DFDG:Technical Report

Muhammad Usman*
School of Science and Engineering
University of Missouri-Kansas City
Missouri, USA
*mucbp@umkc.edu

***Abstract*—DFDG is introduced as a pioneering fusion of federated graph neural networks (GNNs) with dynamic graphs and unsupervised learning for dynamic traffic forecasting in urban settings. Utilizing self-organizing map clustering, speed patterns are clustered at various time intervals, and resulting clusters are assigned to Federated Learning participants. Within the federated setup, participants use neural ordinary differential equations coupled with attention spatial-temporal graph convolution networks to collectively form the DFDG architecture. During training, participants contribute local updates to a global GNN model, with performance weightage reflecting their reliability. This dynamic weighting system influences each participant's significance in subsequent communication rounds, promoting adaptive learning across the federated network. We group homogeneous participants based on their previous round performance and data distribution. Grouping contributes to the overall efficiency and effectiveness of federated learning systems, enhancing model generalization, reducing communication overhead, preserving privacy by aggregating updates from homogeneous client groups, and allowing adaptability to local data characteristics. Extensive benchmark testing validates DFDG's superiority in traffic forecasting, surpassing current state-of-the-art methods. Our method not only advances traffic forecasting but can be adapted for broader discourse on real-time dynamic graph applications.**

I. Methods and Materials

A. Definition and Formulation of Traffic Forecasting Problem

The objective of traffic forecasting is to predict future traffic speeds for a specific horizon, based on previously observed traffic flow from a network of correlated sensors. This network can be represented as a weighted directed graph, $G = (E, V, A)$, with V being a set of N nodes, representing the locations of the sensors on the road network. These sensors are responsible for recording traffic information at their locations. E is a set of edges representing the connections or relationships between the nodes, and A is a weighted adjacency matrix reflecting the proximity or distance between nodes.

The observed traffic flow on G_t is denoted as a graph signal $X_t = \{\mathrm{x}_t^1, \mathrm{x}_t^2, ..., \mathrm{x}_t^N\} \in \mathbb{R}^{N \times F}$, where F denotes the number of features(velocity and volume) on each node at time t. The aim is to learn a function $DFDG(\cdot)$ with parameters (Θ_{Global}), given historical α steps of observations, and to infer the next β horizon. It can be expressed as:

$$[X(t-(\alpha-1), \ldots, X(t)] \xrightarrow[\Theta_{\text{Global}}]{\text{DFDG}(\cdot)} [X(t+1), \ldots, X(t+\beta)]$$

B. Federated GNN learning framework

In this section, we explain how the DFDG algorithm is constructed. It is also diagrammatically shown in Figure 1

1) Normalization: For effective and trustworthy machine learning models, aligning datasets to the same distribution is crucial. To ensure consistency in traffic speed data. We first perform max normalization,

$$X' \leftarrow \frac{X}{X_{\max}}$$

and then apply Z-score standardization,

$$Z \leftarrow \frac{X' - X'_{\mu}}{X'_{\sigma}}$$

which scales the data to a range of 0μ and 1σ.

2) Graph Level Embedding(GLE): The traffic speed data we possess is represented in the form of spatial-temporal graphs, which are captured and aggregated at 5-minute intervals at time t. We transform these graphs into GLEs using A3TGCN [1] and global mean pooling. A3TGCN, comprising Graph Recurrent Units (GRUs) and Graph Convolutional Networks (GCNs), captures spatial and temporal information, respectively, and generates a vector of node features. Subsequently, global mean pooling computes the mean of each feature across all nodes, resulting in a vector that represents the GLE of the entire graph.

3) GLE Dimensionality Reduction: We further reduce the GLE dimensions to three dimensions by employing the t-SNE algorithm. It's effective at revealing intricate relationships within complex datasets. t-SNE handles and preserves complex, non-linear relationships, which is crucial for preserving the local structure of historical traffic speed data. It ensures that data points close together in the high-dimensional space remain close in the low-dimensional space which is useful for traffic speed patterns where local similarities are important. Let $X \leftarrow \{x_1, x_2, ..., x_n\}$ be the high-dimensional GLE_s in $\mathbb{R}^D$, and P be the pairwise similarity matrix defined on X. t-SNE aims to find a low-dimensional representation $Y \leftarrow \{y_1, y_2, ..., y_n\}$ in $\mathbb{R}^d$, where $d \ll D$, such that the pairwise similarities in Y are represented by a Student's t-distribution. The probability P_{ij} that x_i and x_j are neighbors in the high-dimensional space is computed as a conditional probability normalized over all pairs

$$P_{ij} \leftarrow \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq l} \exp\left(-\|x_k - x_l\|^2 / 2\sigma_k^2\right)}$$

Here, σ_i is the variance of the Gaussian distribution centered at x_i, Similarly,

$$Q_{ij} \leftarrow \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1+\|y_k - y_l\|^2)^{-1}}$$

is defined for the low-dimensional space, and the KL-divergence $\sum_{i \neq j} P_{ij} \log\left(\frac{P_{ij}}{Q_{ij}}\right)$ between P and Q is minimized through gradient descent optimization.

4) Clustering Similar Traffic Speed Patterns: The Self-Organizing Map (SOM) is an unsupervised, competitive learning neural network algorithm adept at discovering similar speed patterns in traffic data without the need for labeled information. This makes it particularly suitable for scenarios where labeling can be challenging. The SOM maintains topological relationships, which means that neighboring nodes on the map, or the arrangement of neurons or nodes in a two-dimensional grid, represent similar speed patterns. This spatial organization of nodes allows the map to reflect the similarity relationships in the input data. In the realm of traffic speed patterns, the SOM map simplifies the interpretation and analysis of spatial-temporal relationships by representing different speed pattern clusters. This proves useful in identifying regions with consistent traffic behaviors, such as roads or intersections that consistently exhibit similar traffic speeds. The SOM's approach is robust to noisy data and can update the model as new data becomes available, making it beneficial in dynamic traffic environments. SOMs are advantageous for clustering traffic speed patterns due to their spatial organization, dimensionality reduction, topology preservation, and adaptability to temporal changes. They arrange clusters to reflect the geographic proximity of similar speed patterns as shown in Fig 3, thereby facilitating the visualization and interpretation of high-dimensional traffic data. The 2D grid of a SOM provides a visual representation of clusters, aiding in interpretation and anomaly detection. Known for their flexibility and robustness, SOMs capture complex relationships and are robust to noisy data, making them suitable for dynamic and unpredictable traffic contexts. Lastly, SOMs provide insight into both local and global patterns, which is valuable for gaining insights into different aspects of the traffic data.

The algorithm can be formally expressed in mathematical terms as:

1) Initialize a 2D grid of neurons with weight vectors W_{ij}, where i and j are the indices of the neurons.
2) For each graph level embedding X in Embeddings:
   a) Compute the Euclidean distance between X and each weight vector W_{ij}:
   $$D_{ij} \leftarrow \sqrt{\sum_k (X_k - W_{ijk})^2},$$
   where k is the index.
   b) Find the neuron with the smallest distance. The Best Matching Unit (BMU) or the winning neuron, is the neuron whose weight vector is most similar to the input vector being presented to the network:
   $$(i_{\text{BMU}}, j_{\text{BMU}}) \leftarrow \text{argmin}_{ij} D_{ij}.$$
3) Define a neighborhood function $\lambda_{ij}(t)$ that determines the influence of the winning neuron on its neighbors. It often takes the form of a Gaussian function:
   $$\lambda_{ij}(t) \leftarrow \exp\left(-\frac{(d_{ij}^2)}{2\sigma(t)^2}\right),$$
   where d_{ij} is the distance between neurons (i, j) and $(i_{\text{BMU}}, j_{\text{BMU}})$, and $\sigma(t)$ is the neighborhood size at time t.
4) Update the weights of the neurons based on their distance from the winning neuron:
   $$W_{ij}(t+1) \leftarrow W_{ij}(t) + \alpha(t) \cdot \lambda_{ij}(t) \cdot (X - W_{ij}(t)),$$
   where $\alpha(t)$ is the learning rate at time t.
5) Repeat steps 2-4 for multiple iterations
6) Post training, the SOM maps similar traffic speed patterns to neighboring neurons to form clusters.

5) Grouping Homogeneous Participants: After clustering similar traffic speed patterns, we allocate each one to a federated learning participant. In each communication round, we group participants with similar characteristics based on their previous performance and data distribution. This creates a feature vector consisting of mean square error, singular value decomposition, standard deviation, variance, mean, range, skewness, and kurtosis, all of which are computed from each participant's dataset. Furthermore, we use the elbow method to determine the optimal number of K groups and then train a K-means algorithm on these extracted feature vectors. The K-means algorithm assigns a label to each feature vector, which also indicates the group to which each participant belongs.

6) Federated GNNs Learning: Federated GNN learning, a privacy-preserving machine learning approach, trains GNN models across decentralized participants and enables global model training without the need for raw data exchange. Initially, the global model parameter vector θ initializes itself with random values and distributes them to all the participants. Each participant, k, performs local training using its own private traffic speed dataset, DP_k, to train our custom-built GNN architecture, also known as LocalDFDG, shown in Fig. 2 and will be explained in the subsequent section. Each participant optimizes their local parameters, θ_k, with a local Adam optimizer to minimize the local Minkowski ($P = 1.5$) loss function, $L(\theta_k, DP_k)$. After local training, each participant computes its weightage, W_k, and the difference between its local model and the global model, $\Delta\theta_k = \theta_k - \theta$. To compute W_k, we first calculate the error of the participant model over the training set by

$$\epsilon(y, \hat{y}) \leftarrow \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \quad (1)$$

Then compute the W_k using the ϵ

$$\alpha \leftarrow \frac{1}{2} \cdot \ln\left(\frac{1-\epsilon}{\epsilon}\right) \tag{2}$$

$$W_k \leftarrow \frac{1}{n} \cdot \exp\left(-\alpha \cdot y_i \cdot \hat{y}_i\right) \tag{3}$$

The participants send model updates, $\Delta\theta_k$, and W_k to a central aggregator. The aggregator then multiplies W_k with the updates and aggregates them using the formula $\Delta\theta_{\text{global}} \leftarrow \frac{1}{N}\sum_{k=1}^{N}(\Delta\theta_k \cdot W_k)$ to compute a global update for the global model parameters. Finally, the aggregator updates its global model parameters with the global model update using the formula $\theta \leftarrow \theta + \Delta\theta_{\text{global}}$.

7) The Participant's LocalDFDG Model Architecture: Each participant trains it's LocalDFDG model, the topology is shown in Figure 2. The A3TGCN [1] module captures the Spatial and Temporal information via its GCN and GRU units and produces a context vector C. We further consider a continuous-time(depth) model also known as Neural ODE [2]

$$C(t) = C(0) + \int_0^t f(C(t), \theta)dt$$

Here, $C(t)$ is the system state at time t, $C(0)$ is the initial state, and $f(C(t), \theta)$ is a function parameterized by a neural network that models the system's hidden dynamics. The model can be backpropagated through an Ordinary Differential Equation (ODE) solver without any internal operations [2], allowing it to be treated as a single block within a larger neural network. This approach integrates continuous-time dynamics into a discrete-time neural network, providing a powerful tool for modeling complex systems.

C. Algorithm:

Algorithm 1 shows the construction of DFDF, The entire working functionalities are incorporated in it.

Algorithm 1 The Construction of DFDG

Require: D: Spatial-Temporal Graph Dataset
Ensure: $\Theta_{\text{global}} \rightarrow$ Global GNN Model

$D \leftarrow \{G_{t_1}, G_{t_2}, ..., G_{t_N}\}$
for $G_{t_i} \in D$ **do**
  $G'_{t_i} \leftarrow \frac{G_{t_i}}{\max(D)}$
  $G''_{t_i} \leftarrow \frac{G'_{t_i} - \mu G'_{t_i}}{\sigma G'_{t_i}}$,
  $E_{t_i} \leftarrow \text{A3TGCN}(G''_{t_i})$
end for
$E_{mb} \in \{E_{t_1}, E_{t_2}, ..., E_{t_N}\}$
$E'_{t_i} \leftarrow \text{t-SNE}(E_{mb})$
for $E'_{t_i} \in \{E'_{t_1}, E'_{t_2}, ..., E'_{t_N}\}$ **do**
  $Cluster_k \leftarrow \text{SOM}(E'_{t_i})$ for $k = 1$ to K
end for
$Participants : P \leftarrow \{P_1, P_2, ..., P_K\}$
for k **do**
  $DP_k \leftarrow \{G''_{t_i} | E'_{t_i} \in Cluster_k\}$
  $DP_k \subset D$
end for
procedure FEDERATEDLEARNING
  for $P_i \in G_{\text{homogeneous}}$ **do**
    $\Theta_{\text{global}} \leftarrow \text{DownloadGlobalParameters}()$
    $\Theta^i_{\text{local}} \leftarrow \text{TrainLocalDFDG}(\Theta_{\text{global}}, DP_k)$
    $\text{Weightage}^i \leftarrow \text{ComputeWeightage}(P_i)$
    $\Theta_{\text{global}} \leftarrow \text{FedAvg}(\Theta_{\text{global}}, \Theta^i_{\text{local}} * \text{Weightage}^i)$
  end for
end procedure
for each communication round $\leftarrow 1$ to R **do**
  $G_{\text{homogeneous}} \leftarrow \text{GroupParticipants}(P_{i_{\text{performance}}}, DP_{k_{\text{distribution}}})$
  $\text{FederatedLearning}(G_{\text{homogeneous}})$
  $\text{EvaluateGlobalGNN}(\Theta_{\text{global}}, G_{\text{homogeneous}})$
end for
$\text{PerformForecasting}(\Theta_{\text{global}})$

D. Datasets

1) MetrLA: A traffic forecasting dataset based on Los Angeles Metropolitan traffic conditions. The dataset contains traffic readings collected from 207 loop detectors on highways in Los Angeles County in aggregated 5-minute intervals for 4 months between March 2012 to June 2012. [3]

2) PEMS-BAY: This traffic dataset is collected by California Transportation Agencies (CalTrans) Performance Measurement System (PeMS). It is represented by a network of 325 traffic sensors in the Bay Area with 6 months of traffic readings ranging from Jan 1st 2017 to May 31th 2017 in 5-minute intervals. [3]

II. EXPERIMENTS

A. Experiment 1

Fig. 4 displaying the multi-faceted chart effectively visualizes the performance of various algorithms on the METR-LA dataset, showcasing their comparative strengths and weaknesses. The chart is methodically organized, with distinct sections for each key evaluation metric: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE). Within these sections, sub-charts correspond to different forecast horizons—15, 30, and 60 minutes. These sub-charts feature line plots for each algorithm, including DFDG, facilitating a clear and thorough comparison across various algorithms and timeframes.

The chart particularly highlights the standout performance of the DFDG algorithm on the METR-LA dataset. DFDG demonstrates consistently superior results across multiple metrics and all forecast horizons. It is noteworthy that DFDG records the lowest MAE values for the 15, 30, and 60-minute forecasts, underscoring its exceptional precision in predictions. Additionally, DFDG shows commendable performance in MAPE, especially for the 30 and 60-minute forecasts. In contrast, while algorithms like MegaCRN and MTGNN display competitive results, particularly in RMSE, their performance does not consistently parallel the comprehensive effectiveness exhibited by DFDG across all the evaluated metrics.

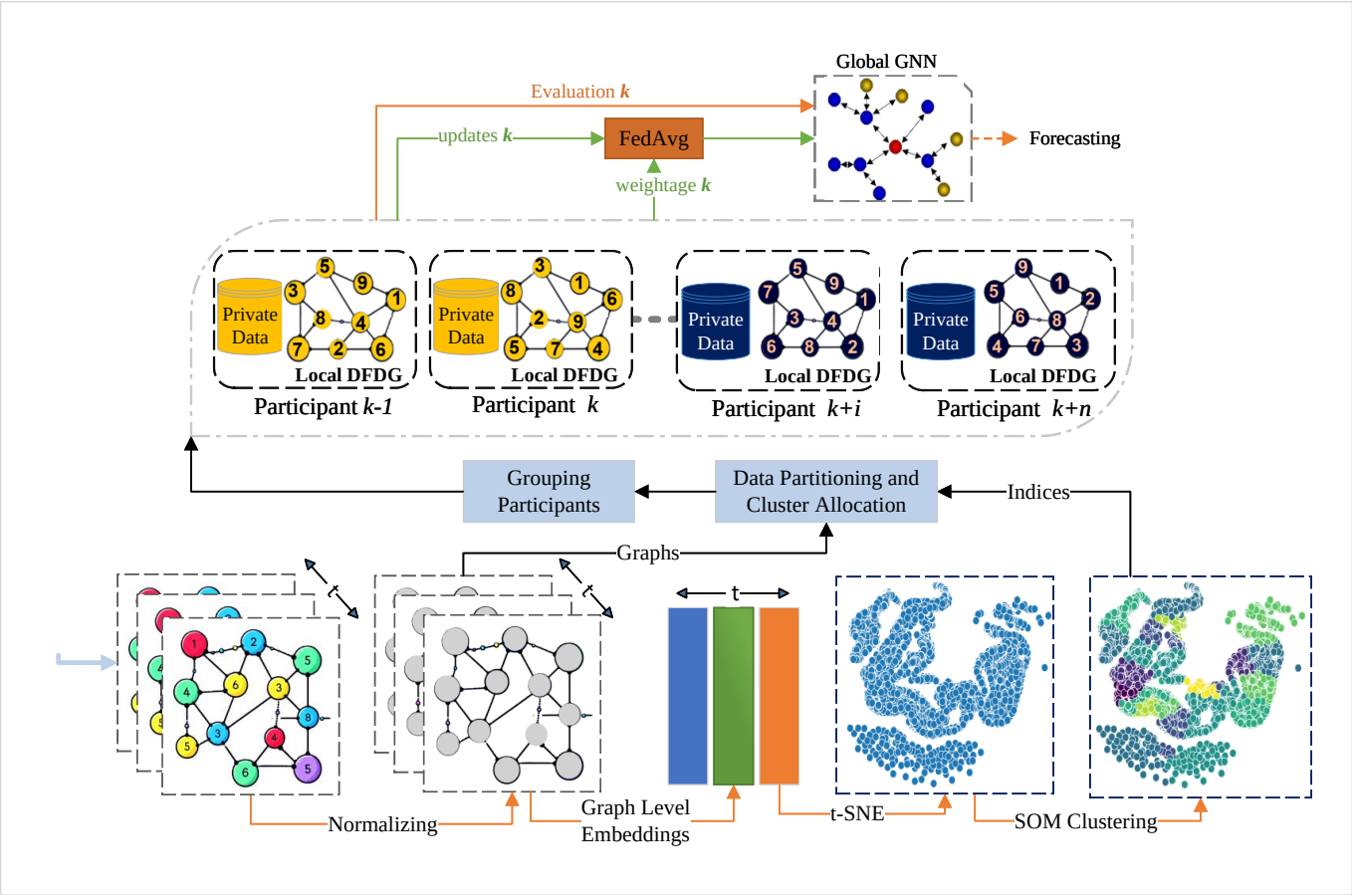

Fig. 1. The schematic delineates a sequential procedure implemented in our methodology

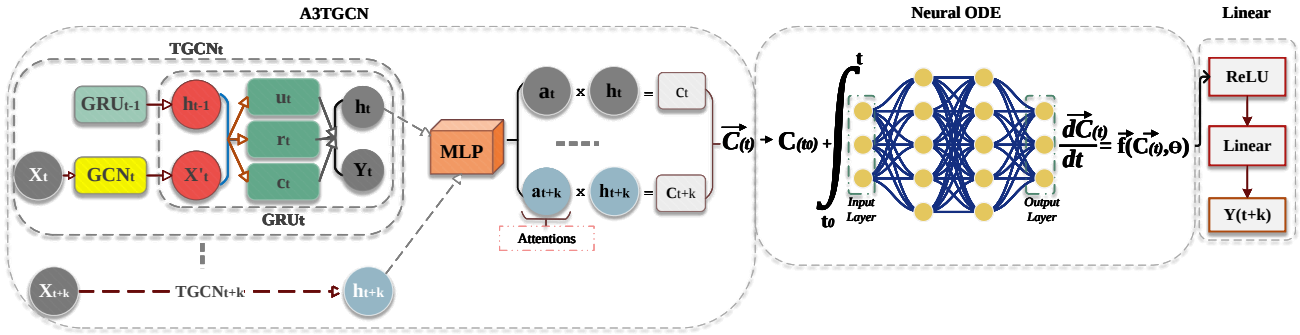

Fig. 2. The schematic illustrates the augmented localDFDG model architecture of participants, an extension of A3TGCN achieved through the integration of Ordinary Differential Equation (ODE) Networks.

B. Experiment 2

Figure 5 effectively presents a series of line charts for different evaluation metrics – Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) – to illustrate the performance of various algorithms on the PEMS-BAY dataset. These charts cover different forecast horizons, specifically 15, 30, and 60 minutes, and each line within the chart corresponds to a different algorithm, highlighting how their performance varies across these metrics and timeframes.

From the charts, it's evident that the DFDG algorithm consistently exhibits superior performance in most metrics across all forecast horizons, particularly excelling in MAE and RMSE. While other algorithms, such as MegaCRN and GW-Net, display competitive performance levels, they do not uniformly match the effectiveness of the DFDG algorithm

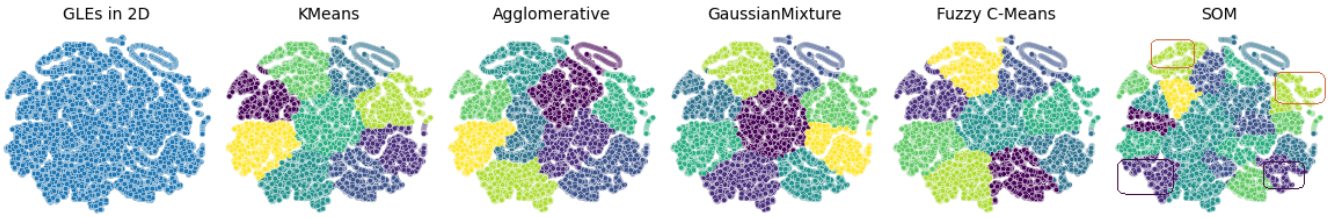

Fig. 3. The diagram delineates clustering techniques for GLEs clustering, highlighting that SOM preserves topological information and local patterns, ensuring accurate spatial relationships, as observed in the encirclement depicting analogous traffic patterns at distinct times (t). In contrast, other approaches, grounded in proximity or distance measures, neglect topological information.

TABLE I
ALGORITHM PERFORMANCE ON METR-LA DATASET

METR-LA	15min/horizon3			30min/horizon6			60min/horizon12		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
HA(Li et al. 2018 [3])	4.16	7.8	13.00%	4.16	7.8	13.00%	4.16	7.8	13.00%
STGCN(Yu,Yin,andZhu2018 [4])	2.88	5.74	7.62%	3.47	7.24	9.57%	4.59	9.4	12.70%
DCRNN(Li et al. 2018 [3])	2.77	5.38	7.30%	3.15	6.45	8.80%	3.6	7.59	10.50%
GW-Net(Wu et al.2019 [5])	2.69	5.15	6.90%	3.07	6.22	8.37%	3.53	7.37	10.01%
STTN(Xu et al.2020 [6])	2.79	5.48	7.19%	3.16	6.5	8.53%	3.6	7.6	10.16%
GMAN(Zheng et al.2020 [7])	2.8	5.55	7.41%	3.12	6.49	8.73%	3.44	7.35	10.07%
MTGNN(Wu et al.2020 [8])	2.69	5.18	6.86%	3.05	6.17	8.19%	3.49	7.23	9.87%
StemGNN(Cao et al.2020 [9])	2.56	5.06	6.46%	3.01	6.03	8.23%	3.43	7.23	9.85%
AGCRN(Bai et al.2020 [10])	2.86	5.55	7.55%	3.25	6.57	8.99%	3.68	7.56	10.46%
CCRNN(Ye et al.2021 [11])	2.85	5.54	7.50%	3.24	6.54	8.90%	3.73	7.65	10.59%
GTS(Shang et al2021 [12])	2.65	5.2	6.80%	3.05	6.22	8.28%	3.47	7.29	9.83%
PM-MemNet(Lee et al.2022 [13])	2.65	5.29	7.01%	3.03	6.29	8.42%	3.46	7.29	9.97%
MegaCRN(R. et al. [14])	2.52	**4.94**	**6.44%**	2.93	**6.06**	7.96%	3.38	**7.23**	9.72%
DFDG(ours)	**1.83**	5.59	6.96%	**2.02**	6.21	**7.65%**	**2.48**	7.49	**9.29%**

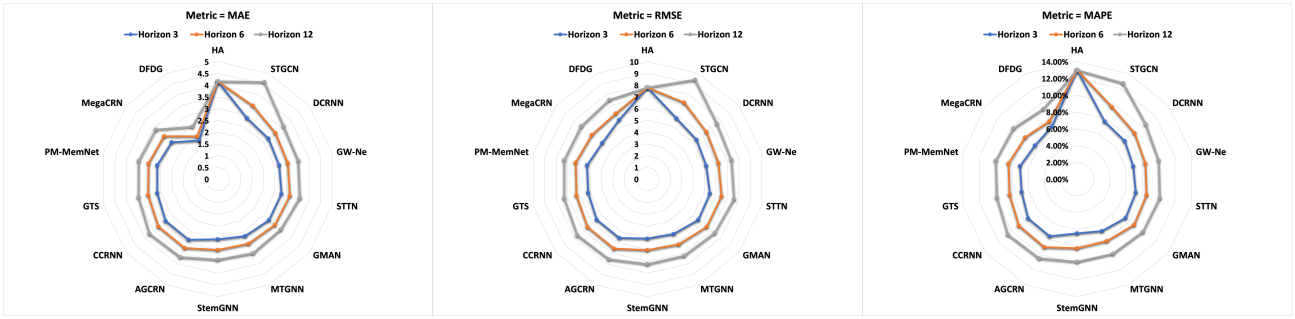

Fig. 4. The Performance of Various Algorithms on the METR-LA Dataset

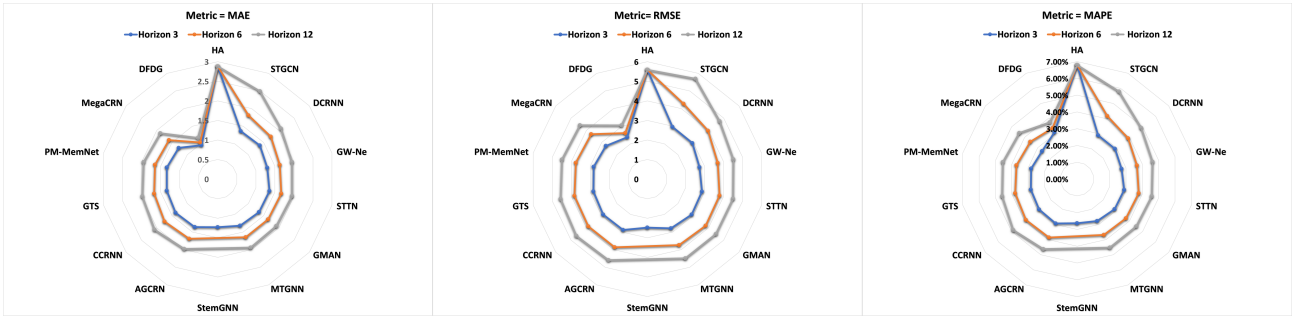

Fig. 5. The Performance of Various Algorithms on the PEMS-BAY Dataset

across all the metrics. This visual representation provides a detailed and comprehensive comparison of the algorithms,

TABLE II
ALGORITHM PERFORMANCE ON PEMS-BAY DATASET

PEMS-BAY	15min/horizon3			30min/horizon6			60min/horizon12		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
HA(Li et al.2018 [3])	2.88	5.59	6.80%	2.88	5.59	6.80%	2.88	5.59	6.80%
STGCN(Yu,Yin,andZhu2018 [4])	1.36	2.96	2.90%	1.81	4.27	4.17%	2.49	5.69	5.79%
DCRNN(Li et al.2018 [3])	1.38	2.95	2.90%	1.74	3.97	3.90%	2.07	4.74	4.90%
GW-Net(Wu et al.2019 [5])	1.3	2.74	2.73%	1.63	3.7	3.67%	1.95	4.52	4.63%
STTN(Xu et al.2020 [6])	1.36	2.87	2.89%	1.67	3.79	3.78%	1.95	4.5	4.58%
GMAN(Zheng et al.2020 [7])	1.35	2.9	2.87%	1.65	3.82	3.74%	1.92	4.49	4.52%
MTGNN(Wu et al.2020 [8])	1.32	2.79	2.77%	1.65	3.74	3.69%	1.94	4.49	4.53%
StemGNN(Cao et al.2020 [9])	1.23	2.48	2.63%		NA			NA	
AGCRN(Bai et al.2020 [10])	1.36	2.88	2.93%	1.69	3.87	3.86%	1.98	4.59	4.63%
CCRNN(Ye et al.2021 [11])	1.38	2.9	2.90%	1.74	3.87	3.90%	2.07	4.65	4.87%
GTS(Shang et al2021 [12])	1.34	2.84	2.83%	1.67	3.83	3.79%	1.98	4.56	4.59%
PM-MemNet(Lee et al.2022 [13])	1.34	2.82	2.81%	1.65	3.76	3.71%	1.95	4.49	4.54%
MegaCRN(R. et al. [14])	1.28	2.72	**2.67%**	1.6	3.68	3.57%	1.88	4.42	4.41%
DFDG (ours)	**0.96**	**2.38**	3.10%	**1.04**	**2.61**	**3.37%**	**1.17**	**3.05**	**3.75%**

offering insights into their respective strengths and weaknesses in terms of accuracy and error management under different forecasting conditions.

REFERENCES

[1] J. Bai, J. Zhu, Y. Song, L. Zhao, Z. Hou, R. Du, and H. Li, "A3t-gcn: Attention temporal graph convolutional network for traffic forecasting," *ISPRS International Journal of Geo-Information*, vol. 10, no. 7, 2021.

[2] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[3] 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.

[4] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI*, 2018, pp. 3634–3640.

[5] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 7 2019, pp. 1907–1913.

[6] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G.-J. Qi, and H. Xiong, "Spatial-temporal transformer networks for traffic flow forecasting," *arXiv preprint arXiv:2001.02908*, 2020.

[7] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, pp. 1234–1241, Apr. 2020.

[8] 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*, ser. KDD '20, 2020, p. 753–763.

[9] D. Cao, Y. Wang, J. Duan, C. Zhang, X. Zhu, C. Huang, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, "Spectral temporal graph neural network for multivariate time-series forecasting," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 17 766–17 778.

[10] L. BAI, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 17 804–17 815.

[11] J. Ye, L. Sun, B. Du, Y. Fu, and H. Xiong, "Coupled layer-wise graph convolution for transportation demand prediction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, pp. 4617–4625, May 2021.

[12] C. Shang, J. Chen, and J. Bi, "Discrete graph structure learning for forecasting multiple time series," in *International Conference on Learning Representations*, 2021.

[13] H. Lee, S. Jin, H. Chu, H. Lim, and S. Ko, "Learning to remember patterns: Pattern matching memory networks for traffic forecasting," in *International Conference on Learning Representations*, 2022.

[14] 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.