Big data analytics project using the CA traffic sensors dataset

(https://github.com/liuxu77/LargeST/tree/main)

Objectives

Data Ingestion and Exploration

- Load multi-year traffic data from the LargeST dataset into a distributed environment (RaaS/Spark).
- Perform preprocessing steps such as handling missing values etc
- Visualize the spatial distribution of traffic sensors across California.

Spatio-Temporal Analysis

- Analyze traffic speed patterns across time (daily and weekly cycles/ week vs. weekend) and space (highways, corridors, regions).
- Identify consistently congested road segments, unusually fast/free-flowing segments, and peak vs off-peak behavior.
- Detect temporal hotspots and periods of abnormal traffic activity.

Graph-Based Network Modeling

- Construct a traffic network where nodes represent sensors and edges encode physical road connectivity using the adjacency matrix.
- Compute network metrics such as degree centrality, betweenness centrality, and clustering coefficient.
- Identify major mobility hubs, structurally important sensors, and potential bottlenecks in the road network.

Clustering of Traffic Sensors

- Cluster sensors based on their temporal traffic patterns and/or graph position in the road network.
- Discover groups of road segments exhibiting similar dynamics (e.g., consistently congested, highly variable, stable/free-flowing).
- Provide insights that could support regional traffic management and planning.

Traffic Forecasting

- Develop/ Apply forecasting models to predict future traffic speeds at selected sensors.

State of the art

Team Members

- Andreea-Daniela Lupu (MIAO)
- Năstasă Baraș Luca (SAI)
- Nechifor Alexandru (MIAO)
- Roșcan Teodor (MISS)

Research & Data Study

Dataset Overview

This dataset is significantly larger in size than the previous existing traffic benchmark datasets.

The dataset consists of ≈ 8,600 traffic sensors deployed on highways and major roads across California and it has 5 years of historical data (2017–2021), providing long-term temporal coverage. Measurements recorded at regular time intervals (e.g., every 5 minutes) are giving very high temporal resolution.

“To undertake a more meticulous analysis of traffic patterns across diverse regions of California, we construct three subsets of CA by selecting three representative areas within CA. The first one is GLA, which contains 3,834 sensors installed in 5 counties of the Greater Los Angeles area: Los Angeles, Orange, Riverside, San Bernardino, and Ventura. The second sub-dataset GBA, includes 2,352 sensors in 11 counties situated in the Greater Bay Area: Alameda, Contra Costa, Marin, Napa, San Benito, San Francisco, San Mateo, Santa Clara, Santa Cruz, Solano, and Sonoma. The smallest sub-dataset SD, comprises 716 sensors only in San Diego county. In addition to the county information, we provide other meta knowledge for each node, including their coordinates, district in PeMS, the highway they are located on, directions of travel, and number of lanes”.

Although the LargeST dataset is primarily used for traffic forecasting, the additional analyses performed in this project (graph-based analysis, clustering, hotspot detection, spatio-temporal profiling, and metadata analysis) can also serve as feature enhancement strategies for forecasting models (feature engineering). We plan to use exactly the preprocessing script provided by the authors of this paper and enhance the dataset through feature engineering manners.

Traffic forecasting is a long-standing research problem due to the complex spatiotemporal dependencies that govern real road networks. Early work relied on classical time-series models—primarily ARIMA and VAR—which assume linearity and stationarity in

temporal dynamics. However, these assumptions break down in real traffic, where congestion patterns are nonlinear, non-stationary, and propagate across the network. As a result, ARIMA and similar linear models perform poorly on large and complex road systems [1,2].

An important methodological detail in the original traffic forecasting research paper [3] is that it does not reimplement complex spatiotemporal architectures from scratch. Instead, the authors rely on existing open-source implementations from official or widely used GitHub repositories. The LargeST paper explicitly follows this practice: to generate their benchmark results, the authors adopt reference implementations of DCRNN, AGCRN, STGCN, GWNet, ASTGCN, STTN, STGODE, DGCRN, DSTAGNN, and D2STGNN, and train these models directly on the LargeST dataset. Their goal is not to reproduce each architecture manually, but to provide a standardized comparison of representative approaches. Below are their benchmarking results:

Table 2: Performance comparisons. We bold the best-performing baseline result. The absence of baselines on the GLA and CA datasets indicates that the models incur out-of-memory issue even when we set batch size to 4 on an A6000 GPU with 48 GB memory. Param: the number of learnable parameters. K: kilo (10^3). M: million (10^6).

Data	Method	Param	Horizon 3			Horizon 6			Horizon 12			Average		
			MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
SD	HL	–	33.61	50.97	20.77%	57.80	84.92	37.73%	101.74	140.14	76.84%	60.79	87.40	41.88%
	LSTM	98K	19.03	30.53	11.81%	25.84	40.87	16.44%	37.63	59.07	25.45%	26.44	41.73	17.20%
	DCRNN	373K	17.14	27.47	11.12%	20.99	33.29	13.95%	26.99	42.86	18.67%	21.03	33.37	14.13%
	AGCRN	761K	15.71	27.85	11.48%	18.06	31.51	13.06%	21.86	39.44	16.52%	18.09	32.01	13.28%
	STGCN	508K	17.45	29.99	12.42%	19.55	33.69	13.68%	23.21	41.23	16.32%	19.67	34.14	13.86%
	GWNET	311K	15.24	25.13	9.86%	17.74	29.51	11.70%	**21.56**	**36.82**	15.13%	**17.74**	29.62	11.88%
	ASTGCN	2.2M	19.56	31.33	12.18%	24.13	37.95	15.38%	30.96	49.17	21.98%	23.70	37.63	15.65%
	STTN	114K	16.22	26.22	10.63%	18.76	30.98	12.80%	22.62	39.09	16.14%	18.69	31.11	12.82%
	STGODE	729K	16.75	28.04	11.00%	19.71	33.56	13.16%	23.67	42.12	16.58%	19.55	33.57	13.22%
	DSTAGNN	3.9M	18.13	28.96	11.38%	21.71	34.44	13.93%	27.51	43.95	19.34%	21.82	34.68	14.40%
	DGCRN	243K	15.34	25.35	10.01%	18.05	30.06	11.90%	22.06	37.51	15.27%	18.02	30.09	12.07%
	D^{2}STGNN	406K	**14.92**	**24.95**	**9.56%**	**17.52**	**29.24**	**11.36%**	22.62	37.14	**14.86%**	17.85	**29.51**	**11.54%**
GBA	HL	–	32.57	48.42	22.78%	53.79	77.08	43.01%	92.64	126.22	92.85%	56.44	79.82	48.87%
	LSTM	98K	20.38	33.34	15.47%	27.56	43.57	23.52%	39.03	60.59	37.48%	27.96	44.21	24.48%
	DCRNN	373K	18.71	30.36	14.72%	23.06	36.16	20.45%	29.85	46.06	29.93%	23.13	36.35	20.84%
	AGCRN	777K	18.31	30.24	14.27%	21.27	34.72	16.89%	**24.85**	**40.18**	20.80%	21.01	34.25	16.90%
	STGCN	1.3M	21.05	34.51	16.42%	23.63	38.92	18.35%	26.87	44.45	21.92%	23.42	38.57	18.46%
	GWNET	344K	17.85	29.12	13.92%	21.11	**33.69**	17.79%	25.58	40.19	23.48%	20.91	**33.41**	17.66%
	ASTGCN	22.3M	21.46	33.86	17.24%	26.96	41.38	24.22%	34.29	52.44	32.53%	26.47	40.99	23.65%
	STTN	218K	18.25	29.64	14.05%	21.06	33.87	17.03%	25.29	40.58	21.20%	20.97	33.78	16.84%
	STGODE	788K	18.84	30.51	15.43%	22.04	35.61	18.42%	26.22	42.90	22.83%	21.79	35.37	18.26%
	DSTAGNN	26.9M	19.73	31.39	15.42%	24.21	37.70	20.99%	30.12	46.40	28.16%	23.82	37.29	20.16%
	DGCRN	374K	18.02	29.49	14.13%	21.08	34.03	16.94%	25.25	40.63	21.15%	20.91	33.83	16.88%
	D^{2}STGNN	446K	**17.54**	**28.94**	**12.12%**	**20.92**	33.92	**14.89%**	25.48	40.99	**19.83%**	**20.71**	33.65	**15.04%**
GLA	HL	–	33.66	50.91	19.16%	56.88	83.54	34.85%	98.45	137.52	71.14%	59.58	86.19	38.76%
	LSTM	98K	20.02	32.41	11.36%	27.73	44.05	16.49%	39.55	61.65	25.68%	28.05	44.38	17.23%
	DCRNN	373K	18.41	29.23	10.94%	23.16	36.15	14.14%	30.26	46.85	19.68%	23.17	36.19	14.40%
	AGCRN	792K	**17.27**	29.70	10.78%	**20.38**	34.82	**12.70%**	**24.59**	42.59	**16.03%**	**20.25**	34.84	**12.87%**
	STGCN	2.1M	19.86	34.10	12.40%	22.75	38.91	14.11%	26.70	45.78	17.00%	22.64	38.81	14.17%
	GWNET	374K	17.28	**27.68**	**10.18%**	21.31	**33.70**	13.02%	26.99	**42.51**	17.64%	21.20	**33.58**	13.18%
	ASTGCN	59.1M	21.89	34.17	13.29%	29.54	45.01	19.36%	39.02	58.81	29.23%	28.99	44.33	19.62%
	STGODE	841K	18.10	30.02	11.18%	21.71	36.46	13.64%	26.45	45.09	17.60%	21.49	36.14	13.72%
	DSTAGNN	66.3M	19.49	31.08	11.50%	24.27	38.43	15.24%	30.92	48.52	20.45%	24.13	38.15	15.07%
CA	HL	–	30.72	46.96	20.43%	51.56	76.48	37.22%	89.31	125.71	76.80%	54.10	78.97	41.61%
	LSTM	98K	19.04	31.28	13.19%	26.49	42.63	19.57%	38.22	60.29	30.28%	26.89	43.11	20.16%
	DCRNN	373K	17.55	28.21	12.68%	21.79	34.27	16.67%	28.56	44.34	23.84%	21.87	34.41	17.06%
	STGCN	4.5M	18.99	32.37	14.84%	21.37	36.46	**16.27%**	**24.94**	**42.59**	**19.74%**	21.33	36.39	**16.53%**
	GWNET	469K	**17.14**	**27.81**	**12.62%**	21.68	**34.16**	17.14%	28.58	44.13	24.24%	21.72	**34.20**	17.40%
	STGODE	1.0M	17.57	29.91	13.91%	**20.98**	36.62	16.88%	25.46	45.99	21.00%	**20.77**	36.60	16.80%

Their work highlights that HL and LSTM perform worst on LargeST dataset, as these

algorithms only consider temporal patterns and fail to capture the spatial nature of traffic networks; AGCRN (RNN-based) and GWNet (TCN-based) perform best, as they use adaptive adjacency matrices, which allow the models to learn dynamic and fully connected graph structures rather than relying solely on static road topology.

Recent dynamic GNNs such as DGCRN and D2STGNN also achieve strong performance on medium-scale datasets by modeling evolving spatial topology, although their computational complexity prevents them from scaling to larger datasets like GLA and CA. These findings highlight the central importance of graph structure in traffic modeling and motivate the graph-based analysis we perform in this project.

Motivation & Relevance

- Traffic prediction is essential for planning, congestion mitigation, and mobility intelligence
- Large-scale spatio-temporal datasets enable regional-level insights
- Dataset size provides a realistic big data environment (multi-year, millions of rows per region)

Technologies & Platforms

Processing Environment

RaaS Cluster Platform, UAIC University

- Distributed Spark processing
- HDFS / distributed storage
- Multi-worker compute environment suitable for multi-year traffic logs

Libraries & Tools

- PySpark – distributed data ingestion + preprocessing
- Spark SQL – querying and aggregations
- Spark MLlib – clustering + model pipelines
- GraphFrames / GraphX / NetworkX – graph modeling
- Matplotlib / Seaborn / Plotly – analytics visualization

- GeoPandas / Kepler.gl – spatial analysis maps

Similar Products & Relevant Links

Traffic Forecasting Models / Benchmarks

- LargeST (official repo): https://github.com/liuxu77/LargeST
- PeMS Traffic Data System: widely used traffic sensor dataset
- DCRNN (Diffusion Convolutional Recurrent Neural Network)
- STGCN (Spatio-Temporal Graph Convolutional Networks)
- ASTGCN (Attention-based STGCN)
- GraphWaveNet

Traffic Analytics Platforms

- Google Maps Traffic Layer
- Waze live traffic analytics
- Open Traffic by Grab
- TomTom Traffic Index

Relevant Research Papers

- "Large-Scale Traffic Speed Forecasting"
  "Graph Neural Networks for Traffic Prediction"
- "Spatio-Temporal Hotspot Detection in Urban Road Networks"

Main Features of the Project

Distributed Data Engineering

- Ingest and process multi-year sensor data using Spark
- Handle missing values, faulty sensors, and irregular sampling

- Efficient partitioning (by region/year/month)

Spatio-Temporal Analytics

- Daily/weekly/seasonal traffic patterns
- Hotspot identification (consistent congestion areas)
- Peak vs off-peak comparisons
- Event/anomaly detection

Road Network Graph Modeling

- Build a sensor network graph from adjacency metadata
- Compute centrality metrics to identify:
    - Hubs
    - Bottlenecks
    - Critical road segments

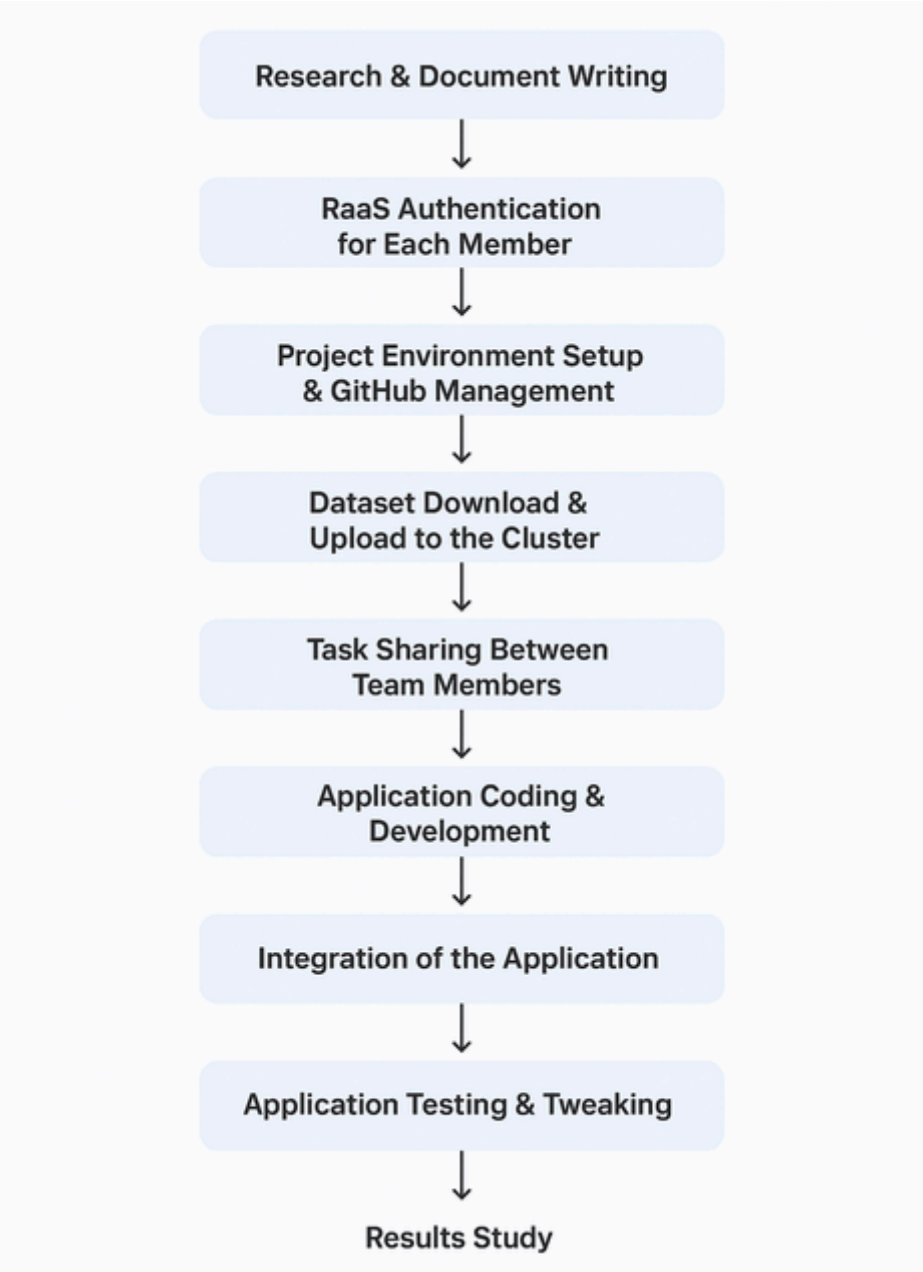
Clustering of Traffic Sensors

- Group sensors with similar behavior
- Region-based pattern discovery
- Insights for urban planning and congestion management

Traffic Forecasting (optional/bonus)

- Predict future speeds for selected sensors
- Compare classical ML vs deep learning approaches

Development Plan



1. Research & Document Writing (All)

We start by getting familiar with the dataset, the problem space, and similar work.
Each team member reads about traffic analytics, GNN forecasting, clustering, graph modeling, etc.
We document the findings, outline the project structure, and agree on what we will build.

---

2. RaaS Authentication for All Members (All)

Everyone sets up their **UAIC RaaS account**, tests access, and verifies that Spark jobs run correctly.
This phase ensures the entire team can work on the cluster without limitations.

---

3. Project Environment Setup & GitHub Management

We create the project repository on GitHub and set up:

- Branching strategy
- Folder structure
- Coding conventions
- Issue tracking / task assignments

On the RaaS cluster, we configure:

- Python environment
- Spark config
- Dependencies for analytics, graph processing, and visualization

This is where the “real workspace” gets organized.

---

4. Dataset Download & Upload to the Cluster

We download the LargeST dataset from the official source and upload it to the distributed storage.
Files are checked for:

- Integrity
- Correct formatting
- Expected folder structure

We also make sure the dataset is properly partitioned (e.g., by region and year) for faster Spark processing.

---

5. Task Sharing Between Team Members

We divide the work so everyone has a clear responsibility. For example:

- One member handles data ingestion + preprocessing (Andreea)
- Another focuses on spatio-temporal analysis (Alex)
- Another builds the graph + computes metrics (Andreea)
- Someone works on clustering (Luca)
- Another member handles forecasting models (Teodor)
- Another prepares the dashboard/visualizations (Luca)
- One person manages integration & documentation (All)

Each task runs in parallel to speed up development.

---

6. Application Coding & Development

This is the longest phase.
We implement everything step by step:

- Spark data ingestion
- Data cleaning (missing values, outliers, bad sensors)
- Spatio-temporal exploration scripts
- Graph construction + centrality metrics
- Clustering algorithms
- Forecasting pipelines
- Visualizations and plots

Each component is tested on small data samples before running on the full multi-year dataset.

---

7. Integration of the Application (All)

All individual components are merged into a coherent workflow:

- Ingestion pipeline →
- Preprocessing →
- Analytics →
- Graph →
- Clustering →
- Forecasting →
- Visual output

We test whether the scripts run in sequence, ensure compatibility (data formats, paths, Spark configs), and remove duplicate code.

8. Application Testing & Tweaking (All)

We run the whole application end-to-end on the full dataset.
This phase includes:

- Fixing bugs
- Adjusting parameters (cluster count, centrality thresholds, anomaly detection filters)
- Re-running long jobs on the RaaS cluster
- Improving speed by better partitioning or caching
- Checking if graphs, clusters, and results make sense

It’s the refining phase where we polish the pipeline.

9. Results Study (All)

After everything runs, we take the outputs and analyze them:

- Maps of sensor distributions
- Spatio-temporal plots (daily/weekly patterns)
- Graph network insights (hubs, bottlenecks)
- Clustering patterns (congested vs free-flow sensors)
- Forecasting performance
- Regional comparisons (GLA, GBA, SD)

Finally, we summarize what we found, explain the trends, and integrate the results into the final documentation.