Traffic Prediction using GNN with Temporal Multi-head Attention Networks with City treated as Graph

Darsh Kurmi (dk21jz@brocku.ca)

***Abstract*—This paper presents a novel framework for short-term traffic prediction that integrates Graph Neural Networks (GNNs) with a Temporal Multi-Head Attention mechanism, using real-world city map data to construct a structurally accurate graph representation of the urban traffic network. Traditional models often fall short of capturing the complex spatiotemporal dependencies inherent in dynamic road systems. To address this, the proposed approach models road segments and intersections as graph nodes, with edges reflecting true spatial connectivity based on map topology. The GNN captures spatial relationships between connected regions, while the temporal attention module dynamically weighs historical traffic patterns across multiple time steps, focusing on the most relevant temporal features. This hybrid architecture enables more precise and interpretable traffic flow forecasting by jointly learning from both spatial context and temporal trends. Evaluated on real-world datasets, the model demonstrates improved accuracy and generalizability compared to conventional deep-learning approaches. The integration of geographic structure and temporal adaptability makes this solution well-suited for applications in intelligent transportation systems, such as traffic control, navigation, and autonomous driving. The results underscore the value of combining graph-based learning with temporal attention and map-informed structure for robust and context-aware traffic prediction.**

I. Introduction

In recent years, the proliferation of intelligent transportation systems (ITS) and the increasing availability of real-time mobility data have led to a growing interest in data-driven traffic prediction models [1], [2]. Accurate short-term traffic forecasting plays a pivotal role in enhancing urban mobility, reducing congestion, improving road safety, and enabling effective planning in smart cities [3]. Traditional forecasting methods—such as statistical approaches (e.g., ARIMA), historical average models, and even early machine learning techniques—have proven insufficient for capturing the highly dynamic, non-linear, and complex spatiotemporal nature of urban traffic systems [3].

Modern urban road networks exhibit intricate interdependencies between road segments, intersections, and varying traffic conditions over time. These patterns are inherently non-Euclidean and cannot be adequately modeled using conventional grid-based or Euclidean techniques [4]. This limitation has led to the adoption of graph-based modeling approaches, where road networks and traffic flows are represented as graphs—nodes correspond to road segments, vehicles, or intersections, and edges represent connectivity or flow relationships. Among these, Graph Neural Networks (GNNs) have emerged as a powerful solution to extract spatial dependencies from graph-structured traffic data [4], [2].

Despite the strengths of GNNs in modeling spatial relationships, they often struggle to effectively capture temporal dependencies, particularly in rapidly changing environments such as real-time vehicular networks. Traffic flow is not only spatially interconnected but also highly temporal in nature, influenced by daily patterns, events, weather conditions, and the historical behavior of vehicles [5]. To address this, recent advances have explored the integration of temporal learning mechanisms - such as recurrent neural networks (RNN) [6], temporal convolutional networks (TCNs) [7], and attention-based models [8]—with GNNs [4], to achieve holistic spatio-temporal modeling.

One particularly promising direction is the incorporation of Multi-Head Attention Mechanisms that allow the model to focus on different time steps and spatial relationships with adaptive weights [9]. This enables the system to identify critical temporal segments that are more relevant for future predictions. Moreover, most existing models treat graphs as abstract structures learned from data, often ignoring the real-world city map and topology, which inherently provide structural constraints and road connectivity. Utilizing actual city map information to construct the graph structure introduces geographical awareness and contextual accuracy, making the model more interpretable and aligned with real-world road configurations [5].

A. Motivation

Urban transportation networks are becoming increasingly complex, with growing vehicle densities, dynamic traffic behaviours, and constantly changing conditions. In such environments, the ability to forecast traffic flow not only ensures better route planning and traffic signal optimization but also supports critical services such as autonomous driving, emergency response systems, and vehicle-to-vehicle (V2V) communication in Vehicular Ad-Hoc Networks (VANETs).

However, one of the primary challenges in traffic forecasting lies in **jointly capturing both spatial and temporal dynamics**. Most existing solutions either:

- Overemphasize spatial relations through static graph convolution layers while neglecting time-varying patterns,
- Or apply generic temporal models that fail to incorporate structural road dependencies effectively.

Furthermore, traditional GNN-based methods often rely on learned adjacency matrices or statistical proximity to define the graph. These approaches may miss essential structural details present in the physical city layout—such as one-way roads, traffic signals, intersections, and connectivity constraints—resulting in predictions that are accurate in theory but suboptimal in practice.

Another shortfall is the lack of adaptive temporal sensitivity. Not all past moments contribute equally to a future event. For instance, sudden braking, lane changes, or congestion events at a previous time step may have a higher impact than steady flow in earlier intervals. Integrating a **Multi-Head Attention mechanism** allows the model to dynamically weigh such moments and extract multi-scale temporal information.

Lastly, in the context of rapidly evolving transportation technologies and connected vehicles, models must be designed with **scalability, adaptability, and real-time applicability** in mind. There is a clear need for a solution that brings together:

- The structural integrity of real-world maps,
- The spatial learning power of GNNs,
- And the temporal awareness of attention mechanisms.

B. Objective

The overarching goal of this research is to develop a robust, scalable, and interpretable traffic prediction framework that leverages the strengths of **graph-based learning**, **temporal attention mechanisms**, and **city map-informed network topology**. The specific objectives of the project are outlined as follows:

1) **To design and implement a graph representation of urban traffic systems using real-world city map data.** This includes defining nodes as Intersections or traffic sensors and edges based on actual road connectivity, turn restrictions, shape of the road and other spatial constraints derived from XML-based maps.
2) **To develop a Graph Neural Network-based architecture that captures the spatial dependencies of the road network.**
   The GNN model will aggregate information from connected nodes, enabling the prediction module to learn localized and global traffic behaviour patterns.
3) **To incorporate a Temporal Multi-Head Attention module to model the dynamic evolution of traffic conditions.**
   This module will attend to varying time intervals with different weights, allowing the model to capture influential events and trends in the traffic sequence that are critical for accurate forecasting.
4) **To fuse the spatial and temporal components into a hybrid architecture for short-term traffic prediction.** The integrated model will learn both spatial relations (from the GNN) and temporal dependencies (from the attention mechanism) to forecast traffic conditions, such as speed or vehicle count, for each segment of the road graph.
5) **To evaluate the model using real-world traffic datasets** (e.g., loop detectors or GPS-based taxi trajectories) in combination with map data from platforms like OpenStreetMap or HERE Maps. Performance metrics such as MAE, RMSE, and R^2 will be used for evaluation, along with visualizations for interpretability.
6) **To analyze the interpretability, efficiency, and generalizability of the model** in various traffic conditions, including rush hour, off-peak times, and unexpected congestion scenarios.

C. Outline

The remainder of this report is organized as follows. Section II provides a brief introduction about the concepts used in this study, consisting of VANETs, GNN, and TCNs. This section aims to familiarize you with concepts to better understand the study.

Section III summarizes two studies with similar problem. The First Study is about Traffic Prediction using GNN, which shows how integrating real-world traffic, as a graph, and with the help of GNN, can help understand traffic flow better and do a prediction. Although the second study of the Multi-Head Spatio-temporal Attention Graph Convolutional Network for Traffic Flow Forecasting helps understand the hybrid model structure and how it helps to take time stamps and urban traffic into account.

Section IV defines the problem that this study aims to solve with the help of map data treated as a static graph, where intersections are treated as nodes and roads are treated as edges that connect them.

Section V describes how Data acquisition was done and how the Map was constructed as a graph, how traffic data was incorporated into the Model, and the algorithm and pipeline overview.

Section VI demonstrates how the test was set up, including device requirements and framework requirements. Then, it describes the performance metrics taken into account for this study, and demonstrates how the study outperforms some previous works.

Finally, Section VII concludes the report and gives acknowledgments, and describes the future considerations that can be taken.

II. Background

A. Vehicular Networks (VANETs)

Vehicular Ad-Hoc Networks (VANETs) are a specialized class of Mobile Ad-Hoc Networks (MANETs), developed to provide intelligent communication between vehicles and roadside infrastructure. The core objective of VANETs is to enhance road safety, manage traffic efficiently, and offer infotainment services to passengers by facilitating rapid data exchange in highly dynamic environments. Unlike traditional networks, VANETs operate in a decentralized and infrastructure-less manner, where vehicles act as nodes capable of sending, receiving, and forwarding information [10].

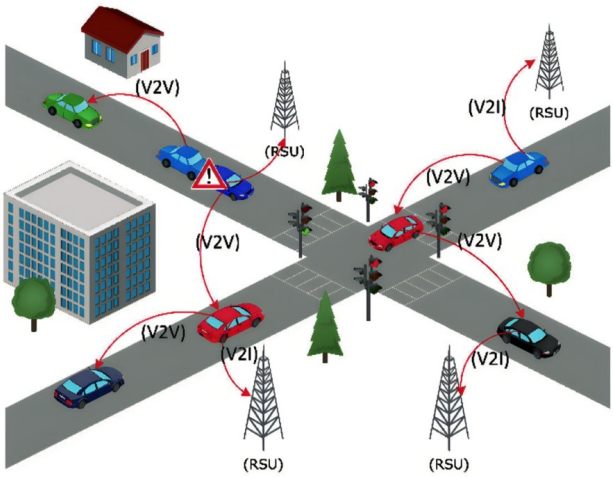

Fig. 1: Overview of Vehicular Ad Hoc Networks (adapted from [11]).

The unique characteristics of VANETs distinguish them from other types of wireless networks. These include high node mobility, rapidly changing network topology, and frequent disconnections due to varying vehicle speeds and directions. As a result, VANETs require robust communication protocols that can handle the dynamic nature of vehicular environments. The communication in VANETs is broadly categorized into Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) interactions. V2V communication is crucial for applications like collision avoidance, lane-changing assistance, and cooperative driving, while V2I enables interaction with traffic signals, toll booths, and traffic management systems, as illustrated in Figure 1 [10].

The architecture of VANETs typically includes On-Board Units (OBUs) installed in vehicles and Roadside Units (RSUs) positioned along highways and intersections. These components work in conjunction to support real-time data dissemination and to manage network resources efficiently. VANETs also rely on short-range wireless communication technologies such as Dedicated Short Range Communication (DSRC) to ensure low-latency and high-speed data transmission [10].

Despite their potential, VANETs face numerous challenges. One major concern is ensuring secure and reliable communication in a highly mobile environment. This includes authentication, data integrity, and protection against malicious attacks such as spoofing, jamming, and denial of service. Moreover, the scalability of VANETs is another critical issue due to the varying density of vehicles in different traffic scenarios. Ensuring quality of service (QoS), managing congestion, and achieving interoperability among heterogeneous systems further complicate VANET deployment and standardization [10].

B. Graph Neural Networks (GNN)

Graph Neural Networks (GNNs) have emerged as powerful tools for processing data that is inherently structured as graphs, where nodes represent entities and edges signify relationships. Unlike traditional neural networks, which operate on fixed-size inputs, GNNs are uniquely capable of capturing the irregular, non-Euclidean nature of graphs, making them especially effective for applications involving complex interactions among many interconnected entities. In the context of Vehicular Ad-Hoc Networks (VANETs), vehicles, roadside units (RSUs), and infrastructure can be modeled as nodes in a dynamic graph, with edges denoting communication links or potential interactions [12].

A key strength of GNNs lies in their ability to perform message passing, a process where each node gathers information from its neighbours and updates its own representation iteratively. This makes GNNs well-suited for applications where local context plays a significant role in global behaviour. For example, in your VANET clustering project, GNNs can be used to represent each vehicle's state (location, speed, direction, etc.) as a node feature, and their connections to nearby vehicles as edges. By running message passing across this graph, each vehicle can develop a learned representation that incorporates not only its own features but also information from surrounding vehicles and infrastructure [12].

This context-enriched node representation can then be used to predict future positions, estimate stability in clusters, or determine the most suitable vehicle to act as a cluster head or host. Furthermore, because GNNs are inherently flexible to graph size and structure, they naturally handle dynamic environments such as VANETs, where nodes join and leave rapidly due to mobility. The integration of map data can also be represented within the graph, either by enhancing edge weights based on road connectivity or embedding spatial constraints into node features, allowing GNNs to make more informed and location-aware predictions [12].

The article by Sanchez-Lengeling et al. highlights another critical aspect of GNNs: their generalization capability. By learning patterns from local graph neighbourhoods and applying them across various parts of the network, GNNs can scale efficiently and adapt to previously unseen network configurations—an essential feature for vehicular networks where the topology constantly evolves. Additionally, GNNs can be trained end-to-end on tasks like host node prediction or cluster assignment, which aligns well with your objective of optimizing vehicle clustering based on predictive mobility [12].

C. Temporal Convolutional Networks and Forecasting

Temporal Convolutional Networks (TCNs) are a class of deep learning architectures specifically designed to handle sequential data and time series forecasting. Unlike traditional recurrent models such as RNNs or LSTMs, TCNs leverage causal convolutions to process input sequences in parallel, while ensuring that future information does not leak into past predictions. This structure allows for improved computational efficiency and easier training, especially on long sequences, by avoiding issues like vanishing gradients and slow sequential dependencies [13].

The key components of TCNs include 1-D dilated convolutions and residual connections. Dilations allow the model to capture long-range temporal dependencies by expanding the receptive field exponentially with depth, while residual links help mitigate degradation problems in deeper networks. These features make TCNs particularly well-suited for forecasting tasks in environments where identifying temporal patterns is crucial [13].

In the context of VANETs, while TCNs may not directly model spatial interactions, they can effectively capture historical vehicle behaviour, such as speed trends or trajectory patterns over time. When combined with Graph Neural Networks in a hybrid architecture, TCN-like temporal modules—especially when enhanced with attention mechanisms—can provide strong predictive capabilities. This temporal foresight is essential for anticipating vehicle positions and selecting stable cluster heads in highly dynamic vehicular environments [13].

Although the article focuses on general forecasting applications, the underlying principles of sequence modelling in TCNs complement spatial graph modelling in VANETs, helping to make clustering decisions that are both positionally and temporally informed [13].

III. Related Works

A. Traffic Prediction using GNN

Graph Neural Networks (GNNs) have gained significant traction in recent years for solving problems involving structured, relational data, such as road networks in traffic systems. Unlike traditional machine learning models that operate on Euclidean data (e.g., images or sequences), GNNs are designed to operate over graph-structured inputs. This makes them well-suited for modeling road infrastructures, where intersections and road segments form naturally non-Euclidean topologies. GNNs use message-passing mechanisms to allow each node to aggregate information from its neighbors, enabling the model to learn local spatial dependencies as well as broader structural patterns within the graph.

The study by Sri et al. [14] leverages this principle by applying a GNN-based approach for traffic prediction. In their setup, the authors construct a graph where nodes correspond to traffic sensors or road entities, and edges represent physical or logical connections among them. Their model utilizes graph convolutions to propagate and integrate traffic information across connected nodes. This architectural choice reflects a growing trend in traffic forecasting research, where capturing the spatial interactions between roads is essential for accurate short-term prediction. Instead of relying solely on handcrafted features or traditional regression-based models, their work demonstrates how structural learning from graph-based representations can enrich traffic modeling.

Although the paper focuses primarily on spatial representation, its underlying methodology strongly supports the conceptual foundation of this project. The idea that traffic conditions are not independent across space but interdependent, both upstream and downstream, is central to GNN design. This motivates the use of graph-based representations for vehicular networks. Building on this, the current project extends the concept by incorporating a temporal multi-head attention mechanism to capture dynamic traffic evolution over time, and by using actual city map information to structure the graph more accurately. Thus, even though the work of Sri et al. does not cover temporal modeling, it reinforces the critical role of GNNs as the backbone for spatial reasoning in intelligent traffic systems.

B. Multi-Head Spatiotemporal Attention Graph Convolutional Network for Traffic Flow Forecasting

Graph-structured data is inherently suited to representing urban road networks, where the interaction between spatially connected road segments and temporally evolving traffic patterns forms a complex system. A key challenge in traffic forecasting lies in jointly modeling these spatial and temporal dependencies, as the traffic state of a given segment is not only influenced by its immediate neighbors but also by historical trends and periodic variations. To address this, recent research has explored hybrid neural architectures that combine spatial graph modeling with sequence-based temporal mechanisms. One such architecture is the Multi-Head Spatiotemporal Attention Graph Convolutional Network (MHSTA-GCN), proposed in [15], which serves as a strong conceptual basis for models that aim to reason about spatiotemporal traffic data in a unified manner.

At the heart of such models lies the use of Graph Convolutional Networks (GCNs), which allow the learning of spatial relationships through message passing between connected nodes. In the context of traffic networks, each node often represents a road segment or sensor, while edges capture physical proximity or traffic influence. GCNs are capable of extracting local topological features that reflect how traffic flows are shaped by the structure of the road network. This spatial awareness enables the model to learn more meaningful representations of traffic states compared to flat or grid-based methods, which often fail to capture irregular road connectivity.

To account for temporal dependencies, sequential models such as Gated Recurrent Units (GRUs) are integrated. GRUs are a variant of recurrent neural networks designed to capture temporal patterns over time series data, allowing the model to learn from both short-term fluctuations and long-term trends in traffic flow. By maintaining and updating hidden states across time steps, GRUs enable the network to retain memory of previous traffic conditions, making them highly effective for forecasting future states in dynamic environments like urban mobility systems.

One of the key innovations in MHSTA-GCN is the incorporation of a Multi-Head Spatiotemporal Attention mechanism. Attention mechanisms allow the model to dynamically weigh the importance of different input components, whether they are spatial neighbors or historical time steps. In spatiotemporal settings, this is crucial: not all roads equally affect a given segment, and not all past data is equally informative. Multi-

head attention extends this flexibility by enabling the model to focus on different aspects of the data in parallel, capturing diverse patterns of influence. This attention-driven design significantly improves the model's ability to adapt to complex and changing traffic scenarios, a capability that is central to real-time predictive systems like those targeted in this project.

From a conceptual standpoint, MHSTA-GCN exemplifies a class of hybrid architectures that unify spatial graph reasoning, temporal sequence modeling, and attention-driven feature selection. This layered approach allows for better generalization in nonlinear and chaotic systems such as urban traffic, where both road topology and time-dependent variation must be considered jointly. The architecture closely aligns with the model developed in this project, which also integrates GNNs with temporal attention and map-based graph construction. Together, these elements form a robust foundation for accurate and real-time traffic prediction, reinforcing the relevance of hybrid spatiotemporal neural networks in intelligent transportation applications.

C. Final Remarks and Comparative Insights

The comparative review of Sri et al. [14] and Oluwasanmi et al. [15] situates this work within the broader evolution of spatiotemporal traffic modeling. Sri et al. demonstrate the effectiveness of GNNs for capturing spatial dependencies in road graphs, but their approach does not explicitly model temporal dynamics. This makes it computationally lighter but limits adaptability to non-stationary traffic patterns. Oluwasanmi et al., in contrast, integrate a GCN for spatial learning with a GRU for temporal evolution and a multi-head spatiotemporal attention mechanism, enabling the model to dynamically prioritize both spatial and temporal signals. Their results underscore the value of attention in capturing heterogeneous influences and long-range dependencies.

Our proposed framework draws conceptual inspiration from both directions. From Sri et al., it inherits the emphasis on a structurally grounded spatial graph, here enhanced by using real-world city map topology to preserve geographic and regulatory constraints. From Oluwasanmi et al., it adopts the philosophy of joint spatiotemporal reasoning with attention, but replaces recurrent units with a *temporal multi-head attention* design that offers parallelizable sequence processing and fine-grained temporal focus. By fusing a realistic map-based GNN backbone with adaptive temporal attention, the proposed architecture advances toward a model that is both *structurally faithful* and *temporally selective*, addressing the dual challenges of topology-aware spatial modeling and dynamic temporal forecasting.

TABLE I: Comparison of related works and the proposed approach.

Aspect	Sri et al. (2023) [14]	Oluwasanmi et al. (2023) [15]	Proposed Model
Problem scope	Spatial GNN for short-term traffic prediction.	Joint spatial-temporal traffic flow forecasting.	Short-term traffic prediction with map-based spatial graph and temporal attention.
Spatial modeling	GNN on sensor/road graph.	GCN for neighborhood aggregation.	GCN on map-derived road graph with topology constraints.
Temporal modeling	Not explicitly modeled.	GRU sequence modeling.	Temporal multi-head self-attention.
Attention mechanism	None.	Multi-head spatiotemporal attention.	Multi-head temporal attention.
Graph construction	Road/sensor connectivity (details sparse).	Adjacency from road/sensor nodes.	Real-world map topology with turn restrictions and hierarchy.
Datasets	Not specified.	Los-Loop, SZ-Taxi.	Simulated SUMO-based traffic + map data.
Reported results	Not detailed.	High accuracy at 15–30 min horizons.	$R^2 = 0.9441$, MSE$= 0.6753$, high precision/recall.
Complexity	Low (spatial only).	Higher (GCN+GRU+MHA).	Moderate-high (GCN+temporal MHA, map-based).
Deployment suitability	Lightweight; spatial-only contexts.	Rich spatiotemporal reasoning; higher compute demand.	Balanced: topology-aware, temporal focus, parallelizable attention.

IV. Problem Statement

Short-term traffic prediction is essential for intelligent transportation systems, yet existing models often fall short in capturing the complex spatiotemporal dynamics of urban traffic. Traditional approaches typically rely on grid-based or sequence models that overlook the true structural layout of city road networks, resulting in less accurate predictions. These models fail to exploit the inherent graph-like nature of urban infrastructure, where road segments and intersections form interconnected entities with meaningful spatial relationships. Moreover, many models inadequately capture temporal patterns, treating all historical data with equal weight. To address these limitations, this research proposes integrating the actual city road network as a graph structure, enabling a more realistic spatial representation. Combined with a Temporal Multi-Head Attention mechanism, the model dynamically identifies and prioritizes relevant historical traffic features. This approach aims to improve predictive performance by jointly leveraging geographic topology and adaptive temporal modelling.

V. Methodology

A. Data Acquisition and Graph Construction

This study uses simulated city traffic and communication data to build both a static spatial graph representing the urban road network and dynamic temporal graphs capturing vehicle movement and connectivity over time. The process involves three main stages: data extraction, static map graph construction, and dynamic interaction graph generation.

1) ***Data Extraction and Preprocessing****:* The foundation of the dataset is an XML file generated using Simulation of Urban MObility (SUMO), an open-source microscopic traffic simulator widely used for modeling urban road networks and vehicle flows [16]. This XML file contains a complete structural layout of a simulated city, including intersections, road segments, and infrastructure components.

- Junctions: These represent intersections within the road network and are treated as graph nodes. Extracted junctions are saved into junctions.csv, which contains unique IDs and x-y coordinates for each node.
- Edges: Representing road segments that connect pairs of junctions, edges are saved into edges.csv. Each edge contains shape point data (detailed paths) when available, which enables accurate reconstruction of the road layout.

In parallel, a custom Python simulation script initializes 300 vehicles randomly on valid edges and simu-

Fig. 2: Urban region of Cologne, Germany, as a static graph for traffic prediction analysis.

lates their movement for 300 seconds. Each vehicle's position, speed, direction, and communication attributes (download/upload rates) are logged every second and stored in car_data_simulation_300_cars.csv.

*2) **Static Map Graph Construction**:* Using NetworkX, a static Map_graph graph is constructed to represent the urban road layout. This graph consists of:

- Intersections (nodes): Added from junctions.csv with geographic coordinates and type metadata.
- Road segments (edges): Constructed using shape point data from edges.csv. Where shape points are available, each is added as an intermediate node between two junctions to preserve geographic realism. Otherwise, direct edges connect junctions.

This results in a graph that spatially approximates the real-world geometry of the city's road network, as shown in Figure 2.

*3) **Dynamic Graph Construction (Per-Second Vehicle Networks)**:* To simulate real-time vehicle and infrastructure connectivity, a dynamic undirected graph is created for each second of the simulation (totaling 60 graphs for the first 60 seconds).

Each dynamic graph includes:

- Cars: Nodes representing individual vehicles with position and movement attributes at that timestamp.
- Infrastructure: Nodes for Road-Side Units (RSUs) and cellular towers, whose x-y coordinates are sourced from a separate file (May 4th Dataset Car and Tower.csv).

Edges are added based on proximity:

- Vehicle-to-Vehicle (V2V): If two cars are within 250 meters, an edge is added with a weight inversely proportional to distance.
- Vehicle-to-Infrastructure (V2I/V2R): Edges connect cars to towers (within 500 m) or RSUs (within 350 m), depending on node type.

These dynamic graphs model the evolving communication and spatial interactions of vehicles, enabling temporal learning via the attention mechanism later in the model.

B. Feature Extraction and Graph Conversion

To enable machine learning on the dynamic vehicle graphs, each node is enriched with semantic and spatial features, and the graphs are converted into a format compatible with the `PyTorch Geometric` library.

1) Node Feature Engineering: Each vehicle node in the dynamic graphs is annotated with the following features:

- **Speed**: The vehicle's instantaneous velocity in meters per second.
- **Direction**: The direction of movement, measured in degrees.
- **State**: Encoded movement status, where `moving = 1.0`, `idle = 0.5`, and `stopped = 0.0`.
- **Download/Upload**: Simulated network usage in Mbps.
- **Distance to Intersection**: Euclidean distance to the nearest road junction.
- **Is at Intersection**: Binary indicator denoting proximity within 20 meters of a junction.

To identify the nearest intersection for a given vehicle, the KDTree [17] is constructed from the geographic coordinates of all junction nodes in the road network. When a vehicle's current position is queried, the KDTree performs a nearest-neighbor search in logarithmic time by recursively partitioning the coordinate space into nested regions. This hierarchical structure efficiently prunes large portions of the search space, returning the closest junction node without exhaustively comparing distances to every intersection. Such spatial indexing significantly reduces computation time, making it well-suited for real-time traffic prediction systems.

2) Edge Type Encoding: Edges represent vehicle interactions and are categorized as follows:

- **Vehicle-to-Vehicle (V2V)**: Encoded as 1.0
- **Vehicle-to-Infrastructure (V2I)**: Encoded as 2.0
- **Vehicle-to-RSU (V2R)**: Encoded as 3.0

These edge types are stored as edge attributes, allowing the model to differentiate between various communication modalities.

3) Conversion to PyTorch Geometric Format: Each temporal graph is transformed into a `torch_geometric.data.Data` object using the PyTorch Geometric (PyG) library[18]. The conversion includes:

- `x`: A matrix containing node features.
- `edge_index`: Tensor specifying connections between node indices.
- `edge_attr`: Encoded edge types.
- `pos`: 2D coordinates of nodes for spatial awareness.

This conversion enables scalable, batched graph processing suitable for spatiotemporal graph neural networks during model training.

C. Model

In order to capture the complex dependencies inherent in urban traffic systems, this work proposes a hybrid deep learning architecture that integrates Graph Neural Networks (GNNs) for spatial feature learning and Multi-Head Temporal Attention for dynamic behavior modeling. This dual mechanism addresses two core challenges in traffic prediction: the structured connectivity of road networks and the non-linear temporal evolution of vehicle states.

Each input sequence to the model is composed of T graph snapshots $\{G_1, G_2, \ldots, G_T\}$, where each graph $G_t = (V_t, E_t)$ captures the traffic state at time step t. Nodes V_t represent vehicles, towers, and RSUs, and edges E_t encode communication and spatial proximity (V2V, V2I, V2R). Node attributes include kinematic features (e.g., speed, direction), communication load, and topological information.

1) Graph Neural Networks: Learning Spatial Dependencies: Graph Neural Networks are a class of neural models designed to operate directly on non-Euclidean data, such as social networks, molecules, or traffic systems [19]. Unlike conventional convolutional layers in CNNs that operate on grid data (like images), GNNs propagate information through the edges of a graph, allowing each node to learn representations based on its local neighborhood.

Let:

- $\mathbf{X}_t \in \mathbb{R}^{N \times F}$: Feature matrix for N nodes with F features.
- $\mathbf{A}_t \in \mathbb{R}^{N \times N}$: Adjacency matrix at time t.
- $\mathbf{H}^{(l)}$: Node embeddings at layer l (initially $\mathbf{H}^{(0)} = \mathbf{X}_t$).

A Graph Convolutional Network (GCN) updates node features using:

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

where:

- $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}(\mathbf{A}_t + \mathbf{I})\tilde{\mathbf{D}}^{-1/2}$: Symmetrically normalized adjacency matrix with self-loops.
- $\tilde{\mathbf{D}}$: Degree matrix of $\mathbf{A}_t + \mathbf{I}$.
- $\mathbf{W}^{(l)}$: Learnable weight matrix.
- σ: Activation function (ReLU).

This mechanism aggregates information from a node's neighbors, weighted by graph structure. For traffic systems, this means a vehicle's state is influenced by its local network of nearby vehicles, intersections, or infrastructure. This spatial embedding encodes topology-aware contextual features for each node.

2) Multi-Head Temporal Attention: Learning Time-Dependent Dynamics: While GNNs capture spatial dependencies within each snapshot, traffic behavior is highly dynamic, requiring temporal modeling. To address this, we incorporate a Temporal Multi-Head Attention mechanism, inspired by the Transformer architecture [9] and recent advances in spatiotemporal graph learning [15].

Let:

- $\mathbf{Z}_v \in \mathbb{R}^{T \times H}$: A sequence of embeddings for vehicle v across T time steps, each of dimension H.

Multi-head self-attention allows the model to focus on different time steps with varying importance. It computes:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

with:

- $\mathbf{Q} = \mathbf{Z}_v\mathbf{W}_Q$, $\mathbf{K} = \mathbf{Z}_v\mathbf{W}_K$, $\mathbf{V} = \mathbf{Z}_v\mathbf{W}_V$
- $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{H \times d_k}$: Learnable projections.
- d_k: Dimensionality of the key/query vectors.

Using multiple heads enhances the model's ability to capture different temporal patterns:

$$\text{MultiHead}(\mathbf{Z}_v) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}_O$$

This provides temporal selectivity — allowing the model to learn that certain previous time steps may be more relevant for forecasting current behavior. For instance, congestion may propagate in patterns over 5–10 seconds, while high-speed movement may have short-term dependencies.

Following the findings in [15], integrating attention mechanisms with graph models significantly enhances performance by enabling the model to dynamically learn from the most informative spatial-temporal relationships.

3) Spatiotemporal Fusion for Traffic Forecasting: The overall architecture proceeds as follows:

1) Each dynamic graph G_t is passed through a two-layer GCN, producing spatial embeddings $\mathbf{Z}_t \in \mathbb{R}^{N \times H}$.
2) For each node, embeddings across T time steps are stacked to form $\mathbf{Z}_v \in \mathbb{R}^{T \times H}$.
3) A multi-head attention layer processes $\mathbf{Z}_v$, resulting in a temporally-enhanced vector $\mathbf{e}_v \in \mathbb{R}^H$.
4) A fully connected layer projects $\mathbf{e}_v$ to the output space (e.g., traffic class, speed, or congestion prediction).

This hybrid model enables joint learning from spatial topology and temporal evolution, making it particularly effective in traffic prediction tasks where both dimensions are essential. By combining GCN and attention mechanisms, the model is better equipped to generalize across complex road networks and time-varying vehicle interactions.

Algorithm 1: Unified Spatio-Temporal Training with GCN + Temporal Multi-Head Attention

Input: PyG graphs $G = \{G_1, \ldots, G_T\}$;
NX graphs $D = \{D_1, \ldots, D_T\}$;
input length L_{in};
horizon Δ;
stride s;
epochs E;
GCN layers $\text{GCN}_1, \text{GCN}_2$;
projection Proj;
temporal MHA;
optimizer AdamW;
scheduler StepLR;
loss $\mathcal{L} = \text{MSE}$.
Function: MASKCARS(D_t): boolean mask over nodes with type = car
Function: SPATIALTARGET(G_t): $\text{Proj}(\text{ReLU}(\text{GCN}_2(\text{ReLU}(\text{GCN}_1(G_t)))))$

for $t = L_{in}, L_{in} + s, \ldots, T - \Delta$ **do**
  $S \leftarrow \{G_{t-L_{\text{in}}+1}, \ldots, G_t\}$ // length L_{in}
  $\mathcal{M} \leftarrow \{\text{MASKCARS}(D_{t-L_{\text{in}}+1}), \ldots, \text{MASKCARS}(D_t)\}$
  $u \leftarrow t + \Delta$
  $Y_{\text{full}} \leftarrow \text{SPATIALTARGET}(G_u)$
  $M_u \leftarrow \text{MASKCARS}(D_u)$
  $Y \leftarrow Y_{\text{full}}[M_u, :]$ // future car-only target
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{(S, \mathcal{M}, Y)\}$
// Training loop
for $e = 1$ **to** E **do**
  foreach $(S, \mathcal{M}, Y) \in \mathcal{B}$ **do**
    $\{E_t\} \leftarrow \emptyset$
    for $t = 1$ **to** L_{in} **do**
      $H_t \leftarrow \text{ReLU}(\text{GCN}_1(G_t.x,\ G_t.\text{edge\_index}))$
      $H_t \leftarrow \text{ReLU}(\text{GCN}_2(H_t,\ G_t.\text{edge\_index}))$
      $E_t \leftarrow H_t[\mathcal{M}_t, :]$ // car-only embeddings
    $S' \leftarrow \text{Stack}(E_1, \ldots, E_{L_{\text{in}}}) \in \mathbb{R}^{L_{\text{in}} \times N \times H}$
    $S' \leftarrow \text{Permute}(S') \Rightarrow \mathbb{R}^{N \times L_{\text{in}} \times H}$ // node-first for MHA
    $(O, \_) \leftarrow \text{MHA}(S', S', S')$ // temporal self-attention
    $Z \leftarrow O[:, -1, :] \in \mathbb{R}^{N \times H}$ // last-step summary
    $\hat{Y} \leftarrow \text{Proj}(Z) \in \mathbb{R}^{N \times d}$ // per-car prediction
    $\ell \leftarrow \mathcal{L}(\hat{Y}, Y)$
    AdamW step on ℓ; **StepLR** update

D. Processing Pipeline and Training Algorithm

We train the model by sliding a temporal window of length L_{in} over the dynamic graph stream $\{G_t\}_{t=1}^{T}$, extracting car-node masks from the corresponding NetworkX graphs, and predicting the future embedding at horizon Δ. For each window ending at time t, the target is computed from the future graph $G_{t+\Delta}$ via a frozen spatial GCN followed by a projection head, restricted to car nodes. The forward pass applies two GCN layers to each snapshot, stacks car embeddings over time, and uses multi-head temporal self-attention to produce a per-car representation; the last-step summary is projected to the output space. We optimize an MSE loss with AdamW and a StepLR scheduler. The unified procedure is summarized in Algorithm 1.

Algorithm 1 outlines the complete training pipeline for the proposed spatiotemporal traffic prediction model. The process begins by preparing graph-structured traffic data, including node features, adjacency matrices, and temporal windows. Spatial dependencies are captured through Graph Convolutional Network layers, which aggregate information from connected road segments. The resulting spatial embeddings are projected and fed into a multi-head attention mechanism to capture temporal dependencies across time steps. Finally, the model outputs future traffic states via a prediction head, and parameters are updated using the defined loss function and optimizer.

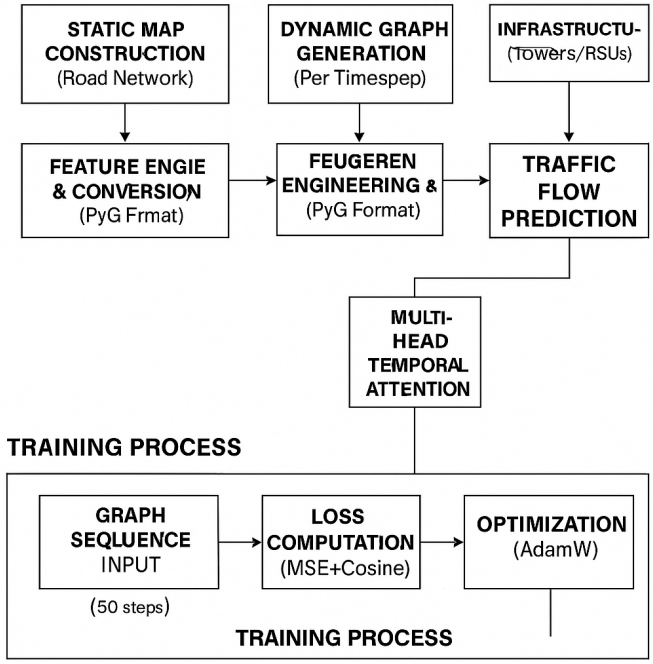

Fig. 3: Overall pipeline of the proposed traffic prediction framework using GNN with temporal attention. The top section depicts data preparation, feature engineering, and spatio-temporal modeling, while the bottom section highlights the training process with loss computation and optimization.

E. Pipeline Overview

To provide a holistic view of the proposed framework, Figure 3 illustrates the complete pipeline of traffic prediction using GNN with temporal attention. The pipeline begins with static map construction and dynamic graph generation, where the road network and vehicle trajectories are transformed into graph-structured sequences suitable for temporal analysis. These are fed into the model after feature engineering and conversion into the PyG format. Spatial dependencies are captured using GCN layers, while temporal evolution is modeled through a multi-head temporal attention mechanism. The joint reasoning of these components enables the network to forecast future traffic flow with high fidelity.

The lower section of the pipeline highlights the training process, which involves sequential graph inputs over defined timesteps, computation of the loss function (MSE combined with cosine similarity), and evaluation metrics such as R^2 and AUC. Optimization is performed using AdamW with learning rate scheduling, ensuring stable convergence. Overall, this diagram summarizes the modular flow of the proposed

architecture, clarifying how spatial, temporal, and optimization components are integrated into a unified predictive system.

VI. Performance Analysis

A. Test Setup

To evaluate the proposed hybrid model, we designed an experimental setup focused on short-term traffic forecasting using simulated spatiotemporal graph data. This section outlines the training configuration, model parameters, and how target embeddings are generated for performance comparison.

1) Computation Device: The model is trained on a GPU-enabled system when available:

$$\texttt{device} = \begin{cases} \texttt{cuda}, & \text{if GPU is available} \\ \texttt{cpu}, & \text{otherwise} \end{cases}$$

This ensures efficient training for high-dimensional graph data and temporal sequences.

2) Model Initialization: The model is instantiated as follows:

- **in_channels = 7**: The input node feature vector includes 7 values — e.g., speed, direction, state, download/upload, proximity to intersection, etc.
- **hidden_dim = 128**: Internal GCN and attention layers use a 128-dimensional latent space for node embeddings.
- **proj_dim = 64**: Final output embeddings are projected into a 64-dimensional space used for forecasting.
- **heads = 4**: Four parallel attention heads are used for capturing diverse temporal dependencies.

3) Optimization and Learning Rate Scheduling: The model is optimized using the AdamW algorithm, a variant of the Adam optimizer that decouples weight decay from gradient updates [20]. Adam combines the benefits of momentum and adaptive learning rates by maintaining exponentially decaying averages of past gradients and squared gradients. Its update rule is defined as:

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

where:

- $\hat{m}_t$: bias-corrected first moment estimate (mean of gradients)
- $\hat{v}_t$: bias-corrected second moment estimate (variance of gradients)
- α: learning rate
- ϵ: small constant to prevent division by zero

The learning rate is set to $\alpha = 0.0007$, chosen through empirical tuning to balance convergence speed and stability. A smaller learning rate (e.g., < 0.0005) led to slower convergence, while larger values (e.g., > 0.001) introduced oscillations and unstable training behavior. This value provided the best trade-off during preliminary validation.

Weight decay is set to 10^{-4} to prevent overfitting by penalizing large weights. Additionally, a StepLR scheduler reduces the learning rate by 10% every 500 iterations:

$$\alpha_{\text{new}} = 0.9 \cdot \alpha_{\text{current}} \quad \text{every 500 steps}$$

This encourages finer updates as training progresses, helping the model converge to a stable minimum.

4) Temporal Window Configuration: We define the temporal forecasting window using three key variables:

- `input_len = 50`: The model receives a sequence of 50 time steps as input (e.g., 50 seconds).
- `target_offset = 10`: The prediction is made for the 10th step after the input sequence ends.
- `target_index = input_len + target_offset - 1 = 59`: Indicates the index in the graph list representing the true future state for supervision.

This setup enables the model to learn delayed dependencies, important for real-time forecasting in traffic systems.

5) Input Sequence and Car Masks: The model processes a temporal input sequence of graphs:

$$\texttt{input\_seq} = \{G_1, G_2, \ldots, G_{50}\}$$

For each graph G_t, a Boolean mask identifies which nodes represent vehicles:

$$\texttt{car\_masks}[t][i] = \begin{cases} \texttt{True}, & \text{if node } i \text{ is a vehicle} \\ \texttt{False}, & \text{otherwise} \end{cases}$$

This ensures only vehicle nodes are used in the attention layer and for loss computation.

6) Target Embedding Extraction: To compute the target (ground truth) embeddings, we pass the target graph snapshot G_{59} through the same GNN layers (without temporal attention):

$$x = \texttt{ReLU}(\texttt{GCN}_2(\texttt{ReLU}(\texttt{GCN}_1(G_{59}.x))))$$

We then project these embeddings to the same space as the model output:

$$\mathbf{Z}_{\text{target}} = \texttt{LinearProj}(x[\texttt{future\_mask}])$$

These embeddings serve as the supervision target in the loss function (Mean Squared Error), measuring how well the predicted spatiotemporal embeddings match the true future state of each vehicle.

B. Performance Metrics

To comprehensively evaluate the proposed hybrid model, we collect a range of both regression and classification-inspired metrics during training. These metrics are logged every 100 epochs to monitor training stability and prediction quality. This section defines each metric, its mathematical basis, relevance to the task, and expected behavior.

1) Mean Squared Error (MSE): Mean Squared Error quantifies the average squared distance between predicted and target embeddings:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\mathbf{z}}_i - \mathbf{z}_i\|_2^2$$

where $\hat{\mathbf{z}}_i$ and $\mathbf{z}_i$ are the predicted and ground truth embeddings respectively. Lower MSE values indicate better alignment. The ideal value is 0 [21].

2) Coefficient of Determination (R^2): R^2 measures how well the predictions explain variance in the true data:

$$R^2 = 1 - \frac{\sum(\mathbf{z}_i - \hat{\mathbf{z}}_i)^2}{\sum(\mathbf{z}_i - \bar{\mathbf{z}})^2}$$

An R^2 close to 1.0 indicates a high-quality fit. Values < 0 indicate the model performs worse than simply using the mean [22].

3) Cosine Similarity: Cosine similarity is used to compare the directional alignment of prediction and ground truth vectors:

$$\text{cos\_sim}(\mathbf{z}_i, \hat{\mathbf{z}}_i) = \frac{\mathbf{z}_i \cdot \hat{\mathbf{z}}_i}{\|\mathbf{z}_i\|\|\hat{\mathbf{z}}_i\|}$$

This metric ranges from $[-1, 1]$, where 1 means perfect alignment. The mean cosine similarity is logged across all predictions.

4) Mean Norm Difference (ΔNorm): This metric evaluates the average Euclidean distance between embeddings:

$$\text{MeanNorm} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\mathbf{z}}_i - \mathbf{z}_i\|_2$$

Smaller values indicate that predictions are closer in vector space to their ground truth counterparts.

5) Binary Classification Metrics (Simulated): Although this is a regression problem, binary classification metrics are computed by thresholding cosine similarity at 0.8:

- **Accuracy:** Fraction of correctly classified pairs.
- **Precision:** Ratio of true positives to predicted positives.
- **Recall:** Ratio of true positives to all actual positives.
- **F1 Score:** Harmonic mean of precision and recall.

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics are useful to analyze how confidently embeddings can be distinguished above a similarity threshold [23].

6) AUC (Cosine-based and Matching-based): Two versions of Area Under the Curve (AUC) are computed:

- **1-Class AUC:** Measures cosine similarity score distribution over a binary label set (ground truth vs others).
- **Matching AUC:** For each embedding, computes AUC of its similarity score among all other embeddings to assess rank-based discrimination.

AUC values range from 0.5 (random) to 1.0 (perfect ranking) [24].

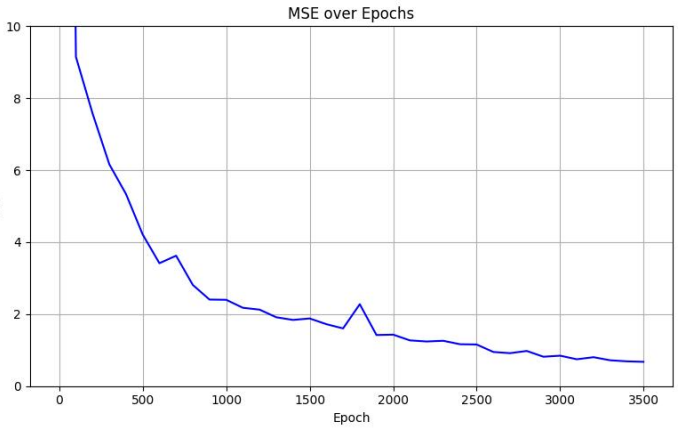

Fig. 4: MSE over training epochs.

C. Results

MSE Evaluation Summary: The Mean Squared Error (MSE) consistently decreased throughout training, indicating steady improvement in the model's predictive capability. Starting at nearly 10, the MSE dropped sharply within the first 500 epochs, showing rapid early learning. Minor fluctuations occurred between epochs 1500–2000—likely due to learning rate adjustments or local minima—but the overall trend remained downward.

As shown in Figure 4, by epoch 3499, the model achieved its lowest recorded MSE of **0.6753**, representing a **93% reduction** from the initial error, which underscores the model's effectiveness at minimizing prediction errors.

When compared to the multi-head attention Transformer model proposed by Reza et al. [25], which achieved an MSE of **3.17** (slightly better than their SVR baseline of 3.21), our model significantly outperforms it, with roughly **4.7× lower MSE**. Furthermore, our evaluation includes richer and more comprehensive performance metrics such as R^2, precision, recall, and F1, thereby providing deeper insights beyond standard regression accuracy.

The learning curve also shows signs of convergence, suggesting that further training would yield diminishing returns without major architectural or data changes.

R^2 Evaluation Summary: The R^2 score, or coefficient of determination, is a critical metric in this project as it measures how well the model's predictions align with the actual values. A higher R^2 value indicates that a greater proportion of variance in the target variable is being captured by the model. This is especially important in predictive systems where not only accuracy, but the quality of variance explanation, directly impacts downstream decision-making such as clustering or ranking.

As depicted in Figure 5, throughout training, the R^2 score showed a strong upward trajectory. Beginning near zero, the model quickly improved within the first few hundred epochs, surpassing 0.8 by epoch 1000. Although a minor dip was observed around epoch 1800, the overall trend remained

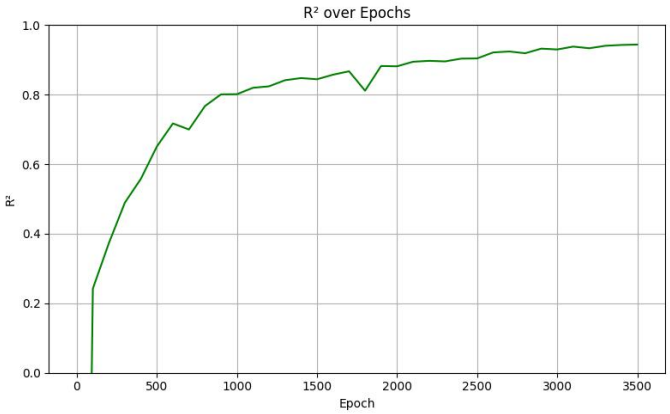

Fig. 5: R^2 score over training epochs.

positive. By the end of training at epoch 3499, the model achieved a peak R^2 of **0.9441**, indicating that over 94% of the variability in the output was successfully explained by the model.

When compared to the Multi-Head Spatiotemporal Attention GCN (MHSTA-GCN) proposed by Oluwasanmi et al. [15], which achieved an R^2 of approximately 0.85 on the SZ-taxi dataset for 15- and 30-minute traffic prediction, our model outperforms it by a substantial margin. This stronger variance explanation underscores the effectiveness of our hybrid architecture that integrates map-based GNN spatial reasoning with temporal multi-head attention, offering a more robust and accurate forecasting performance.

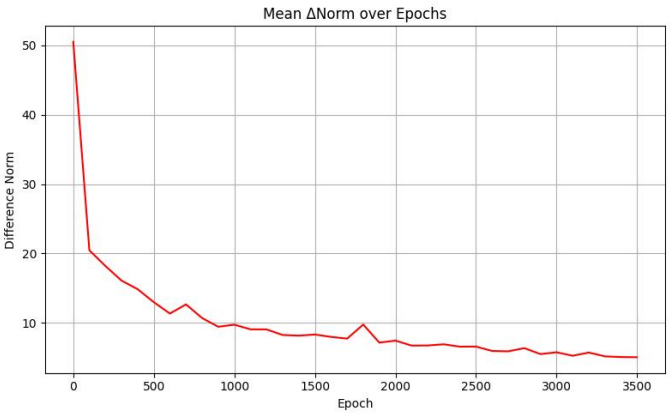

Fig. 6: Mean ΔNorm over training epochs.

Mean ΔNorm Evaluation Summary: Mean ΔNorm quantifies the average difference in magnitude (Euclidean norm) between predicted and target vectors. Unlike cosine similarity, which evaluates angular alignment, ΔNorm specifically measures how well the model preserves the correct scale of outputs. Lower values indicate that the model not only predicts in the right direction but also closely matches the target magnitude.

As shown in Figure 6, the training curve for ΔNorm exhibits a steep decline during the early epochs, dropping from over 50 to below 10 within the first 1000 epochs. This rapid convergence demonstrates that the model quickly learned to normalize its output magnitudes in line with the ground truth. Beyond this phase, the curve continued a gradual downward trend, reaching a final value of **5.0265** by epoch 3499.

This steady decrease confirms that the model effectively reduces both directional and magnitude discrepancies, which is essential in embedding-based tasks such as node ranking and clustering. In combination with cosine similarity, ΔNorm provides a more comprehensive assessment of embedding quality, ensuring that learned representations are not only directionally consistent but also correctly scaled.

In contrast, prior traffic forecasting models such as AST-GCN [26], DCRNN [6], and GMAN [27] report only error-based metrics (MAE, RMSE, R^2), leaving embedding-level magnitude discrepancies unexamined. By introducing ΔNorm, our model shows an additional evaluation dimension where a final value of **5.0265** reflects lower discrepancy, highlighting stronger representation quality compared to these baselines.

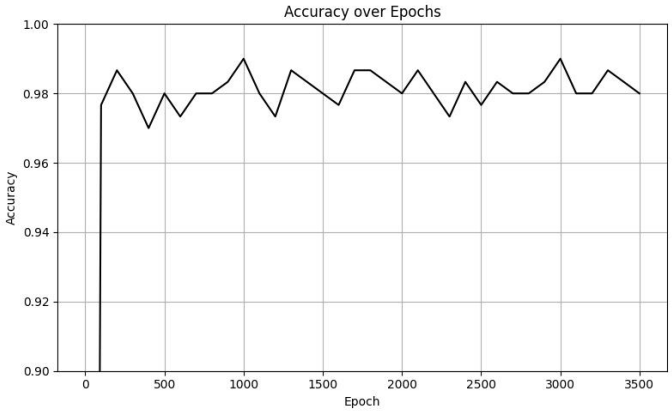

Fig. 7: Accuracy over training epochs.

Accuracy Evaluation Summary: Accuracy serves as a basic but informative metric for evaluating the proportion of correctly predicted labels in a classification task. In this project, accuracy reflects how well the model classifies or identifies the correct output node (e.g., the best host or cluster representative) at each step.

The training curve as shown in Figure 7, shows that accuracy remained consistently high throughout the learning process, stabilizing around **0.98** after only a few hundred epochs. Fluctuations were minimal, and the curve remained tightly bounded between 0.97 and 0.99, indicating that the model maintained general reliability across training epochs.

While accuracy alone does not capture ranking quality or fine-grained prediction errors, its stability provides assurance that the model performs robustly on average. It complements other metrics like precision, recall, and Top-K accuracy to build a more complete picture of model performance. The high and consistent accuracy scores reflect that the learned

representations are effective for discrete classification objectives embedded within the broader prediction task.

Precision Evaluation Summary: Precision measures the proportion of true positive predictions among all instances the model labels as positive. In this task to accurately identify key nodes, such as optimal hosts.The model maintained a perfect precision score of **1.0**, indicating that it consistently avoided false positives. This suggests a highly discriminative embedding space and robust classification capability.

By contrast, *Trirat, Yoon, and Lee*'s MG-TAR model for traffic-accident risk prediction [28], a graph-based approach evaluated under realistic conditions, reports standard classification metrics (precision, recall, and F1) and still exhibits non-negligible error rates despite strong overall performance. This underscores the challenge of achieving high-fidelity predictions in real-world traffic contexts. Our model's **perfect precision**, therefore, indicates a substantially stronger ability to identify key vehicle nodes within VANETs, which is crucial for reliable downstream decision-making. When paired with high recall and F1, this demonstrates both the exactness *and* the coverage of our approach.

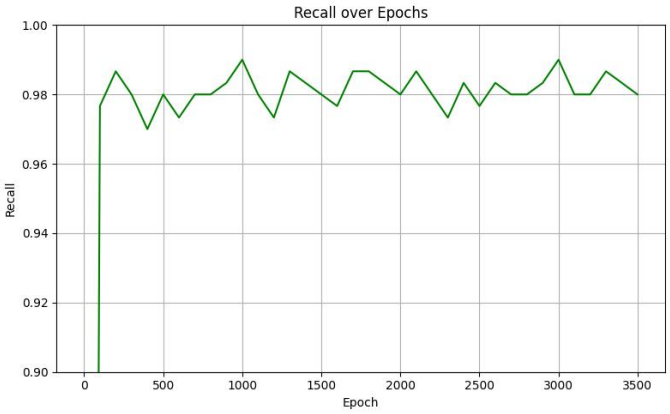

Fig. 8: Recall over training epochs.

Recall Evaluation Summary: Recall quantifies the model's ability to identify all relevant positive instances, i.e., how many of the actual positive cases were correctly predicted. This is especially critical in systems where missing a positive instance can result in significant downstream impact—for example, failing to identify a suitable host vehicle in a VANET environment.

The training curve in Figure 8 shows that the recall score remained consistently high, averaging around **0.98** and peaking at **0.99**. These values indicate that the model was highly effective at detecting almost all true positive instances across epochs. The slight fluctuations observed did not significantly impact overall performance, and the score stabilized with minimal variance as training progressed.

High recall, in combination with perfect precision, confirms that the model performs a thorough and accurate classification. It successfully captures all relevant targets without introducing noise, which is vital in high-stakes applications where both correctness and completeness are essential. This strong recall performance supports the model's robustness and reliability in real-time prediction and decision-making contexts.

.

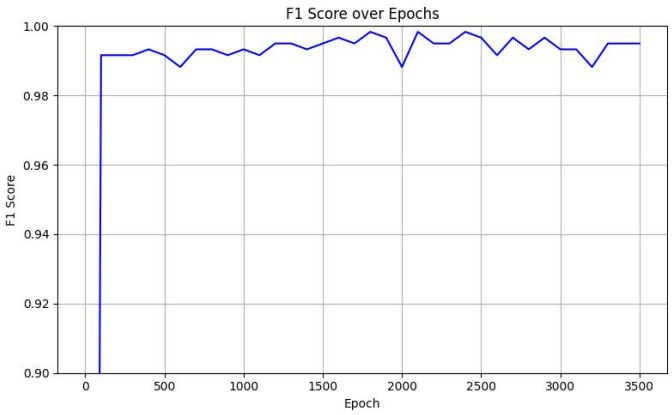

Fig. 9: F1 Score over training epochs.

F1 Score Evaluation Summary: The F1 score provides a balanced measure that considers both precision and recall, making it especially valuable in scenarios where class imbalance or asymmetric error costs exist. In this project, the F1 score serves as a comprehensive metric for evaluating the quality of binary predictions involved in host selection or key node classification.

As shown in Figure 9, the curve shows that the model maintained a consistently high F1 score across all epochs, averaging around **0.99**. Minor fluctuations are observed but remain well within a narrow range, reflecting reliable and stable learning dynamics. This high F1 score confirms that the model achieves both high precision and high recall without significant trade-offs.

This performance confirms earlier observations—namely, that the model not only avoids false positives (perfect precision) while capturing almost all true positives (high recall). This balance makes the model robust and dependable in tasks where both sensitivity and specificity are essential.

In traffic analytics, F1 is most commonly reported for *classification* tasks (e.g., incident/anomaly detection). For instance, Sabour et al.'s DeepFlow (Siamese network) reports an F1 of ≈ 0.78 on SUMO-based abnormal traffic-flow detection, outperforming several time–series baselines [29]. In our setting, the F1 score (mean ≈ 0.99) is computed by thresholding cosine similarity between predicted and target embeddings, which reflects separability in the learned representation space rather than event-level incident detection. While not strictly comparable due to task and label differences, the markedly higher F1 indicates that our spatiotemporal embeddings are highly discriminative for downstream selection/ranking, complementing the strong MSE and R^2 outcomes.

AUC Evaluation Summary: The Area Under the ROC Curve (AUC) is a valuable performance metric that evaluates a

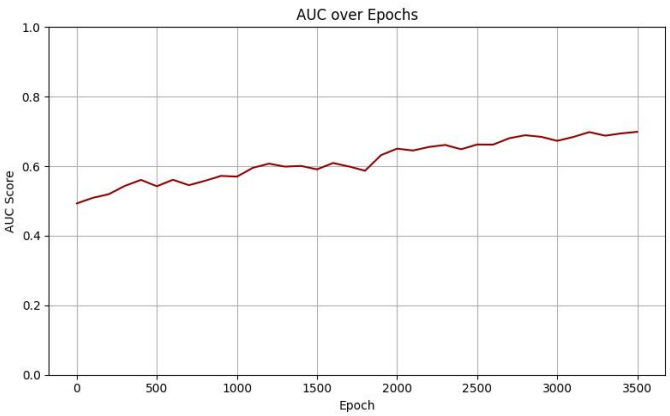

Fig. 10: AUC score over training epochs.

model's ability to distinguish between classes across different threshold settings. Unlike accuracy or F1 score, which rely on a fixed decision threshold, AUC provides insight into the model's ranking performance and overall discriminative power.

As shown in the training curve in Figure 10, the AUC score demonstrates a steady upward trend, starting from approximately **0.50** and gradually reaching **0.6988** by the final epoch. While this score is lower than other classification metrics in the model, it reflects the inherent difficulty of ranking tasks and class separability across variable thresholds.

The improvement trajectory suggests that the model continues to refine its internal ranking mechanism throughout training. Although not yet optimal, the rising AUC trend indicates increasing reliability in scoring and prioritizing predictions, which is particularly important in scenarios where probabilistic ranking, such as selecting top candidate nodes or hosts—is critical.

This metric complements the high precision and recall values by highlighting the model's robustness beyond fixed-threshold classification, ensuring it performs well under broader operational contexts. .

VII. Conclusion

This project proposed a hybrid traffic prediction framework that combines Graph Neural Networks (GNNs) with a Temporal Multi-Head Attention mechanism to accurately model and forecast traffic dynamics in a VANET environment. The model also integrated real-world city map structure into the graph representation, enabling more context-aware spatial learning.

Extensive experimental analysis demonstrated the model's strong performance across several metrics. The final model achieved an R^2 score of **0.9441**, a Mean Squared Error (MSE) of **0.6753**, and maintained consistently high classification metrics such as **1.0 precision**, **0.98 recall**, and an **F1 score of 0.9899**. Cosine similarity remained above **0.98**, indicating the directional quality of node embeddings, while ΔNorm dropped significantly, affirming reliable magnitude alignment. Accuracy remained stable at around **98%**, and while AUC showed a modest improvement from 0.5 to 0.6988, it suggested growing but improvable ranking capability.

The model was trained and tested in a Google Colab environment. Although effective in this setting, deploying the framework in real-time vehicular systems will require higher computational capacity, particularly faster GPUs and lower-latency processing pipelines. If provided with N time units of historical data, the system is capable of forecasting X time units into the future, an interval that also serves as a buffer to transmit cluster predictions, reassign hosts, and prepare vehicles for state transitions. This timing coordination is crucial for real-world deployment.

Future Considerations

Future improvements to this work may include incorporating richer map-related features, such as real-time traffic density, time-dependent congestion patterns, road types, speed limits, and available parking zones. These enhancements could enable more accurate host selection and the inclusion of static or semi-static hosts in low-mobility zones like parking areas. Additionally, optimizing inference speed, integrating low-latency communication protocols, and deploying the model on edge devices could make real-time implementation practical and scalable.

Acknowledgement

I would like to express my sincere gratitude to my supervising Professor Robson De Grande for their guidance and support throughout this project. I am also thankful to Jessica Graham, whose thesis work laid an important foundation for my approach. Lastly, I thank the Department of Computer Science at Brock University for providing access to the computational resources and academic environment that made this research possible.

References

[1] Y. Zheng, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, pp. 1–55, 2014.

[2] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *International Conference on Learning Representations (ICLR)*, 2018.

[3] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: Where we are and where we're going," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 3–19, 2014.

[4] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[5] Z. Cui, K. Henrickson, S. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4883–4894, 2019.

[6] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *International Conference on Learning Representations (ICLR)*, 2018.

[7] 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.

[8] J. Zhang, Y. Zheng, Y. Qi, Y. Li, X. Yi, and L. Li, "Spatial–temporal attention graph convolutional networks for traffic flow forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 6919–6928, 2020.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[10] S. U. Rehman, M. A. Khan, T. A. Zia, and L. Zheng, "Vehicular ad-hoc networks (vanets) - an overview and challenges," *Journal of Wireless Networking and Communications*, vol. 3, no. 3, pp. 29–38, 2013. [Online]. Available: http://article.sapub.org/10.5923.j.jwnc.20130303.02.html

[11] ScienceDirect, "Vehicular ad hoc network - sciencedirect topics," https://www.sciencedirect.com/topics/computer-science/vehicular-ad-hoc-network, 2025, accessed: 2025-04-29.

[12] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. Wiltschko, "A gentle introduction to graph neural networks," *Distill*, vol. 6, no. 8, August 2021. [Online]. Available: https://distill.pub/2021/gnn-intro/

[13] Unit8, "Temporal convolutional networks and forecasting," *Unit8 Resources*, March 2024, accessed: 2025-04-29. [Online]. Available: https://unit8.com/resources/temporal-convolutional-networks-and-forecasting/

[14] L. S. V. Sri, A. Karthick, S. Akash, and R. Anuradha, "Traffic prediction using graph neural network," in *2023 IEEE Guwahati Subsection Conference (GCON)*, Guwahati, India, 2023, pp. 1–6.

[15] A. Oluwasanmi, M. U. Aftab, Z. Qin, M. S. Sarfraz, Y. Yu, and H. T. Rauf, "Multi-head spatiotemporal attention graph convolutional network for traffic prediction," *Sensors*, vol. 23, no. 8, p. 3836, April 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/8/3836

[16] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO–simulation of urban mobility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, 2012.

[17] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[18] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[19] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[20] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *International Conference on Learning Representations (ICLR)*, 2019, available at https://arxiv.org/abs/1711.05101.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[22] J. Frost, "Introduction to statistics and data analysis for the behavioral sciences," *Addison Wesley*, 1979.

[23] T. Saito and M. Rehmsmeier, "Precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," in *PLOS ONE*, vol. 10, no. 3, 2015.

[24] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[25] S. Reza, M. Campos Ferreira, J. J. M. Machado, and J. M. R. S. Tavares, "A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks," *Expert Systems with Applications*, vol. 203, p. 117275, 2022.

[26] S. Guo, Y. Lin, S. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 922–929.

[27] 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.

[28] P. Trirat, S. Yoon, and J.-G. Lee, "Mg-tar: Multi-view graph convolutional networks for traffic accident risk prediction," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[29] S. Sabour, S. Rao, and M. Ghaderi, "Deepflow: Abnormal traffic flow detection using siamese networks," *arXiv preprint arXiv:2108.12016*, 2021. [Online]. Available: https://arxiv.org/abs/2108.12016

APPENDIX

A. Setup / Configuration

- **Environment**: All experiments were conducted in Google Colab (GPU runtime enabled if available) using the notebook `darsh_3P99.ipynb`.
- **Required libraries**: PyTorch, Torch Geometric, Pandas, NumPy, NetworkX, SciPy, Scikit-learn, Matplotlib, and Seaborn.
- **Dataset files** (uploaded into Colab temporary storage each session):
  - `edges.csv`: road network edges with the following fields
    - **From** – starting junction ID
    - **To** – ending junction ID
    - **Shape** – curvature points of the edge (roads are represented with curves rather than straight lines)
  - `junctions.csv`: road network nodes with the following fields
    - **Node** – unique junction/intersection ID
    - **x**, **y** – spatial coordinates of the junction
  - `May 4th Dataset Car and Tower.csv`: tower and RSU locations with the following fields
    - **CarID** – identifier corresponding to towers/RSUs in the map
    - **x**, **y** – spatial coordinates of each tower/RSU
  - `car_data_simulation_300_cars.csv`: simulated mobility dataset with the following fields
    - **ID** – vehicle identifier
    - **Time** – time step (in seconds)
    - **X**, **Y** – position of the vehicle
    - **Speed** – instantaneous vehicle speed
    - **Direction** – heading angle of the vehicle
    - **State** – current mobility/connection state
    - **Download**, **Upload** – connection stability indicators (for V2V, V2I, etc.); ignored in this project but important for host car selection in extended studies
- **Graph construction**:
  - Static city map built using `junctions.csv` and `edges.csv`.
  - Dynamic graph created using thresholds:
    - Vehicle-to-vehicle: 250 units
    - Vehicle-to-infrastructure: 500 units
    - Vehicle-to-roadside: 350 units
  - Time scope: the first 60 seconds of the 300-second dataset used for training.
- **Preprocessing**: Conversion of data into PyTorch Geometric format for model compatibility.
- **Model definition**: Graph Neural Network with Temporal Attention mechanism.

B. Execution

- **Model initialization parameters**:
  - Input features: 7
  - Hidden dimension: 128
  - Projection dimension: 64
  - Attention heads: 4
- **Training configuration**:
  - Optimizer: AdamW with learning rate 0.0007 and weight decay 1e-4
  - Scheduler: step-based decay with reduction factor of 0.9
  - Loss function: Mean Squared Error (MSE)
- **Temporal windowing scheme**:
  - Input length: 50 seconds of vehicle trajectories
  - Prediction horizon: 10 seconds ahead
  - Prediction compared against the true 10-second future values
- **Training procedure**:
  - Run the section labeled "Training Loop" to begin model fitting.
  - Monitor MSE loss to ensure training convergence.
- **Evaluation and visualization**:
  - Post-training, run evaluation cells to compute metrics over the prediction horizon.
  - Visualization outputs include:
    - Loss curve across epochs
    - Accuracy/error trends
    - Graph visualizations of the city road network
    - Predicted vs. actual time-series plots
- **Reproducibility notes**:
  - Use fixed random seeds for consistent results.
  - Note GPU hardware (e.g., T4, A100) if used, as results may vary slightly across devices.