as:

$$r_{t} = \sigma(\vartheta_{r*G}(x_{t}||H_{t-1}))$$

$$u_{t} = \sigma(\vartheta_{u*G}(x_{t}||H_{t-1}))$$

$$h_{t} = tanh(\vartheta_{h*G}(x_{t}||u_{t} \odot H_{t-1}))$$

$$H_{t} = r_{t} \odot H_{t-1} + (1 - r_{t}) \odot h_{t}$$
(9)

where X_t and H_t denote the input and output at time step t, respectively. σ represents the *sigmoid* activation function. r and u correspond to the reset gate and update gate, respectively. *G denotes the dynamic memory network module, while $\vartheta_r, \vartheta_u, \vartheta_h$ are the learnable parameters associated with the relevant memory network module.

E. Transfer Attention Mechanism

To mitigate the discrepancy between historical data and the prediction target leading to errors, we employ a transfer attention mechanism to transform the learned hidden features from historical data. Specifically, we first linearly transform the encoder's output $H_n \in \mathbb{R}^{N \times D}$, historical time embedding $T_n \in \mathbb{R}^p$, and future embeddings $T_F = (T_{n+1}, T_{n+2}, ..., T_{n+m}) \in \mathbb{R}^{m \times p}$ into queries, keys, and values, represented as:

$$Q = \forall (H_n, T_F) W^Q, K = \forall (H_n, T_n) W^K, V = \forall (H_n, T_n) W^V$$
(10)

where $W^Q, W^K, W^V \in \mathbb{R}^{(D+p) \times d_k}$ serve as learnable parameters, and \forall () denotes a broadcasting operation. Subsequently, the transfer attention can be expressed as:

$$H_{TA} = attention(H, T_F, T_n)$$

= $softmax(\frac{QK^T}{\sqrt{d_K}}V)$ (11)

Finally, the feature fusion between H_n and $H_{TA} \in \mathbb{R}^{m \times N \times D}$ is achieved using residual connections to obtain the input for the decoder:

$$H_{out} = MLP(\forall (H_n, H_{TA})) \tag{12}$$

where $H_{out} = (H_{n+1}, H_{n+2}, ..., H_{n+m}) \in \mathbb{R}^{m \times N \times D}$ correspond to the hidden features from time points n+1 to n+m for the prediction target. By employing T_N and T_F , these features undergo transfer learning to adapt more effectively to the state of the prediction target time points.

F. Encoder-Decoder Architecture

The traditional encoder-decoder architecture typically employs the Recurrent Multi-step Prediction (RMP) method for forecasting. However, recurrent decoding has inherent limitations, including: (i) error accumulation due to recurrent predictions, and (ii) the sequential nature of recursion, which restricts the model's ability for parallel computation, thus limiting the improvement of inference speed. [32] demonstrates that Parallel Multi-step Prediction (PMP) methods can achieve comparable or even better results than RMP when appropriate techniques are applied. Therefore, two variants are designed to implement and investigate these prediction methods:



6



Fig. 6: Comparison of Recurrent Multi-step Prediction (RMP) and Parallel Multi-step Prediction (PMP).

PM-DMNet(R) As illustrated in Figure 6(a), PM-DMNet(R) employs the classic Recurrent Multi-step Prediction (RMP) method, where Y_t is derived through single-step prediction. Subsequently, the predicted Y_t serves as the input for predicting Y_{t+1} , iterating this process until the complete prediction output is obtained.

PM-DMNet(P) Inspired by [32], PM-DMNet(P) adopts the Parallel Multi-step Prediction (PMP) method. As shown in Figures 6(b), during the decoding phase, the encoder's output H_n is first processed through TAM to obtain H_{out} , which alleviates the discrepancy between historical data and prediction targets, aligning it more closely with the state of the prediction targets. Subsequently, $H_{out} = (H_{n+1}, H_{n+2}, \dots, H_{n+m})$ and $T_F = (T_{n+1}, T_{n+2}, \dots, T_{n+m})$ are segmented and input into PDMGRU to predict the corresponding targets Y = $(Y_{n+1}, Y_{n+2}, \ldots, Y_{n+m})$. Since recursive compilation is not required, the prediction targets Y can be predicted in parallel, avoiding the issue of accumulating prediction errors with recursion steps.

The details of model training and prediction are presented in Algorithm 1.

V. EXPERIMENTAL SETUP

A. Datasets & Settings

In this section, experiments are conducted on ten real-world datasets to validate the effectiveness of the proposed PM-DMNet. The datasets used are categorized into four types: bike demand datasets include NYC-Bike14 [4], NYC-Bike15 [33], and NYC-Bike16 [22]; taxi demand datasets include NYC-Taxi15 [33] and NYC-Taxi16 [22]; traffic flow datasets include PEMSD4 [34], PEMSD7 [20], and PEMSD8 [34]; and traffic speed datasets include PEMSD7(M) and PEMSD7(L) [9]. Detailed information about the datasets and the training set divisions can be found in Table II. Moreover, Unlike traffic flow and traffic speed datasets, the traffic demand datasets have two dimensions: 'Pick-up' and 'Drop-off'. We set n = 12historical time steps to predict m = 12 future time steps.

All experiments are conducted on a server equipped with an NVIDIA GeForce GTX 4090 GPU. The Adam optimizer is used for model optimization, and the Mean Absolute Error (MAE) is adopted as the loss function. The hyper-parameter

Algorithm 1 Training algorithm of PM-DMNet.

- **Input:** The traffic dataset O, encoder's function $f_{en}(\cdot)$, decoder's function $f_{de}(\cdot)$, TAM's function $f_{tam}(\cdot)$, prediction type T, scheduled sampling function $f_{ss}(\cdot)$
- 1: repeat
- select a input $X \in \mathbb{R}^{n \times N \times C}$, label $\hat{Y} \in \mathbb{R}^{m \times N \times C}$, 2: time information t, initialize hidden state H_0 .
- compute $T_t = T_{d(t)}^D \odot T_{w(t)}^W$ for i in 1, 2, ..., n do 3:
- 4:
- compute $H_i = f_{en}(X[i,\ldots], H_{i-1}, T_i)$ 5:
- 6: end for
- if T = PM-DMNet(P) then 7:
- Initialize a zero tensor $Y_{in} \in \mathbb{R}^{m \times N \times C}$ as the input 8: to the decoder.
- compute $H_{out} = f_{tam}(H_n, T_N, T_F)$ 9:
- compute $Y = f_{de}(Y_{in}, H_{out}, T_F)$ 10:
- end if 11:
- if T = PM-DMNet(R) then 12:
- set iter = 1; 13:
- Initialize a zero tensor $Y_{in} \in \mathbb{R}^{N \times C}$ as the input to 14: the decoder.
- 15: for q in 1, 2, ..., m do compute $Y[q, :] = f_{de}(Y_{in}, H_{m+q-1}, T_{n+q})$ 16: compute $\varepsilon_i = f_{ss}(iter)$ 17: generate a random number $\mu \sim \mathbf{N}(0, 1)$. 18: 19: if $\mu < \varepsilon_i$ then $Y_{in} = \hat{Y}[q, \ldots].$ 20: else 21: $Y_{in} = Y[q, \ldots]$ 22: end if 23: end for 24: end if 25: Calculate loss L by using MAE. 26:
- Update model parameters according to loss L. 27: 28: **until** convergence of the model is achieved Output: learned model.

TABLE II: Statistics of datasets.

Data type	Datasets	Nodes	Time steps	Time Range	Time interval	Train/Val/Test
Bike Demand	NYC-Bike14	128	4392	04/2014 - 09/2014	1 hour	7/1/2
	NYC-Bike15	200	2880	01/2015 - 03/2015	30 min	7/1/2
	NYC-Bike16	250	4368	04/2016 - 06/2016	30 min	7/1.5/1.5
Taxi Demand	NYC-Taxi15	200	2880	01/2015 - 03/2015	30 min	7/1/2
	NYC-Taxi16	266	4368	04/2016 - 06/2016	30 min	7/1.5/1.5
Traffic Flow	PEMSD4	307	16992	01/2018 - 02/2018	5min	6/2/2
	PEMSD7	883	28224	05/2017 - 08/2017	5min	6/2/2
	PEMSD8	170	17856	07/2016 - 08/2016	5min	6/2/2
Traffic Speed	PEMSD7(M)	228	12672	05/2012 - 06/2012	5min	6/2/2
	PEMSD7(L)	1026	12672	05/2012 - 06/2012	5min	6/2/2

settings for the model under the two different prediction methods, such as the temporal embedding dimension p, node embedding dimension d, memory network matrix dimension M, batch size, and learning rate, are detailed in Table III. During training, an early stopping strategy is employed to terminate training and prevent over-fitting. Additionally, a scheduled sampling strategy [35] is applied to PM-DMNet(R) to enhance its robustness.

TABLE III: Model hyper-parameter settings.

Datasets	PM-E	DMNet(P)	PM-E	$p \frac{\text{PM-DMNet(R)}}{d}$		batchsize	learning rate
NYC-Bike14	20	10	20	10	10	64	0.03
NYC-Bike15	12	6	12	6	10	64	0.03
NYC-Bike16	20	10	20	10	10	64	0.03
NYC-Taxi15	20	10	20	10	10	64	0.03
NYC-Taxi16	20	10	20	10	10	64	0.03
PEMSD4	24	12	20	10	10	64	0.03
PEMSD7	24	12	24	12	10	64	0.03
PEMSD8	20	10	12	6	10	64	0.03
PEMSD7(M)	8	4	10	5	10	64	0.03
PEMSD7(L)	16	8	20	10	10	64	0.03

B. Baselines

To compare performance, the following 24 baselines with official code are compared with PM-DMNet:

- 1) Traditional Models:
- HA [36]: It utilizes historical averages to iteratively predict the future.
- ARIMA [1]: It integrates moving averages into an autoregressive model.
- VAR [37]: It is a statistical model capable of capturing spatial dependencies.
- 2) Machine Learning Models:
- SVR [2]: It uses support vector machines for prediction.
- XGBoost [38]: It is a classical and widely adopted machine learning model.
- 3) Deep Learning Models:
- LSTM [39]: It makes predictions through iterations.
- TCN [40]: It employs causal convolutions and dilated convolutions to capture temporal correlations.
- STGCN [4]: It uses graph convolution and onedimensional convolutional neural networks to separately extract spatial and temporal correlations.
- STGCN [9]: It combines TCN with GCN to extract spatio-temporal dependencies.
- DCRNN [7]: It combines diffusion convolution and GRU to extract spatiotemporal correlations.
- STG2Seq [41]: It captures temporal dependencies from both long-term and short-term perspectives.
- GWN [8]: It integrates gated TCN and adaptive graph GCN to capture spatiotemporal dependencies.
- ASTGCN [34]: It performs attention mechanism analysis on spatio-temporal convolutions to extract dynamic spatio-temporal correlations.
- LSGCN [42]: It uses graph convolutional networks and a novel cosine graph attention network to capture long-term and short-term spatial dependencies.
- STFGNN [20]: It designs a spatio-temporal fusion graph to capture local spatio-temporal correlations.
- STSGCN [20]: It constructs a three-dimensional graph for graph convolution to capture spatio-temporal correlations between nodes.
- MTGNN [21]: It employs self-learned adjacency matrices and a time convolution module to capture spatio-temporal correlations between different variables.
- CCRNN [5]: It designs a Coupled Layer-wise Graph Convolution for prediction.

- STFGNN [20]: It designs a spatio-temporal fusion graph to capture local spatio-temporal correlations.
- STGODE [43]: It leverages neural ODE to reconstruct GCN, alleviating the over-smoothing problem in deep GCNs.
- GTS [44]: It learns the graph structure among multiple time series and simultaneously makes predictions using GNN.
- ESG [19]: It designs an evolving structure learner to construct a series of adjacency matrices. These matrices not only receive information from the current input but also maintain the hidden states of historical graph structures.
- MVFN [18]: It uses graph convolution and attention mechanisms to extract local and global spatial features. Additionally, it employs multi-channel and separable temporal convolutional networks to extract overall temporal features.
- STWave [17]: It uses the DWT algorithm to decouple traffic data for modeling. Additionally, it designs a novel local graph attention network to efficiently and effectively model dynamic spatial correlations.
- MegaCRN [31]: It designs a Meta-Graph Learner to construct dynamic graphs, addressing temporal-spatial heterogeneities.

C. Metrics

The following three evaluation metrics are chosen to assess model performance: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Empirical Correlation Coefficient (CORR).

$$MAE = \frac{1}{\phi} \sum_{i=1}^{\phi} |Y_i - \hat{Y}_i|$$
(13)

$$RMSE = \sqrt{\frac{1}{\phi} \sum_{i=1}^{\phi} (Y_i - \hat{Y}_i)^2}$$
(14)

$$MAPE = \frac{1}{\phi} \sum_{i=1}^{\phi} |\frac{Y_i - \hat{Y}_i}{Y_i}|$$
(15)

$$CORR = \frac{1}{N} \sum_{n=1}^{N} \frac{\sum_{i=1}^{\phi} (Y_{n,i} - \overline{Y}_n) (\hat{Y}_{n,i} - \overline{\hat{Y}}_n)}{\sqrt{\sum_{i=1}^{\phi} (Y_{n,i} - \overline{Y}_n)^2 (\hat{Y}_{n,i} - \overline{\hat{Y}}_n)^2}}$$
(16)

where ϕ represents the length of the predicted sequence, and \overline{Y}_n and $\overline{\hat{Y}}_n$ denote the mean values of the true and predicted values at node *n*, respectively. A smaller value of these metrics indicates higher prediction accuracy and better prediction performance.

VI. EXPERIMENTS

A. Performance Comparison

Table IV presents the results of our model and baselines across different datasets. Clearly, optimal results are achieved by our model across all five datasets. XGBoost, being a

TABLE IV: Performance comparison between PM-DMNet and the baselines on five traffic demand datasets. The best results are highlighted in bold, and the second-best results are underlined.

Mathad NYC-Bike16		Ν	NYC-Taxi16		N	NYC-Bike14		NYC-Bike15			NYC-Taxi15				
Method	RMSE	MAE	CORR	RMSE	MAE	CORR	RMSE	MAE	CORR	RMSE	MAE	CORR	RMSE	MAE	CORR
XGBoost	4.0494	2.4689	0.4107	21.1994	11.6806	0.4416	10.3137	4.8228	0.3322	8.1780	2.7175	0.1289	44.1421	14.8994	0.2195
DCRNN	3.2274	1.8973	0.6601	14.8318	8.4835	0.6671	6.3259	2.7483	0.5184	3.8320	1.2645	0.2844	16.6155	5.6424	0.4909
STGCN	3.7829	2.2076	0.5933	14.6473	7.8435	0.7257	8.5412	3.5833	0.4481	5.6169	1.6101	0.2529	28.1391	9.1844	0.3454
STG2Seq	3.7843	2.2055	0.5413	19.2077	10.4925	0.5389	10.8561	4.4999	0.3751	8.2462	2.3272	0.1855	39.4318	12.8251	0.3764
STSGCN	2.8846	1.7538	0.7126	10.9692	5.8299	0.8242	7.8272	3.2998	0.4656	5.4722	1.6086	0.2373	28.0221	8.9541	0.3695
MTGNN	2.7791	1.6595	0.7353	10.9472	5.9192	0.8249	6.3548	2.8172	0.5154	3.9407	1.2947	0.2640	18.1113	5.9255	0.5284
CCRNN	2.7674	1.7133	0.7333	9.8744	5.6636	0.8416	7.4890	3.5197	0.4861	4.4359	1.5249	0.2681	23.0052	8.5411	0.4049
GTS	2.9258	1.7798	0.6985	12.7511	7.2095	0.7348	6.7053	2.9446	0.5044	4.1698	1.3632	0.2654	17.8672	6.0408	0.4462
ESG	2.6727	1.6129	0.7449	8.9759	5.0344	0.8592	6.3503	2.7972	0.5175	3.8054	1.2293	0.2756	16.7635	5.5279	0.5247
MVFN	2.6981	1.6565	0.7380	8.7953	4.9433	0.5607	6.4116	2.8228	0.5131	3.9282	1.2928	0.2793	16.2687	5.5613	0.5296
MegaCRN	2.7480	1.6321	0.7425	8.7082	4.9082	0.8619	6.3258	2.8005	0.5185	3.9459	1.2681	0.2836	15.4985	5.2107	0.5398
PM-DMNet(P)	2.5631	1.5566	0.7709	8.4699	4.7682	0.8674	5.8790	2.5687	0.5274	3.5302	1.1678	0.2849	14.6360	4.8126	0.5509
PM-DMNet(R)	2.5964	1.5667	0.7638	8.4659	4.7635	0.8675	5.8656	2.5582	0.5246	3.7118	1.1947	0.2700	14.7843	4.8629	0.5429

19 2.9 9.5 6.4 4.2 18 2.8 9.0 4.0 6.0 ESC 6.0 KWSE 5.6 ESC RMSE 2.7 MTG 16 3.8 8.5 15 26 3.6 8.0 14 5.2 2.5 3.4 13 7.5 3.2 4.8 2.4 4 5 6 7 8 9 101112 4 5 6 7 8 9 101112 23 4 5 6 7 8 9 101112 2 4 5 6 7 8 9 101112 ż 4 5 6 7 8 9 101112 Horizon Horizon Horizon Horizon Horizon (a) RMSE on NYC-Bike16 (b) RMSE on NYC-Taxi16 (c) RMSE on NYC-Bike14 (d) RMSE on NYC-Bike15 (e) RMSE on NYC-Taxi15 6. PM-DMNet(F 5.4 PM-DMNet(P PM-DMNet(PM-DMNet(2.9 PM-DN 6.0 5.2 2.8 1.3 9 5.0 Mye 2.7 Ψ¥ 5.5 UAE MAE MAE MAE 4.8 2.5 1.2 5.0 2.4 4.6 2: 4.4 1 5 6 7 8 9 101112 4 5 6 7 8 9 101112 5 6 7 8 9 101112 5 6 7 8 9 101112 4 5 6 7 8 9 101112 2 3 2 3 2 34 2 Horizon Horizon Horizon Horizon Horizon (f) MAE on NYC-Bike16 (g) MAE on NYC-Taxi16 (h) MAE on NYC-Bike14 (i) MAE on NYC-Bike15 (j) MAE on NYC-Taxi15

Fig. 7: Prediction error at each horizon on five raffic demand datasets.

machine learning model, fails to capture the nonlinear relationships within traffic condition, resulting in its inferior performance. DCRNN, STGCN, and STG2Seq utilize predefined graph structures to capture spatio-temporal correlations within traffic condition, yielding satisfactory outcomes. However, due to the fixed weights in these predefined graph structures, the inability to capture dynamic correlations leaves significant room for improvement. MTGNN and GTS demonstrate commendable progress by learning graph structures adaptively from the data. Nevertheless, these adaptive graphs remain static and fail to capture the dynamic relationships between nodes. MegaCRN employs a meta-graph learner to construct dynamic graphs for extracting correlations between nodes. However, it does not consider the influence of temporal information on traffic patterns, which limits its performance. PM-DMNet excels by leveraging a dynamic memory network to dynamically extract features by identifying the most analogous traffic patterns based on historical data. Figure 7 illustrates the prediction errors of PM-DMNet compared to three baseline models across different prediction horizons. It is observed that, except for the initial three prediction steps, PM-DMNet consistently achieves lower prediction errors than the baseline models. Additionally, the error growth rate of PM-DMNet across all time horizons is slower than that of the baseline

models. Benefiting from the functionality of the evolving graph, ESG achieves comparable short-term prediction performance to PM-DMNet. However, as the prediction horizon expands, the error growth rate of ESG becomes significantly faster than that of PM-DMNet, resulting in an overall performance inferior to PM-DMNet. By leveraging temporal information corresponding to the prediction targets, PM-DMNet substantially reduces prediction uncertainty, thereby enhancing performance.

Table V presents the results of our model and baseline models on traffic flow/speed datasets. It is observed that, except for PEMSD8 where STWave slightly outperforms PM-DMNet (P) and is comparable to PM-DMNet (R), our model achieves the best performance across all datasets. Figure 8 shows the prediction errors of PM-DMNet and the two other best baseline models at different prediction horizons. From Figure 8, it is evident that the error gaps between models are more pronounced in the flow datasets compared to the speed datasets, indicating that predicting traffic speed is more challenging than predicting traffic flow. STWave utilizes the DWT algorithm to decompose traffic data into two separate low-frequency and high-frequency sequences, modeling them independently while considering the impact of temporal information, resulting in good performance on

Methods		PEMSD4	4		PEMSD	7		PEMSD	3	P	EMSD7(1	M)	Р	EMSD7(L)
methous	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
HA	59.24	38.03	27.88%	65.64	45.12	24.51%	59.24	34.86	27.88%	8.63	4.59	14.35%	9.03	4.84	14.90%
ARIMA	48.80	33.73	24.18%	59.27	38.17	19.46%	44.32	31.09	22.73%	13.20	7.27	15.38%	12.39	7.51	15.83%
VAR	38.61	24.54	17.24%	75.63	50.22	32.22%	29.81	19.19	13.10%	7.61	4.25	10.28%	8.09	4.45	11.62%
SVR	44.56	28.70	19.20%	50.22	32.49	14.26%	36.16	23.25	14.64%	7.47	4.09	10.03%	8.11	4.41	11.58%
LSTM	40.65	26.77	18.23%	45.94	29.98	13.20%	35.17	23.09	14.99%	7.51	4.16	10.10%	8.20	4.66	11.69%
TCN	37.26	23.22	15.59%	42.23	32.72	14.26%	35.79	22.72	14.03%	7.20	4.36	9.71%	7.29	4.05	10.43%
STGCN	34.89	21.16	13.83%	39.34	25.33	11.21%	27.09	17.50	11.29%	6.79	3.86	10.06%	6.83	3.89	10.09%
DCRNN	33.44	21.22	14.17%	38.61	25.22	11.82%	26.36	16.82	10.92%	7.18	3.83	9.81%	8.33	4.33	11.41%
GWN	39.66	24.89	17.29%	41.50	26.39	11.97%	30.05	18.28	12.15%	6.24	3.19	8.02%	7.09	3.75	9.41%
ASTGCN(r)	35.22	22.93	16.56%	37.87	24.01	10.73%	28.06	18.25	11.64%	6.18	3.14	8.12%	6.81	3.51	9.24%
LSGCN	33.86	21.53	13.18%	41.46	27.31	11.98%	26.76	17.73	11.20%	5.98	3.05	7.62%	6.55	3.49	8.77%
STSGCN	33.65	21.19	13.90%	39.03	24.26	10.21%	26.80	17.13	10.96%	5.93	3.01	7.55%	6.88	3.61	9.13%
AGCRN	32.26	19.83	12.97%	36.55	22.37	9.12%	25.22	15.95	10.09%	5.84	2.99	7.42%	6.04	3.13	7.75%
STFGNN	32.51	20.48	16.77%	36.60	23.46	9.21%	26.25	16.94	10.60%	5.74	2.93	7.28%	5.96	3.07	7.71%
STGODE	32.82	20.84	13.77%	37.54	22.59	10.14%	25.97	16.81	10.62%	5.66	2.97	7.36%	5.98	3.22	7.94%
STWave	30.39	18.50	12.43%	33.88	19.94	8.38%	23.40	13.42	8.90%	5.39	2.66	6.76%	5.87	2.88	7.25%
MegaCRN	31.03	19.07	12.71	33.83	20.42	8.68%	24.15	15.19	9.88%	5.40	2.67	6.73%	5.84	2.88	7.19%
PM-DMNet(P)	30.36	18.34	12.05%	33.33	19.35	8.05%	23.35	13.55	9.04%	5.33	2.61	6.55%	5.79	2.81	7.13%
PM-DMNet(R)	30.68	18.37	12.01%	33.15	19.18	7.95%	23.22	13.40	8.87%	5.36	2.60	6.57%	5.81	2.79	6.99%

TABLE V: Performance comparison between PM-DMNet and the baselines on five traffic flow/speed datasets. The best results are highlighted in bold, and the second-best results are underlined.



Fig. 8: Prediction error at each horizon on five flow/speed datasets.

traffic flow datasets. However, on speed datasets, due to the inherent differences between traffic speed and traffic flow, the DWT algorithm struggles to decompose useful high and lowfrequency sequences, causing STWave's performance to be on par with MegaCRN. PM-DMNet does not rely on sequence decomposition for modeling, thus avoiding the difficulties associated with ineffective decomposition, leading to excellent performance on both flow and speed datasets.

B. Computation Cost

To compare and demonstrate the computational efficiency of our model, we evaluate the training time, inference time, and GPU cost of selected models. The batch size for all models is set to 32. Table VI shows the computational costs of PM-DMNet compared to baseline models. As observed in Table VI, the training and inference times of PM-DMNet(P) are significantly lower than those of other baselines, and it also outperforms PM-DMNet(R), demonstrating the advantages of the dynamic memory network and PMP in terms of computa-

TABLE VI: The computation cost on four datasets.

Dataset	Model	Tainning Time (s/epoch)	Inference Time (s)	GPU Cost (GB)
	PM-DMNet(P)	4.17	0.29	1.44
NVC Dilea16	PM-DMNet(R)	7.26	0.47	1.46
IN I C-DIKETO	ESG	20.83	1.65	15.60
	MegaCRN	7.04	0.68	2.00
	PM-DMNet(P)	4.43	0.29	1.50
NYC-Taxi16	PM-DMNet(R)	7.53	0.46	1.50
	ESG	22.93	1.91	16.50
	MegaCRN	6.60	0.66	2.23
	PM-DMNet(P)	14.87	1.53	1.73
DEMCD4	PM-DMNet(R)	25.21	2.35	1.70
PEMSD4	STWave	56.32	7.46	5.79
	MegaCRN	24.60	3.71	2.44
	PM-DMNet(P)	33.77	4.08	4.79
DEMOD7	PM-DMNet(R)	41.68	4.09	4.75
PEMSD/	STWave	272.95	36.53	16.76
	MegaCRN	104.12	16.91	7.57

tional speed and memory usage. Despite ESG's strong performance, its high GPU cost and relatively slow processing speed present challenges in deployment. Although STWave employs a novel graph attention mechanism to optimize modeling speed, its complex network structure still demands substantial GPU resources and long training times. MegaCRN uses RMP methods while adopting a simple adaptive graph convolution method to extract spatial correlations between nodes, resulting in lower training time and GPU cost. Therefore, on datasets with fewer nodes, MegaCRN's training time is comparable to that of PM-DMNet(P), which also uses RMP methods. However, on large-scale node datasets, the $O(N^2)$ complexity of GCN still requires higher training time and GPU cost. In contrast, the dynamic memory network used by PM-DMNet(P) has a time complexity of O(N), significantly lower than the $O(N^2)$ complexity of graph convolution networks (GCNs). Consequently, on PEMSD7, PM-DMNet(P) exhibits faster training speed and lower GPU cost than MegaCRN, showcasing the computational speed advantages of our model.

C. Complexity Analysis

The computation complexity for feature aggregation in GCN is $O(N^2)$, and both the computation of attention matrices and feature aggregation in attention mechanisms are also $O(N^2)$. For DMN, the computation complexity for calculating similarity weights and feature aggregation is O(NM), where M is a constant value significantly smaller than N. When M is much smaller than N, the time complexity of DMN can be considered as O(N). Therefore, compared to GCN and attention mechanisms, DMN exhibits notable advantages in terms of time and memory complexity.

D. Recurrent Multi-step Prediction vs. Parallel Multi-step Prediction

In this subsection, the performance of PMP and RMP prediction methods is compared. Tables IV and V present the results of PM-DMNet(P) and PM-DMNet(R) on traffic demand and traffic flow/speed datasets, respectively. As shown in Table IV, PM-DMNet(P) outperforms PM-DMNet(R) on three datasets for traffic demand prediction tasks and matches PM-DMNet(R) on two datasets, indicating that PM-DMNet(P) has certain advantages over PM-DMNet(R) in traffic demand prediction tasks. This is further evidenced by the per-step prediction errors shown in Figure 7.

However, as seen in Table V, PM-DMNet(R) exhibits a performance advantage over PM-DMNet(P) in traffic flow/speed tasks. In Figure 8, it is shown that PM-DMNet(P) has significantly larger prediction errors in the initial time steps compared to PM-DMNet(R), but the errors of both methods are comparable in the later time steps. This phenomenon might be attributed to the different time intervals of the datasets. The traffic demand datasets are collected at 30-minute intervals, resulting in more pronounced differences between historical data and prediction targets, where the PMP method performs better than the RMP method. In contrast, traffic flow/speed datasets are collected at 5-minute intervals, creating more continuity between historical data and prediction targets, thereby giving the RMP method an edge over the PMP method in performance.

VII. ABLATION STUDY

A. Effectiveness Analysis of model components

In this section, ablation experiments are conducted on the key components of PM-DMNet to validate their effectiveness. To investigate the impact of different modules, the following variants are designed:

W/O Decoder: This variant removes the decoder component and predicts using an MLP layer directly applied to the encoder's output. Since the decoding process is omitted, this variant is identical for both PM-DMNet(P) and PM-DMNet(R).

W/O TAM: In this variant, the Transfer Attention Module (TAM) is excluded. Instead, the prediction is made using the output H_n from the encoder, replacing the output H_{n+1} from the transfer attention mechanism.

W/O DMN: This variant substitutes the Dynamic Memory Network (DMN) module with an MLP layer for making predictions.

W/O NAPL: This variant removes the Node Adaptive Parameter learning (NAPL) module and uses a linear layer instead.

TABLE VII: Ablation experiments for each module.

datacet	variante	Р	M-DMNet	(P)	PM-DMNet(R)			
uataset	variants	RMSE	MAE	CORR	RMSE	MAE	CORR	
	PM-DMNet	2.5631	1.5566	0.7709	2.5964	1.5667	0.7638	
	W/O Decoder	2.6308	1.5949	0.7602	2.6308	1.5949	0.7602	
NYC-Bike16	W/O TAM	2.6341	1.5859	0.7599	/	/	/	
	W/O DMN	3.1756	1.8078	0.6728	3.9676	2.2438	0.4815	
	W/O NAPL	3.1800	1.8057	0.6726	3.2525	18.265	0.6689	
dataaat	maionto	PM-DMNet(P)			PM-DMNet(R)			
uataset	variants	RMSE	MAE	MAPE	RMSE	MAE	MAPE	
	PM-DMNet	30.36	18.34	12.05%	30.68	18.37	12.01%	
	W/O Decoder	33.31	20.15	13.28%	33.31	20.15	13.28%	
PEMSD4	W/O TAM	30.75	18.42	12.10%	/	/	/	
	W/O DMN	35.03	21.40	14.29%	39.74	25.32	17.22%	
	W/O NAPL	34.84	21.19	14.31%	34.98	21.3	14.29%	

Table VII presents the performance of PM-DMNet(P) and PM-DMNet(R) alongside their variants. It is evident from the table that PM-DMNet(P) and PM-DMNet(R) outperform all other variants, demonstrating the effectiveness of each component.

For the **W/O Decoder** variant, the pattern matching process is omitted during the decoding stage, and predictions are made directly using an MLP layer. As a result, this variant can only utilize historical data information and lacks the ability to leverage the time point information of the prediction target. Consequently, its performance is inferior to PM-DMNet(P) and PM-DMNet(R).

The performance of the **W/O TAM** variant also falls short of PM-DMNet(P). This indicates that the discrepancy between historical data and the prediction target leads to a performance decline, validating our proposed solution. This shows that using a suitable method for parallel prediction can achieve results comparable to or better than serial prediction.

The **W/O DMN** variant's performance is significantly inferior to both PM-DMNet models, highlighting the feasibility of our approach to use a memory network to match and extract the most representative traffic patterns.

Similarly, the performance of the W/O NAPL variant is lower than that of the two PM-DMNet models, underscoring



Fig. 9: Ablation experiment of time embedding.

the necessity for the model to learn the unique traffic patterns of each node.

B. Effectiveness Analysis of GCN and DMN

To validate the differences in performance and computational cost between GCN and DMN, a variant named DGCNet is designed. This variant uses dynamic graph convolution instead of DMN. The formula for dynamic graph convolution is expressed as follows:

$$E_t^d = F_t \odot T_t \tag{17}$$

$$A_t^d = ReLU(E_t^d E_t^{d^T}) \tag{18}$$

$$H_t = (I_N + D^{-\frac{1}{2}} A_t^d D^{-\frac{1}{2}}) X \Theta$$
(19)

where $A_t^d \in \mathbb{R}^{N \times N}$ represents the dynamic graph at time point t, D is the degree matrix of A_t^d , and I_N represents the identity matrix. Similar to PM-DMNet, DGCNet can be divided into two variants based on the prediction method: DGCNet(P) and DGCNet(R).

TABLE VIII: Ablation experiment of GCN and DMN

Dataset	model	RMSE	MAE	MPAE	Train Time (s/epoch)	Inference Time (s)	GPU Cost (GB)
PEMSD7 (16)	DGCNet(P) PM-DMNet(P) DGCNet(R) PM-DMNet(R)	33.81 23.35 33.38 23.22	19.60 13.55 19.39 13.40	8.28% 9.04% 8.06% 8.87%	231.10 49.16 237.98 81.43	23.62 5.68 24.00 7.67	13.04 2.96 12.58 3.45
PEMSD8 (64)	DGCNet(P) PM-DMNet(P) DGCNet(R) PM-DMNet(R)	23.99 33.33 23.55 33.15	13.95 19.35 13.70 19.18	9.29% 8.05% 8.92% 7.95%	7.99 7.82 13.56 13.48	1.00 0.84 1.32 1.26	3.20 1.64 2.87 1.49

Experiments are conducted on PEMSD7 and PEMSD8, with a batch size of 16 for PEMSD7 and 64 for PEMSD8. Table VIII presents the results of GCN and DMN on these two datasets. It can be observed that PM-DMNet outperforms DGCNet, indicating that DMN can achieve excellent performance without relying on GCN. Additionally, while PM-DMNet's computational metrics are slightly better than DGCNet on the smaller PEMSD8 dataset, the difference is

not significant. However, on the larger PEMSD7 dataset, PM-DMNet's computational metrics are significantly superior to those of DGCNet, demonstrating the advantage of DMN's O(N) complexity over GCN's $O(N^2)$ complexity in large-scale node scenarios.

C. Effectiveness Analysis of time embedding

To validate the impact of intra-daily time features and weekly time features on the model, two variants are designed for this subsection:

use day: The dynamic memory network is updated using only intra-daily time feature embeddings in this variant.

use week: The dynamic memory network is updated using only weekly time feature embeddings in this variant.

Experiments are conducted on four datasets to observe the influence of time information on model performance across different types of data.

Figure 9 presents the performance of PM-DMNet(P) and PM-DMNet(R) along with their variants. It can be observed that when only one type of time feature embedding is used, the model's performance generally decreases. Except for the NYC-Taxi16 dataset, where **use week** outperforms **use day** in PM-DMNet(P), the performance of **use day** is superior to **use week** in all other cases. This indicates that intra-daily information typically has a greater impact on model performance than weekly information. Additionally, in the PEMSD7(M) dataset, the performance of **use day** is comparable to that of their original models, while the performance of **use week** varies significantly. This suggests that, unlike other types of data, traffic speed data shows less pronounced differences between weekdays and weekends, exhibiting high similarity.

VIII. HYPER-PARAMETER ANALYSIS

To validate the impact of hyperparameters on model performance, hyperparameter experiments are conducted on the PEMSD8 dataset. Specifically, in this section, we investigate the effects of the temporal embedding dimension p, the dimension d of the node embedding matrix E in the node adaptive module, and the dimension M of the memory network matrix. In these experiments, other parameters are kept constant while only the parameter under study is changed.



(a) Sensitivity of Parameter p to PM- (b) Sensitivity of Parameter p to PM DMNet(P) DMNet(R)

Fig. 10: Sensitivity analysis of parameter p on PEMSD8.

A. Sensitivity to p

The parameter p is set to {5, 10, 15, 20, 25, 30} to evaluate its sensitivity on model performance. In Figure 10, the performance of the model under different values of p is shown. It can be seen that the model performs relatively stable when p is between 10 and 25. Additionally, across various settings, PM-DMNet(R) consistently exhibits lower errors compared to PM-DMNet(P).



(a) Sensitivity of Parameter *d* to PM-(b) Sensitivity of Parameter *d* to PM-DMNet(P) DMNet(R)

Fig. 11: Sensitivity analysis of parameter d on PEMSD8.

B. Sensitivity to d

The parameter d is set to $\{2, 5, 10, 20\}$ to evaluate its sensitivity to model performance. The performance of the model with different values of d is shown in Figure 11. It is observed that d does not significantly affect model performance; however, it greatly impacts the training speed. When d is set between 5 and 10, the model trains quickly while maintaining performance. Therefore, d is recommended to be set around 5 to 10.

C. Sensitivity to M

The parameter M is set to $\{5, 10, 15, 20\}$ to evaluate its sensitivity to model performance. The performance of the model with different values of M is shown in Figure 12.



(a) Sensitivity of Parameter M to PM-(b) Sensitivity of Parameter M to PM-DMNet(P) DMNet(R)

Fig. 12: Sensitivity analysis of parameter M on PEMSD8.

It is observed that the model achieves stable and excellent performance when M is between 5 and 20. Therefore, M is set to 10.

IX. VISUALIZATION

To explore whether the Node Adaptive Parameter module captures the unique traffic patterns of each node, we utilize T-SNE [45] to visualize the node embedding matrix E used in the module trained on NYC-Taxi16 dataset.



Fig. 13: Visualization of node embeddings E. The nodes in the red-bordered area are nodes 215 and 222, the node in the blue-bordered area is node 26.

Figure 13 illustrates the visualization results of the node embeddings E. From the figure, it can be observed that certain nodes exhibit a clustering phenomenon, while a few nodes overlap, indicating high similarity in traffic patterns among them. Moreover, there are nodes that are far apart, suggesting significant differences in their traffic patterns.

To further verify the high similarity in traffic patterns among nearby nodes and the differences in traffic patterns among distant nodes, we select adjacent nodes within the redbordered area, specifically Node 215 and Node 222, as well as a distant node within the blue-bordered area, Node 26, for visualization of their traffic demand data. Figures 14(a) and 14(b) respectively illustrate the trend changes in the 'Pick-up' and 'Drop-off' features of the traffic demand for these three nodes. It is evident that the trends for Node 215 and Node 222 are highly similar, indicating a strong correlation between them. Meanwhile, the trend for Node 26 is notably different from the other two nodes, suggesting a significant difference



Fig. 14: Visualization of real traffic demand on NYC-Taxi16.

in their traffic patterns. The visualization results above confirm that the Node Adaptive Parameter module can learn the traffic patterns of individual nodes effectively.

X. CONCLUSION

This paper proposes a novel traffic prediction model, PM-DMNet. PM-DMNet employs a new dynamic memory network module that learns the most representative traffic patterns into a memory network matrix. During prediction, the model extracts pattern features by matching the current traffic pattern with the memory network matrix. Additionally, PM-DMNet supports both parallel and sequential Multi-step prediction methods to meet different needs. To further enhance the accuracy of parallel Multi-step prediction, a transfer attention mechanism is introduced to mitigate the disparity between historical data and prediction targets. Extensive experiments validate the effectiveness of PM-DMNet. In future work, further methods for extracting features from patterns are planned to be explored.

REFERENCES

- G. E. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Journal* of the American statistical Association, vol. 65, no. 332, pp. 1509–1526, 1970.
- [2] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," *IEEE transactions on intelligent transportation* systems, vol. 5, no. 4, pp. 276–281, 2004.
- [3] J. Guo, W. Huang, and B. M. Williams, "Adaptive kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 50–64, 2014.
- [4] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proceedings of the AAAI* conference on artificial intelligence, vol. 31, no. 1, 2017.
- [5] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," *Advances in neural information processing systems*, vol. 33, pp. 17804–17815, 2020.

- [6] F. Li, J. Feng, H. Yan, G. Jin, F. Yang, F. Sun, D. Jin, and Y. Li, "Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution," ACM Transactions on Knowledge Discovery from Data, vol. 17, no. 1, pp. 1–21, 2023.
- [7] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations*, 2018.
- [8] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 1907– 1913.
- [9] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," in *Proceedings* of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 3634–3640.
- [10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2016.
- [11] W. Weng, J. Fan, H. Wu, Y. Hu, H. Tian, F. Zhu, and J. Wu, "A decomposition dynamic graph convolutional recurrent network for traffic forecasting," *Pattern Recognition*, vol. 142, p. 109670, 2023.
- [12] J. Jiang, C. Han, W. X. Zhao, and J. Wang, "Pdformer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction," in AAAI. AAAI Press, 2023.
- [13] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," in *Proceedings of the AAAI conference* on artificial intelligence, vol. 34, no. 01, 2020, pp. 1234–1241.
- [14] S. Guo, Y. Lin, L. Gong, C. Wang, Z. Zhou, Z. Shen, Y. Huang, and H. Wan, "Self-supervised spatial-temporal bottleneck attentive network for efficient long-term traffic forecasting," in 2023 IEEE 39th International Conference on Data Engineering (ICDE). IEEE, 2023, pp. 1585–1596.
- [15] X. Kong, K. Wang, M. Hou, F. Xia, G. Karmakar, and J. Li, "Exploring human mobility for multi-pattern passenger prediction: A graph learning framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 16148–16160, 2022.
- [16] X. Kong, Z. Shen, K. Wang, G. Shen, and Y. Fu, "Exploring bus stop mobility pattern: a multi-pattern deep learning prediction framework," *IEEE Transactions on Intelligent Transportation Systems*, 2024.
- [17] Y. Fang, Y. Qin, H. Luo, F. Zhao, B. Xu, L. Zeng, and C. Wang, "When spatio-temporal meet wavelets: Disentangled traffic forecasting via efficient spectral graph attention networks," in 2023 IEEE 39th International Conference on Data Engineering (ICDE). IEEE, 2023, pp. 517–529.
- [18] D. Zhang and J. Li, "Multi-view fusion neural network for traffic demand prediction," *Information Sciences*, vol. 646, p. 119303, 2023.
- [19] J. Ye, Z. Liu, B. Du, L. Sun, W. Li, Y. Fu, and H. Xiong, "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*, 2022, pp. 2296–2306.
- [20] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 01, 2020, pp. 914–921.
- [21] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "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*, 2020, pp. 753–763.
- [22] J. Ye, L. Sun, B. Du, Y. Fu, and H. Xiong, "Coupled layer-wise graph convolution for transportation demand prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 5, 2021, pp. 4617–4625.
- [23] J. Weston, S. Chopra, and A. Bordes, "Memory networks," in 3rd International Conference on Learning Representations, ICLR 2015, 2015.
- [24] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," *Advances in neural information processing systems*, vol. 28, 2015.
- [25] L. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to remember rare events," in *International Conference on Learning Representations*, 2016.
- [26] A. Madotto, C.-S. Wu, and P. Fung, "Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 1468– 1478.

- [27] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. v. d. Hengel, "Memorizing normality to detect anomaly: Memoryaugmented deep autoencoder for unsupervised anomaly detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1705–1714.
- [28] H. Lv, C. Chen, Z. Cui, C. Xu, Y. Li, and J. Yang, "Learning normal dynamics in videos with meta prototype network," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 15425–15434.
- [29] Y.-Y. Chang, F.-Y. Sun, Y.-H. Wu, and S.-D. Lin, "A memory-network based solution for multivariate time-series forecasting," *arXiv preprint* arXiv:1809.02105, 2018.
- [30] H. Lee, S. Jin, H. Chu, H. Lim, and S. Ko, "Learning to remember patterns: Pattern matching memory networks for traffic forecasting." International Conference on Learning Representations, 2022.
- [31] R. Jiang, Z. Wang, J. Yong, P. Jeph, Q. Chen, Y. Kobayashi, X. Song, S. Fukushima, and T. Suzumura, "Spatio-temporal meta-graph learning for traffic forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 7, 2023, pp. 8078–8086.
- [32] S. Lin, W. Lin, W. Wu, F. Zhao, R. Mo, and H. Zhang, "Segrnn: Segment recurrent neural network for long-term time series forecasting," *arXiv* preprint arXiv:2308.11200, 2023.
- [33] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li, "Revisiting spatialtemporal similarity: A deep learning framework for traffic prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 5668–5675.
- [34] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatialtemporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 922–929.
- [35] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," *Advances in neural information processing systems*, vol. 28, 2015.
- [36] J. D. Hamilton, Time series analysis. Princeton university press, 2020.
- [37] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results," *Journal of transportation engineering*, vol. 129, no. 6, pp. 664– 672, 2003.
- [38] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.
- [39] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing* systems, vol. 27, 2014.
- [40] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv* preprint arXiv:1803.01271, 2018.
- [41] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Q. Z. Sheng, "Stg2seq: spatial-temporal graph to sequence model for multi-step passenger demand forecasting," in 28th International Joint Conference on Artificial Intelligence, IJCAI 2019. International Joint Conferences on Artificial Intelligence, 2019, pp. 1981–1987.
- [42] R. Huang, C. Huang, Y. Liu, G. Dai, and W. Kong, "Lsgcn: Long shortterm traffic prediction with graph convolutional networks." in *IJCAI*, vol. 7, 2020, pp. 2355–2361.
- [43] Z. Fang, Q. Long, G. Song, and K. Xie, "Spatial-temporal graph ode networks for traffic flow forecasting," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 364–373.
- [44] C. Shang and J. Chen, "Discrete graph structure learning for forecasting multiple time series," in *Proceedings of International Conference on Learning Representations*, 2021.
- [45] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." Journal of machine learning research, vol. 9, no. 11, 2008.





Wenchao Weng received his Bachelor's degree in Information and Computing Science from Zhejiang Wanli University in 2019 and his Master's degree in Computer Technology from Hangzhou Dianzi University in 2024. He is currently pursuing a Ph.D. in Computer Science and Technology at Zhejiang University of Technology. His research interests include data mining, spatio-temporal graph neural networks, and traffic forecasting.

Mei Wu received the Bachelor's degree from Shandong University in China in 2022 and is currently pursuing a Master's degree in Computer Science at Hangzhou Dianzi University. Her main research interests focus on spatiotemporal graph data mining and intelligent transportation systems.



Hanyu Jiang is currently pursuing a Bachelor's degree at Hangzhou Dianzi University. His primary research focuses on the combination of bioinformatics and deep learning, specifically in the areas of multimodal and deep generative models.



Wanzeng Kong (Senior Member, IEEE) received the Ph.D. degree from the Department of Electrical Engineering, Zhejiang University, in 2008. He was a Visiting Research Associate with the Department of Biomedical Engineering, University of Minnesota Twin Cities, Minneapolis, MN, USA, from 2012 to 2013. He is currently a Full Professor and the Director of the Cognitive Computing and BCI Laboratory, School of Computer Science and Technology, Hangzhou Dianzi University. His current research interests include machine learning, pattern recognition,

and cognitive computing.



Xiangjie Kong (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 2004 and 2009, respectively. He is a Professor with College of Computer Science and Technology, Zhejiang University of Technology, China. Previously, he was an Associate Professor with the School of Software, Dalian University of Technology, China. He has published over 200 scientific papers in international journals and conferences (with over 180 indexed by ISI SCIE). His research interests include urban computing, mobile comput-

ing, and computational social science. He is a Senior Member of the IEEE, a Distinguished Member of CCF, and is a member of ACM.



Feng Xia (Senior Member, IEEE) received the BSc and PhD degrees from Zhejiang University, Hangzhou, China. He is a Professor in School of Computing Technologies, RMIT University, Australia. Dr. Xia has published over 300 scientific papers in journals and conferences (such as IEEE TAI, TKDE, TNNLS, TC, TMC, TBD, TCSS, TNSE, TETCI, TETC, THMS, TVT, TITS, TASE, ACM TKDD, TIST, TWEB, TOMM, WWW, AAAI, ICLR, SIGIR, WSDM, CIKM, JCDL, EMNLP, and INFOCOM). His research interests include artificial

intelligence, graph learning, brain science, digital health, and robotics. He is a Senior Member of IEEE and ACM, and an ACM Distinguished Speaker.