Project: Federated GNNs Learning from Dynamic Graphs for Traffic Forecasting by Muhammad Usman

This is the updated and continued form of our previous report. Last portion of the technical section contains the updated information.

Introduction:
This project, conducting by Muhammad Usman, focuses on the critical role of traffic forecasting in optimizing transportation systems and urban planning. Traditional forecasting methods often struggle with the complexity and dynamism of transportation networks. The study explores the application of Federated Graph Neural Networks (GNNs), a cutting-edge machine learning technique, to enhance traffic forecasting using dynamic graphs.

Objectives:
The primary objective of this project is to develop an advanced methodology using Federated GNNs for predicting traffic patterns in scenarios with evolving network structures.

Challenges:
Training distributed dynamic GNNs poses several challenges, including massive parameter communication, model convergence issues, and workload imbalance among workers. Addressing these challenges is crucial for achieving efficient and accurate traffic forecasting.

Data Sources:
The project utilizes openly available dynamic graph benchmark datasets, including METR_LA and PEMS_Bay, as primary data sources.

Methodologies:
The project's approach centers on Federated GNNs as the core machine learning framework for traffic forecasting. The methodology comprises data collection from real-world traffic historical speed data, dynamic graph embedding, unsupervised data partitioning, the implementation of a standalone Federated Learning Framework, and model evaluation using metrics like MAE, RMSE, and MAPE.

Timeline:
The project is expected to span a duration of 4-6 months to complete the outlined tasks and achieve the defined objectives.

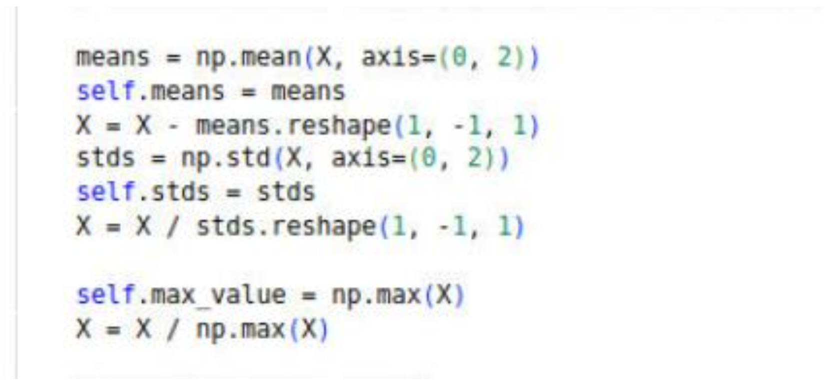
Project Summarization.
In “Federated GNNs Learning from Dynamic Graphs for Traffic Forecasting" project the focus is on enhancing traffic speed forecasting in the context of complex and ever-changing transportation networks. The project's objectives include developing an advanced methodology that utilizes Federated Graph Neural Networks (GNNs) to predict traffic patterns. Challenges related to

distributed GNN training are addressed, and openly available benchmark datasets like METR_LA and PEMS_Bay are used as primary data sources. The proposed methodology encompasses data collection, dynamic graph embedding, unsupervised data partitioning, grouping relevant clients based on their previous performance and its data distribution, the implementation of a standalone Federated Learning algorithm, and model evaluation. This project is anticipated to span 4-6 months, aiming to provide valuable insights into traffic forecasting techniques using state-of-the-art machine learning approaches.

Technical Report:

1. Graph Normalization:

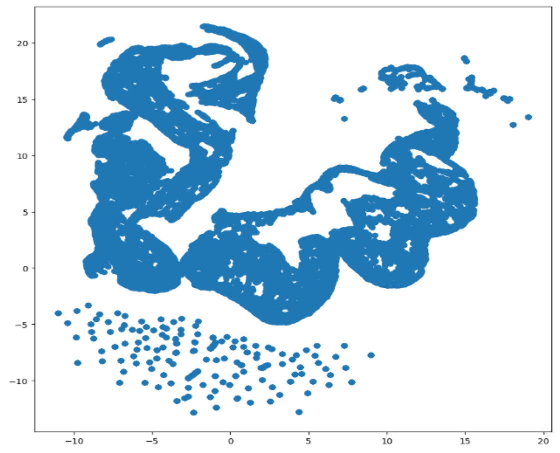
We first normalize all the Graphs features(X) by zero mean and unit standard deviation. After that the overall features are again normalized with the maximum value of the entire dataset.



```
means = np.mean(X, axis=(0, 2))
self.means = means
X = X - means.reshape(1, -1, 1)
stds = np.std(X, axis=(0, 2))
self.stds = stds
X = X / stds.reshape(1, -1, 1)

self.max_value = np.max(X)
X = X / np.max(X)
```

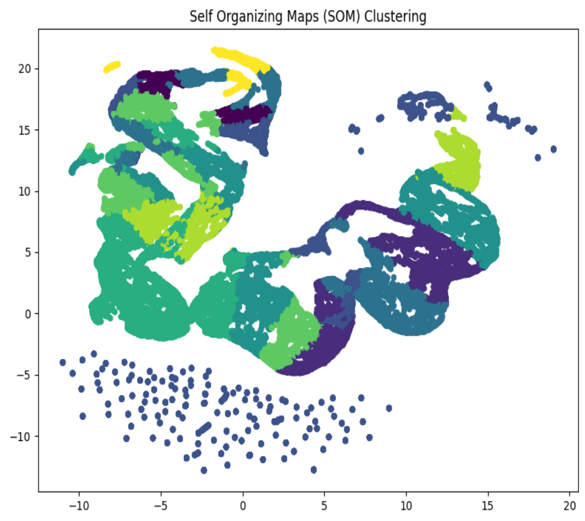
2. Converting all the graphs to its Graph Level Embeddings using A3TGCN:

We then convert all the graphs to Graph Level Embeddings for low level vector representation. All the graphs' embeddings have been plotted and shown in the following image.



3. Unsupervised Client Construction based on similar traffic patterns:

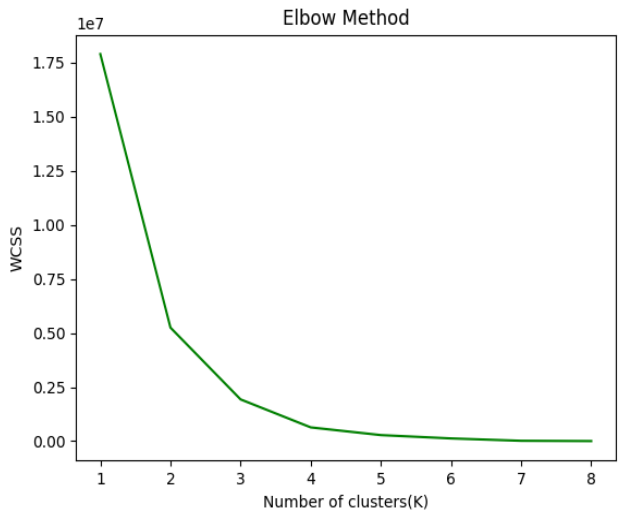
We then create clusters of embeddings using unsupervised learning and assign each cluster to a client. This means that all the graph embeddings within a specific cluster serve as an index to retrieve graphs from the main dataset and assign them to that specific client. Here is a detailed result of clustering the graph-level embeddings using the SOM method, for better understanding.

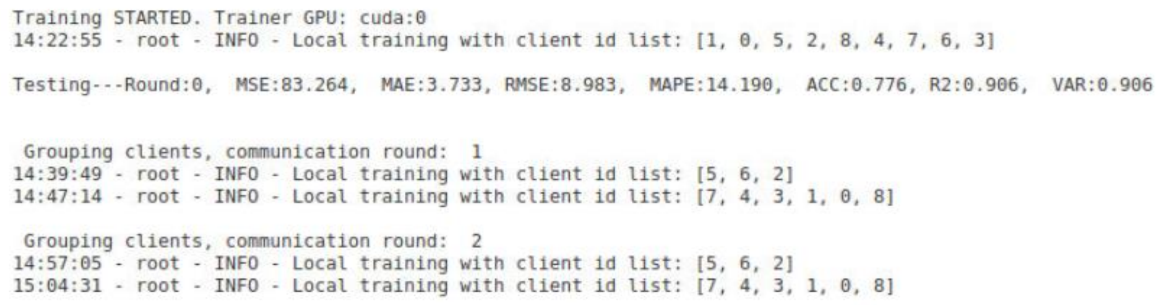


Each color on the map represents a cluster. The above image has nine colors, so there are nine clusters. Since we are dealing with temporal graphs, similar patterns that occur at a specific time stamp and belong to a cluster have been colored the same. Same colors at different places on the map represent which cluster these dynamic patterns are associated with.

4. Grouping the clients based on their data distribution and previous performance:

After the initial round of communication between the server and clients, the server proceeds to categorize clients based on their past performances and data distribution. It extracts fourteen statistical features from each client and utilizes a k-means algorithm to assign labels to them, effectively grouping clients with similar labels. To determine the optimal number of clusters, we employ the Elbow method. Here's an example that suggests that having "K=2" groups may yield better results. A snapshot of the output is also provided for better understanding.



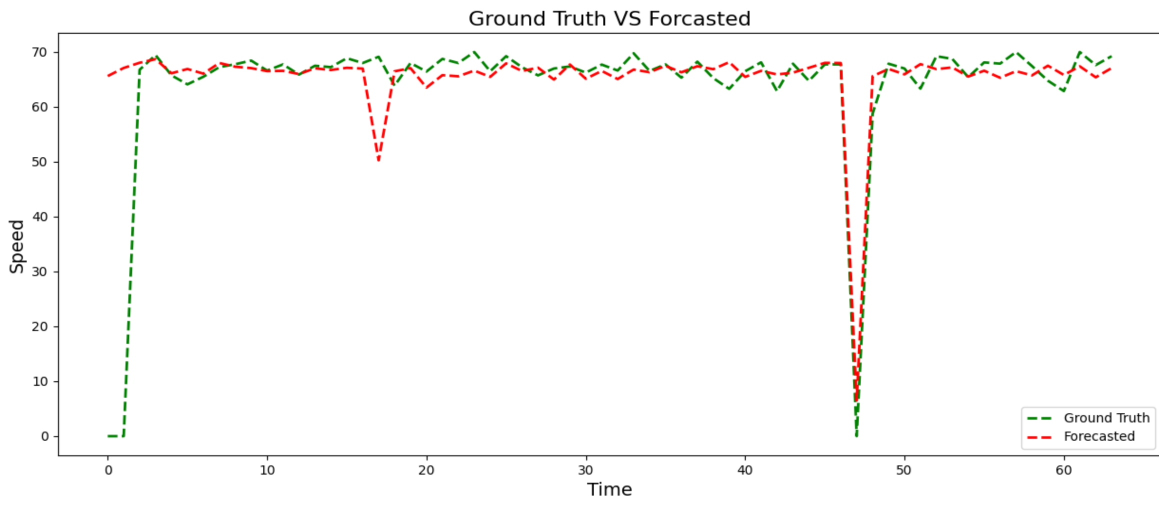


```
Training STARTED. Trainer GPU: cuda:0
14:22:55 - root - INFO - Local training with client id list: [1, 0, 5, 2, 8, 4, 7, 6, 3]

Testing---Round:0,  MSE:83.264,  MAE:3.733, RMSE:8.983,  MAPE:14.190,  ACC:0.776, R2:0.906,  VAR:0.906


 Grouping clients, communication round:  1
14:39:49 - root - INFO - Local training with client id list: [5, 6, 2]
14:47:14 - root - INFO - Local training with client id list: [7, 4, 3, 1, 0, 8]

 Grouping clients, communication round:  2
14:57:05 - root - INFO - Local training with client id list: [5, 6, 2]
15:04:31 - root - INFO - Local training with client id list: [7, 4, 3, 1, 0, 8]
```

5. Training Federated Global Attention based GNN Model:

In this example, we trained a federated GNN model with 9 clients using the Metr_LA dataset. Initially, the dataset was partitioned among these 9 clients based on their shared characteristics. Each client possessed its own local dynamic graph dataset and employed a GNN-based A3TGCN topology for modeling. Clients trained their models locally and periodically updated the server (global model) with their local parameters. The server, in turn, incorporated these updates from each client. In this project, all clients utilized the A3TGCN machine learning topology, which combines GCN (Graph Convolutional Network) with an attention mechanism to capture spatial information in dynamic graphs and GRU (Graph Recurrent Unit) to capture temporal information.

The server employed the FedAvg mechanism to aggregate parameters received from the client groups. The resulting figure illustrates the algorithm's performance on unseen data, with an MAE score of 3.73 achieved yet for 12 Horizons (60 minutes time frames). Below illustration forecast the speed forecasting for 12 Horizons in the future on Metr_LA dataset.



<u>6. Applying Adaptive Boosting to federated GNN model.</u>

Ada boosting in federated learning refers to the process of enhancing or optimizing the training process to improve model performance using Weighted Updates. Assign different weights to clients' updates based on their performance. This allows us to give more importance to clients that provide better results and reduce the influence of clients with noisy or unreliable data and results.

1. **For each iteration (t = 1, 2, ..., T)**:

   a. **Train a client**:

   - Train a client classifier using the current sample weights. The client classifier should aim to minimize the error

   b. **Calculate the Weighted Error**

   - Calculate the weighted error of the client classifier over the training set.

$$w_i^{(t+1)} = w_i^{(t)} \cdot \exp\left(-\alpha_t \cdot y_i \cdot h_t(x_i)\right)$$

   c. **Compute the Client eight**:

   - Compute the weight of the client classifier in the final model:

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

   d. **Update Client Weights before FedAvg at server side.**

   - Update the sample weights for the next iteration:

$$\epsilon_t = \sum_{i=1}^{N} w_i^{(t)} \cdot 1(h_t(x_i) \neq y_i)$$

7. Evaluation and validation:

The Performance of algorithm is evaluated on PEMS_BAY and METR-LA dataset, however I am still adding new features to the algorithm to increase the performance.

Validation on METR-LA

METR-LA	15min/horizon3			30min/horizon6			60min/horizon12		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
HA(Lietal.2018)	4.16	7.80	13.00%	4.16	7.80	13.00%	4.16	7.80	13.00%
STGCN(Yu,Yin,andZhu2018)	2.88	5.74	7.62%	3.47	7.24	9.57%	4.59	9.40	12.70%
DCRNN(Lietal.2018)	2.77	5.38	7.30%	3.15	6.45	8.80%	3.60	7.59	10.50%
GW-Net(Wuetal.2019)	2.69	5.15	6.90%	3.07	6.22	8.37%	3.53	7.37	10.01%
STTN(Xuetal.2020)	2.79	5.48	7.19%	3.16	6.50	8.53%	3.60	7.60	10.16%
GMAN(Zhengetal.2020)?	2.80	5.55	7.41%	3.12	6.49	8.73%	3.44	7.35	10.07%
MTGNN(Wuetal.2020)	2.69	5.18	6.86%	3.05	6.17	8.19%	3.49	7.23	9.87%
StemGNN(Caoetal.2020)†	2.56	5.06	6.46%	3.01	6.03	8.23%	3.43	7.23	9.85%
AGCRN(Baietal.2020)	2.86	5.55	7.55%	3.25	6.57	8.99%	3.68	7.56	10.46%
CCRNN(Yeetal.2021)	2.85	5.54	7.50%	3.24	6.54	8.90%	3.73	7.65	10.59%
GTS(Shang,Chen,andBi2021)?	2.65	5.20	6.80%	3.05	6.22	8.28%	3.47	7.29	9.83%
PM-MemNet(Leeetal.2022)	2.65	5.29	7.01%	3.03	6.29	8.42%	3.46	7.29	9.97%
MegaCRN(AAAI-23)	2.52	4.94	6.44%	2.93	6.06	7.96%	3.38	7.23	9.72%
Ours	3.13	7.399	12.02%	3.3	8.124	12.38%	3.56	8.723	13.47%

Validation on PEMS_Bay

PEMS-BAY	15min/horizon3			30min/horizon6			60min/horizon12		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
HA(Lietal.2018)	2.88	5.59	6.80%	2.88	5.59	6.80%	2.88	5.59	6.80%
STGCN(Yu,Yin,andZhu2018)	1.36	2.96	2.90%	1.81	4.27	4.17%	2.49	5.69	5.79%
DCRNN(Lietal.2018)	1.38	2.95	2.90%	1.74	3.97	3.90%	2.07	4.74	4.90%
GW-Net(Wuetal.2019)	1.30	2.74	2.73%	1.63	3.70	3.67%	1.95	4.52	4.63%
STTN(Xuetal.2020)	1.36	2.87	2.89%	1.67	3.79	3.78%	1.95	4.50	4.58%
GMAN(Zhengetal.2020)?	1.35	2.90	2.87%	1.65	3.82	3.74%	1.92	4.49	4.52%
MTGNN(Wuetal.2020)	1.32	2.79	2.77%	1.65	3.74	3.69%	1.94	4.49	4.53%
StemGNN(Caoetal.2020)†	1.23	2.48	2.63%	NA			NA		
AGCRN(Baietal.2020)	1.36	2.88	2.93%	1.69	3.87	3.86%	1.98	4.59	4.63%
CCRNN(Yeetal.2021)	1.38	2.90	2.90%	1.74	3.87	3.90%	2.07	4.65	4.87%
GTS(Shang,Chen,andBi2021)?	1.34	2.84	2.83%	1.67	3.83	3.79%	1.98	4.56	4.59%
PM-MemNet(Leeetal.2022)?	1.34	2.82	2.81%	1.65	3.76	3.71%	1.95	4.49	4.54%
MegaCRN(AAAi-2023)	1.28	2.72	2.67%	1.60	3.68	3.57%	1.88	4.42	4.41%
Ours	1.590	3.549	5.16%	1.60	3.67	5.3%	1.735	4.01	5.06%