# Urban Transportation Network Optimization with Real-Time Graph Updates

## I. Executive Summary

This report presents a comprehensive framework for the "Urban Transportation Network Optimization with Real-Time Graph Updates" project, a compelling endeavor for a 4th-year Computer Science Engineering undergraduate major project. The core innovation of this initiative lies in its departure from conventional, static traffic management paradigms. Instead, it proposes treating urban transportation networks as dynamic, continuously evolving graph structures, akin to a "digital nervous system" that can perceive, learn, and adapt to real-time conditions. This foundational shift addresses critical limitations in current systems, which often suffer from "structural blindness" and a reactive posture towards congestion and inefficiencies.

The project's methodology centers on a multi-layer graph representation, allowing for the intricate modeling of diverse transportation modes and their interconnections. A pivotal component involves the development of dynamic edge weight calculations, leveraging Graph Neural Networks (GNNs) to predict travel times based on a multitude of real-time and historical factors. To manage the inherent computational complexity of real-time optimization across large networks, the framework incorporates strategies such as graph partitioning and hierarchical optimization.

The technical implementation relies on a robust stack, including Python for development, OSMnx for geographic data modeling, GraphML for data exchange, TensorFlow for machine learning, and Redis for high-performance data caching. Recognizing the specific context of an undergraduate project in India, the report provides practical adjustments, particularly regarding data acquisition challenges, by advocating for the strategic use of open-source traffic simulators for data generation and controlled experimentation.

Expected outcomes include demonstrable improvements in simulated travel times,

enhanced predictability of trip durations, and a foundational understanding of how dynamic graph optimization can contribute to smarter, more efficient, and sustainable urban environments. This project offers a significant opportunity for an undergraduate student to engage with cutting-edge research and apply advanced computational techniques to a pressing societal challenge.

## II. Project Vision and Core Innovation: Revolutionizing Urban Traffic Management

Urban transportation systems worldwide, and especially within the rapidly expanding metropolitan areas of India, are grappling with escalating challenges. These include pervasive traffic congestion, protracted commute times, and mounting environmental concerns stemming from increased fuel consumption and emissions.[1] Existing traffic management systems frequently exhibit a fundamental limitation often described as "structural blindness." They possess the capacity to measure current conditions but struggle to comprehend how these conditions will evolve or how various segments of the transportation network influence one another. This reactive nature means that problems are addressed only after they manifest, rather than being anticipated and mitigated proactively.

A single incident, such as a traffic accident during rush hour, serves as a stark illustration of this systemic vulnerability. Such an event does not merely affect the immediate road segment; its repercussions cascade throughout the entire network as drivers divert to alternative routes, leading to unforeseen congestion on secondary roads that would typically operate smoothly. Traditional systems, by treating each road segment in isolation, fail to account for these intricate, interconnected relationships that fundamentally govern real-world traffic behavior. Furthermore, many current approaches overlook the dynamic nature of road networks, assuming fixed capacities regardless of influential factors like weather, special events, or time of day. They also tend to focus on individual transportation modes, missing crucial opportunities for synergistic coordination between vehicular traffic, public transit, and alternative modes like cycling or walking.

The proposed project directly confronts these limitations by introducing a transformative approach: conceptualizing the urban transportation network as a continuously evolving graph structure. This paradigm represents a profound shift from the static, map-like view of conventional systems to a "living map" that constantly updates itself, learns from observed patterns, and predicts future conditions. This is not merely an incremental enhancement to existing systems; it embodies a fundamental change in the operational philosophy of urban traffic management. The emphasis moves from merely reacting to current traffic jams to actively predicting and preventing their formation. This necessitates the development of robust data pipelines capable of ingesting and processing real-time information, coupled with mechanisms for continuous model retraining and adaptation. The success of such a system inherently depends on its ability to capture and leverage the complex, dynamic interdependencies within the network, a capability for which Graph Neural Networks (GNNs) are particularly well-suited. Consequently, any localized optimization or routing decision must be considered within the broader context of the entire network, acknowledging that changes in one area invariably affect others.

## III. The Dynamic Transportation Graph: A Deeper Dive into Multi-Layer Representation

The foundational element of this project is the reimagining of a city's transportation infrastructure not as a simplistic, flat road map, but as a sophisticated, multi-dimensional graph structure. This multi-layer representation, also known as a multilayer network, allows for the modeling of diverse transportation modes and their intricate interconnections.[4] Each distinct mode of transport—such as the road network, public transit routes (buses, trains, metro), bicycle lanes, and pedestrian pathways—is represented as a separate "layer" within this overarching graph. This approach moves beyond the limitations of traditional unidimensional networks, which typically represent only one type of link or relationship.[4]

Within this multi-layer framework, each individual layer possesses its own unique characteristics and operational constraints. For instance, the road network layer would meticulously detail attributes pertinent to vehicular traffic, including lane capacities, speed limits, and specific turn restrictions at intersections. Conversely, the public transit layer would integrate data such as bus or train schedule frequencies, vehicle capacities, and precise station locations. Similarly, dedicated layers for bike lanes and pedestrian paths would incorporate attributes like surface quality, accessibility, and walkability scores.

The critical aspect of this multi-layer design lies in its ability to model interconnections between these disparate modes. Specific geographic points, such as bus stops, metro stations, park-and-ride facilities, or bike-sharing docks, serve as crucial "nodes" that connect these different layers. These inter-layer connections represent points where travelers can seamlessly transfer from one mode of transportation to another.[5] This detailed representation of nodes and edges, both within and across layers, is paramount for enabling the system to suggest truly multimodal routes. For example, instead of merely identifying the fastest car route, the system could recommend a journey involving driving to a transit station, utilizing a train for the main segment, and then completing the final leg with a shared bicycle, all while factoring in real-time conditions across each mode.

The granularity and heterogeneity inherent in this multi-layer graph model are essential. A simple, homogenous graph structure would be insufficient to capture the diverse data types and complex relationships present in a real-world urban transportation system. The ability to define specific attributes for nodes (e.g., intersection ID, traffic signal state for roads; station ID, schedule for public transit) and edges (e.g., length, current speed for roads; frequency, travel time for public transit) within each layer, along with the logic for inter-layer connections, is fundamental to the model's efficacy. This flexibility supports the choice of GraphML as a data exchange format, given its capacity to handle application-specific attributes and grouped graphs.[6] The careful design of this schema, potentially leveraging a property graph model, becomes a critical first step in implementation.[8]

The ambition of enabling seamless multimodal routing, while theoretically powerful, presents a complex optimization challenge, particularly for an undergraduate project. The integration of heterogeneous data sources, such as disparate timetables and real-time operational information from various transit agencies, poses a significant hurdle.[9] Therefore, while the multi-layer concept is central, a pragmatic approach for an undergraduate project might involve initially focusing on a simplified multimodal model (e.g., road network combined with one public transit line) or prioritizing the dynamic optimization of the road network with a clear roadmap for future multimodal expansion. This strategic focus manages the complexity while still demonstrating the core innovation.

The following table summarizes the key components of this multi-layer transportation graph:

**Table 1: Key Components of the Multi-Layer Transportation Graph**

| Component Type | Description | Node Attributes (Examples) | Edge Attributes (Examples) | Inter-layer Connection Logic |
|---|---|---|---|---|
| Road Network Layer | Represents vehicular roads and intersections. | Intersection ID, Coordinates, Traffic Signal State | Length, Speed Limit, Lane Count, Current Speed, Congestion Level | Connects to Public Transit/Bike/Pedestrian layers at shared points (e.g., parking facilities, bus stops). |
| Public Transit Layer | Represents bus, train, or metro routes and stations. | Station ID, Type (Bus Stop, Metro Station), Schedule, Capacity | Route ID, Frequency, Scheduled Travel Time, Real-time Delay | Connects to Road/Bike/Pedestrian layers at transit hubs. |
| Bike Lane Layer | Represents dedicated and shared cycling paths. | Bike-sharing Station ID, Available Bikes/Docks | Path Type, Surface Quality, Length, Elevation Change | Connects to Road/Public Transit/Pedestrian layers at bike hubs. |
| Pedestrian Path Layer | Represents sidewalks, walkways, and pedestrian zones. | Point of Interest, Building Entrance, Crosswalks | Walkability Score, Length, Elevation Change, Safety Rating | Connects to all other layers at points of access (e.g., building entrances, crosswalks). |
| Inter-layer Connections | Virtual links enabling transfers between different transportation modes. | N/A (represented by shared physical nodes) | Transfer Time, Transfer Cost (e.g., waiting time, walking distance), Accessibility | Facilitates multimodal journeys by linking nodes across different layers. |

## IV. Intelligent Prediction with Graph Neural Networks: Dynamic

## Edge Weight Calculation

A fundamental limitation of traditional shortest-path algorithms in urban transportation is their assumption of constant edge weights, implying that the "cost" of traversing a road segment remains uniform regardless of real-world conditions. This is patently unrealistic; a highway segment that takes five minutes to traverse at 2 AM could easily require thirty minutes during peak rush hour.[10] To overcome this, the project emphasizes the development of time-dependent edge weights that accurately reflect real-world conditions.

These dynamic edge weights are derived from sophisticated predictive models that estimate travel times based on a rich array of factors. These include historical traffic patterns, current traffic density (obtained from sensors or simulated data), prevailing weather conditions, information about scheduled events (e.g., concerts, sporting events), and even insights gleaned from social media activity that might indicate emerging issues or unusual traffic behaviors.[11] Machine learning techniques are instrumental in identifying the complex patterns and correlations among these diverse factors that collectively influence traffic flow.[13] The system is designed to continuously update these edge weights as new information becomes available. For instance, if traffic sensors detect a sudden drop in speeds on a particular road segment, the algorithm immediately recalculates the weights for that edge and propagates the impact throughout the network, ensuring a responsive system that adapts to changing conditions rather than relying on outdated assumptions.

The effective transformation of this diverse data into meaningful, real-time features that accurately represent travel cost and can be consumed by the GNN is a significant undertaking in data preprocessing and feature engineering. For an undergraduate project, this might necessitate a focused approach, perhaps prioritizing historical patterns and current traffic density, or simplifying the integration of external factors by using pre-calculated average impacts rather than real-time feeds. The computational overhead associated with continuous feature engineering must be carefully considered to maintain real-time performance.

Graph Neural Networks (GNNs) are particularly well-suited for this task due to their inherent ability to understand complex network relationships, where the connections between nodes are as crucial as the nodes themselves. GNNs learn to predict traffic flow by analyzing how congestion propagates through the network, identifying relationships that may not be immediately apparent to human observers. For example, they can discern how school dismissal times in one neighborhood consistently affect traffic patterns miles away, or how specific weather conditions influence the interplay between public transit usage and road congestion.[17] This learning capability allows the system to make increasingly accurate predictions as it processes more data, developing an understanding of the underlying dynamics that govern traffic flow, even for unusual situations it has not directly encountered before.

Common GNN architectures employed in traffic forecasting include Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs).[17] These are often combined with temporal models such as Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRUs) to effectively capture both spatial and temporal dependencies within traffic data, forming Spatio-Temporal GNNs (ST-GNNs) or Temporal Graph Convolutional Networks (T-GCNs).[17] While GNNs are powerful, they can be computationally intensive, especially for large-scale dynamic graphs.[19] The aspiration for "real-time" updates creates a tension between the desire for highly accurate, complex GNN models and the necessity for rapid inference to provide timely routing recommendations. This suggests that model selection should prioritize a balance of accuracy and efficiency, potentially requiring model simplification (e.g., fewer layers, smaller hidden dimensions) to align with undergraduate project resources. Techniques such as graph partitioning can aid in managing this computational load by breaking large graphs into smaller, more manageable sub-networks for parallel processing.[34] The strategic use of Redis for caching frequently accessed information is also a crucial architectural decision to accelerate data retrieval and maintain system responsiveness.[36]

For dynamic travel time prediction, models capable of handling both time-series data and spatial dependencies are paramount. The Kalman filtering algorithm is notable for its ability to continuously update state variables with new observations, demonstrating superior accuracy during peak hours compared to link-based data.[38] Other established data-driven methods include Support Vector Regression (SVR), K-nearest Neighbor (KNN), Autoregressive Integrated Moving Average (ARIMA), Decision Trees, and Artificial Neural Networks (ANN).[38] Deep learning models like LSTM and Transformer networks have also shown strong performance in capturing complex temporal patterns.[38] Specifically, GCNNs and T-GCNs are designed to learn traffic flow patterns within a graph context, efficiently representing the network and origin-destination demand.[23] The following table provides a comparative overview of suitable machine learning models for this project:

**Table 2: Machine Learning Models for Dynamic Travel Time Prediction**

| Model Type | Key Characteristics | Suitability for Real-time Graph Context | Computational Efficiency Notes | Relevant Snippets |
|---|---|---|---|---|
| Kalman Filter | State-space model, recursive estimation | Excellent for continuous updates of state variables (e.g., travel time) | Relatively efficient for sequential data; good for noisy data [38] | [38] |
| LSTM (Long Short-Term Memory) | Recurrent Neural Network, captures long-term temporal dependencies | Good for temporal patterns on individual links/nodes; can be integrated with GNNs for spatio-temporal modeling [38] | Can be computationally intensive for very long sequences or large networks [39] | [17] |
| GCN (Graph Convolutional Network) | Graph-based, aggregates features from neighbors | Excellent for capturing static spatial relationships in the graph [17] | Requires entire graph structure for training, can be memory-intensive for large graphs [17] | [17] |
| GAT (Graph Attention Network) | Graph-based, uses attention mechanism for weighted aggregation | Excellent for capturing dynamic spatial relationships by assigning importance scores to neighbors [17] | Can be more computationally demanding than GCNs due to attention mechanism [17] | [17] |
| T-GCN (Temporal Graph Convolutional Network) | Combines GCN with GRU/LSTM for spatio-temporal modeling | Best for simultaneously capturing spatial and temporal dependencies in traffic data [23] | Computationally more demanding than standalone GCN/LSTM; benefits from parallelism [23] | [23] |

|  |  |  |  |  |
|---|---|---|---|---|
| ST-GCN (Spatio-Temporal Graph Convolutional Network) | General class of models combining GCN/GAT with temporal components | Excellent for holistic spatio-temporal traffic prediction [17] | Varies by specific architecture; can be resource-intensive for large networks [22] | [17] |
| DCRNN (Diffusion Convolutional Recurrent Neural Network) | Models spatial dependencies via diffusion process, temporal via RNN | Highly effective for capturing complex spatio-temporal dynamics [29] | Can be computationally and memory-intensive for large highway networks, benefits from graph partitioning [30] | [29] |

## V. Real-Time Optimization and Scalability Considerations

Optimizing an entire city's transportation network in real-time presents formidable computational challenges. This task necessitates processing colossal volumes of data and executing complex calculations within extremely tight timeframes. The system's design must strike a delicate balance between the quality of the optimization solutions and the speed at which these solutions are generated, ensuring that recommendations are timely enough to influence current traffic conditions effectively. The sheer volume of incoming data, coupled with the low-latency requirements of real-time systems, creates a significant challenge regarding data processing and retrieval. This necessitates highly efficient data ingestion, processing, and storage mechanisms. Big data analytics platforms, such as Apache Hadoop and Apache Spark, are often employed in large-scale systems to manage and analyze the vast streams of sensor data from traffic sensors, GPS-equipped vehicles, and surveillance cameras.[40]

To mitigate the computational burden, particularly when dealing with large-scale graphs, graph partitioning techniques are indispensable. These techniques involve dividing the extensive transportation network into smaller, more manageable segments or "sub-networks" that can be processed independently and in parallel.[34]

This approach identifies natural boundaries within the transportation network, where traffic flow exhibits a degree of independence. By optimizing each partition separately while maintaining coordination between adjacent areas, the overall computational load is significantly reduced. Popular software frameworks for graph partitioning, such as METIS, Scotch, and KaHIP, offer advanced capabilities like multi-level partitioning and hybrid methods, which can substantially enhance partitioning quality and decrease computational complexity.[34] Google, for instance, leverages graph partitioning in road networks to accelerate shortest path algorithms like Dijkstra's, demonstrating the practical utility of this approach.[42] For an undergraduate project, implementing a full-scale dynamic graph partitioner might be overly ambitious. However, understanding the underlying principles and applying a pre-partitioned or simplified partitioned network (e.g., focusing on a single, well-defined segment or a few interconnected sub-networks) would provide a valuable demonstration of the concept. Existing libraries can be utilized for partitioning, allowing the GNN model to be applied to these smaller, more manageable subgraphs. This strategy directly addresses the scalability issue by enabling parallelism.

Further enhancing scalability and efficiency, hierarchical optimization approaches are crucial for balancing local and global optimization objectives. This typically involves a multi-level control framework where different layers address distinct scopes and time horizons. At a lower level, localized decisions, such as optimizing traffic signal timings for individual intersections, are handled. Simultaneously, an upper level coordinates broader traffic patterns across an entire corridor or the entire network, providing high-level guidance to the local controllers. This often manifests as a two-level control system: an upper level focusing on macroscopic demand balancing (e.g., using Macroscopic Fundamental Diagram - MFD) and a lower level performing detailed signal timing optimization within subnetworks.[43] Model Predictive Control (MPC) is frequently employed at both levels to determine optimal control sequences by performing numerical optimization over a defined prediction horizon.[43] To further reduce online computational complexity at the lower level, distributed multi-agent control schemes can be implemented, allowing multiple agents to solve sub-problems in parallel while coordinating with their neighbors.[43]

The hierarchical approach inherently distributes control, offering a solution to the scalability challenges of centralized optimization for vast networks. However, coordinating these distributed components introduces its own layer of complexity. For an undergraduate project, a complete hierarchical system might be too extensive. A more feasible approach could involve simulating the interaction between a simplified "global" controller (e.g., one that sets broad traffic flow targets or adjusts signal cycles based on macroscopic predictions) and a "local" controller (e.g., one that optimizes signals for a single intersection or a small cluster of intersections) that receives and acts upon these targets. This would effectively demonstrate how local decisions can be influenced by global guidance, even if the global optimization itself is simplified or relies on heuristic rules. This strategic simplification allows for a focused demonstration of the core principles without overwhelming the project scope.

## VI. Technical Implementation Framework: A Practical Toolkit

The successful implementation of an urban transportation network optimization system with real-time graph updates necessitates a carefully curated technology stack capable of handling demanding computational and data requirements. The selection of tools is paramount for both performance and feasibility within an academic project context.

**Python** serves as the primary development language for this project. Its extensive ecosystem of scientific computing libraries, combined with its versatility and ease of integration with diverse data sources and analytical tools, makes it an ideal choice for complex, data-intensive applications like this.

For building the foundational graph representation of the urban area, **OSMnx** provides essential functionality. This open-source Python package facilitates the retrieval, modeling, analysis, and visualization of real-world street network data from OpenStreetMap.[45] It can convert geographic information into graph structures suitable for algorithmic analysis, allowing for the extraction of road networks, public transit routes, and even pedestrian paths for a chosen area. OSMnx also offers convenient tools to impute missing road speeds and calculate travel times, which are crucial for dynamic edge weight determination.[45] For practical implementation, students should focus on mastering its geocoding, graph creation functions (

graph_from_polygon), and attribute enrichment capabilities (add_edge_speeds, add_edge_travel_times).[47]

**GraphML** is designated as the standard format for storing and exchanging graph data within the project. This XML-based file format, developed by the graph drawing community, supports a wide range of graph structure constellations, including directed, undirected, mixed graphs, and application-specific attributes.[6] Its adoption ensures compatibility with various analytical tools and enables efficient data sharing between different components of the system, particularly useful for saving and loading the base graph structure or sharing it among team members.

**TensorFlow**, a robust end-to-end platform for machine learning, powers the project's core intelligence: the Graph Neural Networks. It is specifically chosen for its capabilities in learning complex traffic patterns and making predictions about future conditions.[49] TensorFlow GNN (TF-GNN), a library built on TensorFlow, is designed for heterogeneous graphs, representing distinct types of nodes and edges, which aligns perfectly with the multi-layer graph model of urban transportation. It also supports distributed learning, crucial for scaling to larger datasets.[49] While powerful, TF-GNN can have a steep learning curve for undergraduates, as its API is "very exact on the structures, input shapes, and formats required to model successfully".[50] Therefore, starting with simpler GNN models or leveraging existing Keras examples for traffic forecasting within TensorFlow is a pragmatic approach.[54]

For managing real-time data and ensuring system responsiveness under heavy computational loads, **Redis** provides high-performance caching. As an open-source, in-memory data store, Redis offers ultra-fast access to frequently used information.[36] Its ability to store data directly in RAM makes it an excellent choice for real-time applications, such as caching dynamically updated edge weights or current traffic conditions. Students should focus on implementing basic key-value operations and potentially explore Redis Pub/Sub for cache invalidation strategies.[36] While RedisGraph, a module built on Redis, allows for storing and querying graph data structures, it might introduce additional complexity beyond the scope of an initial undergraduate project.[57]

The following table summarizes the core technologies and their specific roles within the project:

### Table 3: Core Technologies and Their Role in the Project

| Technology | Primary Function | Key Features for this Project | Relevant Snippets |
|---|---|---|---|
| Python | General-purpose programming, scripting | Rich ecosystem of ML/graph libraries, ease of integration, readability | Technical Implementation Framework |

| | | | |
|---|---|---|---|
| OSMnx | Network data acquisition and modeling | Real-world street network extraction, attribute enrichment, graph conversion | 45 |
| GraphML | Graph data serialization and exchange | Standard XML-based format, supports complex graph structures and attributes | 6 |
| TensorFlow/Keras | Machine learning and GNNs | Scalable GNN implementation, spatio-temporal modeling capabilities, high-level API | 49 |
| Redis | Real-time caching and data store | In-memory data store, low-latency data access, supports various data structures | 36 |

## VII. Adapting the Project for a 4th-Year CSE Undergraduate in India

Implementing a project of this ambition, particularly one reliant on real-time data, within the academic constraints of a 4th-year CSE undergraduate program in India requires careful adaptation.

### Data Acquisition Challenges and Strategies in India

A primary hurdle lies in acquiring high-fidelity, real-time traffic data for Indian cities. While India has made strides in implementing AI-powered traffic solutions in metropolitan areas like Bengaluru, Pune, and Chandigarh [1], publicly accessible, granular real-time data suitable for academic research remains limited.[58] Government Open Data platforms (e.g., data.gov.in, apisetu.gov.in) exist and provide valuable historical or aggregated datasets, but they often lack the real-time flow or speed information necessary for dynamic graph updates.[59] Commercial APIs, such as HERE Traffic API, do offer real-time data but typically come with usage limitations or associated costs that may be prohibitive for an academic project.[65] While object detection datasets like DriveIndia provide rich image data for computer vision tasks, they do not directly offer the traffic flow metrics required for network optimization.[68]

This scarcity of publicly available, real-time, granular traffic data for India is not merely an obstacle; it represents a significant real-world challenge that the project can implicitly address. For the undergraduate project, this data gap can be reframed as a practical constraint that necessitates innovative workarounds. Instead of attempting to acquire live sensor data, the project can demonstrate its capabilities using *simulated data* that closely mimics realistic Indian traffic patterns. This approach allows the student to focus on the algorithmic and modeling complexities without being bogged down by data collection logistics or prohibitive costs.

To address this, several strategies can be employed:

- **Focus on a specific, smaller area:** Rather than attempting to model an entire city, concentrating on a few key intersections, a small grid network, or a university campus environment significantly reduces the scale of graph construction and the volume of data required for real-time processing.
- **Leverage publicly available aggregated data and infer real-time conditions:** Historical data, such as that available from TomTom Traffic Index [58] or xMap.ai [59], can be used to establish baseline traffic patterns (e.g., typical speeds during rush hour versus off-peak). Synthetic "real-time" fluctuations can then be introduced to these baselines to simulate dynamic conditions.
- **Utilize traffic simulation software for data generation:** This is arguably the most practical and robust approach for an undergraduate project to obtain realistic, dynamic data for training and testing the GNN models and optimization algorithms.

The following table summarizes the data availability challenges and proposes concrete workarounds for an undergraduate project in the Indian context:

**Table 4: Data Availability and Challenges for Real-Time Traffic in India**

| Data Type | Source/API (if available) | Availability/Limitations | Potential Workarounds for UG Project | Relevant Snippets |
|---|---|---|---|---|
| Real-time Traffic Flow/Speed | Commercial APIs (HERE Traffic API), Smart City Initiatives | Limited public access, commercial/paid, usage quotas, often aggregated [65] | Traffic simulators (SUMO, CityFlow, CARLA) for synthetic data; focus on a small pilot area [70] | 1 |
| Historical Traffic Patterns | TomTom Traffic Index, xMap.ai, Government Open Data (data.gov.in) | Aggregated data, may not be granular enough for specific segments [58] | Use as baseline for synthetic real-time data generation; focus on a specific city/region [58] | 58 |
| Incident Data (accidents, road closures) | Smart City Initiatives, News APIs (limited public access) | Often reactive, not real-time for granular impact modeling | Simulate incidents within the chosen traffic simulator; use predefined scenarios | 2 |
| Weather Data | Open Weather APIs (e.g., OpenWeatherMap) | Generally accessible, but integration requires careful mapping to traffic impact [11] | Integrate simplified weather effects (e.g., reduced speed during rain) into simulator or GNN features [11] | 11 |
| Public Transit Schedules/Real-time Tracking | Government Open Data (data.gov.in), Transit APIs (limited) | Varies by city/agency, real-time tracking often proprietary [61] | Focus on road network primarily; if multimodal, use simplified schedules or a single transit line within | 9 |

| | | | | |
|---|---|---|---|---|
| | | | simulator [9] | |

**Scope Adjustment for Project Feasibility**

Given the inherent complexities and data limitations, several scope adjustments are advisable to ensure the project's successful completion within an undergraduate timeline:

- **Focus on a Single City/Area:** Instead of attempting to model an entire sprawling metropolitan area, narrowing the scope to a specific zone, a few key intersections, or a contained university campus network significantly reduces the scale of graph construction, data processing, and model training. This allows for deeper exploration of core concepts.
- **Prioritize Road Network Optimization:** While multimodal transportation is a compelling long-term vision, perfecting the dynamic optimization of the road network should be the primary focus. Full multimodal integration can be explicitly designated as a future work direction. If multimodal aspects are critical to the project's unique value proposition, a pragmatic approach would be to model only two modes (e.g., road and bus transit) with a simplified transfer model, demonstrating the concept rather than comprehensive implementation.
- **Simplified GNN Models:** Instead of immediately aiming for state-of-the-art, multi-layered Spatio-Temporal GNNs (ST-GNNs), it is advisable to begin with a simpler GCN or GAT architecture combined with a basic recurrent layer (e.g., GRU or a small LSTM) for spatio-temporal prediction.[17] The objective for an undergraduate project should be to demonstrate the *concept* of GNNs for dynamic graphs and their utility in traffic prediction, rather than achieving benchmark-beating performance.
- **Simulated Real-Time Data:** As previously discussed, relying heavily on traffic simulators to generate dynamic traffic data for training and testing is the most practical strategy. This circumvents the complexities and costs associated with real-world sensor deployment or commercial API subscriptions, allowing the student to control the data environment and focus on algorithmic development.
- **Focus on a Key Output:** Rather than attempting to build a full-fledged, end-to-end traffic management system, the primary output could be a well-defined module, such as a dynamic routing application or a traffic signal optimization module, operating within a simulated environment. This provides a

tangible and demonstrable outcome.

### Leveraging Open-Source Resources and Simulators

The Python ecosystem, coupled with open-source traffic simulators, provides an invaluable toolkit for this project, acting as a crucial proxy for real-world complexity and data.

- **Traffic Simulation Software:**
  - **SUMO (Simulation of Urban MObility):** This is a free and open-source microscopic traffic simulator that allows for modeling intermodal traffic systems, including road vehicles, public transport, and pedestrians. It can import real-world road networks from OpenStreetMap and provides a powerful API (TraCI) for remote control and real-time data integration, making it highly versatile for generating synthetic traffic data and testing control algorithms.[72]
  - **CityFlow:** Positioned as a faster, open-source alternative to SUMO, CityFlow is designed for large-scale city traffic scenarios. It offers a user-friendly Python interface for controlling traffic signals and accessing real-time vehicle information, making it particularly suitable for reinforcement learning environments and large-scale simulations.[70]
  - **CARLA Simulator:** While primarily focused on autonomous driving research, CARLA offers a powerful API to control traffic generation, pedestrian behaviors, and sensor data. It can operate in a "fast simulation" mode that disables rendering, making it suitable for rapid testing of planning and control algorithms without the graphical overhead.[67]
- **Open-Source Libraries:** Beyond OSMnx and TensorFlow, Python's ecosystem includes NetworkX, which is fundamental for general graph manipulation. Libraries like Scikit-learn can be employed for simpler machine learning models, either for baseline comparisons or for initial travel time prediction before transitioning to GNNs.

The use of simulation as a proxy for real-world complexity is a strategic decision that manages expectations and allows the student to concentrate on the core algorithmic and modeling challenges. The choice of simulator should align with the project's specific sub-goals: SUMO for detailed microscopic traffic flow modeling, CityFlow for large-scale scenarios and reinforcement learning-based signal control, or CARLA for interactions with autonomous vehicle behaviors and sensor data.

The following table provides a comparative overview of these open-source traffic simulation software options:

**Table 5: Open-Source Traffic Simulation Software Comparison**

| Simulator | Key Features | Real-time Integration Capabilities | Suitability for Undergrad Projects | Relevant Snippets |
|---|---|---|---|---|
| SUMO | Microscopic, intermodal traffic modeling, OSM import, GUI & command line [72] | TraCI API for external control, data input/output [72] | Excellent for general traffic modeling, control algorithms, and detailed analysis [84] | [70] |
| CityFlow | Faster than SUMO, large-scale city traffic, Python interface for RL [70] | Python API for real-time control of traffic signals and vehicle data access [70] | Excellent for reinforcement learning-based traffic signal control and large-scale simulations [70] | [70] |
| CARLA Simulator | Autonomous driving research focus, high-fidelity rendering, flexible API [71] | Python API for actor control, sensor data, traffic generation [67] | Good for projects involving autonomous vehicle interaction, sensor data, and realistic visualization; potentially high resource usage [71] | [67] |

## VIII. Guided Development Roadmap: Tasks and Milestones

This section outlines a phased approach to the project's execution, providing a structured roadmap with key tasks and milestones tailored for a typical 4th-year undergraduate project timeline (approximately 6-12 months). The project's ambitious nature necessitates an iterative development process, where the initial focus is on demonstrating core innovation on a controlled, smaller scale. This strategic approach helps manage scope and ensures achievable deliverables within the academic period.

**Phase 1: Foundation and Data Modeling (Weeks 1-8)**

This initial phase is dedicated to establishing the theoretical and practical groundwork for the project.

- **Task 1.1: Literature Review and Problem Refinement:** A deep dive into existing research is crucial. This involves thoroughly understanding multi-layer graph theory [4], dynamic routing algorithms, the application of GNNs in traffic prediction [17], and real-time optimization techniques.[40] Concurrently, the broad project idea must be refined into a precise problem statement and a manageable scope for the chosen urban area. This might involve selecting a specific set of intersections, a small grid network, or a university campus environment for initial implementation.
- **Task 1.2: Base Graph Construction:** Utilizing OSMnx, the static representation of the chosen road network (and potentially simplified public transit or bike paths) will be extracted from OpenStreetMap data.[45] This geographic information will then be converted into a suitable graph format, such as NetworkX, and subsequently saved using GraphML for persistence and interoperability.[6] Essential static attributes like speed limits and lane counts will be incorporated into this base graph.[45]
- **Task 1.3: Initial Data Acquisition/Simulation Setup:** A critical step involves identifying and setting up a suitable traffic simulator (e.g., CityFlow or SUMO) for the chosen area.[70] This task includes learning how to access and extract fundamental traffic data such as flow, speed, and queue lengths through the simulator's API.[70] Initial historical traffic patterns can be generated within the simulator to serve as a baseline for dynamic updates.

**Expected Output:** A refined project scope document detailing the specific area and objectives, an initial static graph model of the transportation network, and a working simulator setup capable of generating basic traffic flow data.

**Estimated Duration:** 8 weeks.

**Phase 2: Dynamic Modeling and Prediction (Weeks 9-20)**

This phase focuses on injecting dynamism into the graph and building the predictive intelligence.

- **Task 2.1: Dynamic Edge Weight Implementation:** A module will be developed to continuously update the edge weights of the graph based on simulated real-time traffic data (e.g., current speeds and volumes) obtained from the simulator.[10] This module will implement a function to calculate the travel time for each edge, reflecting current conditions.
- **Task 2.2: GNN Model Selection and Implementation:** A simplified GNN architecture will be chosen, such as a basic Graph Convolutional Network (GCN) or Graph Attention Network (GAT) layer, combined with a lightweight recurrent layer (e.g., GRU or a small LSTM) to capture spatio-temporal features.[17] This model will be implemented using TensorFlow/Keras.[49] The emphasis will be on demonstrating the GNN's ability to learn from the dynamic graph.
- **Task 2.3: Real-time Data Pipeline with Redis:** Redis will be set up as a high-performance cache for storing and retrieving real-time traffic data, including current speeds and predicted travel times for key segments.[36] A data ingestion pipeline will be established to feed real-time data from the simulator into Redis, and a retrieval mechanism will allow the GNN model to access this data efficiently.[86]
- **Task 2.4: Training and Initial Validation:** The developed GNN model will be trained using historical and simulated real-time data. Its performance will be evaluated against standard metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE).[87]

**Expected Output:** A functional module for dynamic edge weight calculation, a trained GNN model capable of traffic prediction, successful integration of Redis for real-time data caching, and a preliminary report on the model's performance metrics.

**Estimated Duration:** 12 weeks.

**Phase 3: Optimization and Evaluation (Weeks 21-28)**

This phase focuses on applying the predictive capabilities to an optimization problem and rigorously evaluating the system.

- **Task 3.1: Routing/Optimization Algorithm Integration:** A shortest path algorithm (e.g., Dijkstra's or A*) will be implemented to leverage the dynamically updated edge weights and suggest optimal routes within the simulated environment. Alternatively, if traffic signal optimization is the primary focus, the GNN's predictions will be integrated into a simple adaptive traffic signal control logic within the chosen simulator.[81]

- **Task 3.2: Performance Evaluation:** Key Performance Indicators (KPIs) relevant to the project's goals will be defined and measured. These may include average travel time reduction, congestion reduction metrics (e.g., queue length, travel time index), and overall prediction accuracy.[87] The evaluation will aim to quantify the benefits of the dynamic, GNN-based approach.
- **Task 3.3: Iterative Refinement:** Based on the performance evaluation, iterative refinements will be made to the GNN model, the real-time data pipeline, and the optimization logic to improve overall system performance.

**Expected Output:** A functional prototype demonstrating dynamic routing or adaptive traffic signal optimization within the simulated environment, accompanied by a comprehensive performance evaluation report detailing the system's efficacy.

**Estimated Duration:** 8 weeks.

**Phase 4: Documentation and Presentation (Weeks 29-32)**

The final phase is dedicated to synthesizing the project's work into a formal report and preparing for demonstration.

- **Task 4.1: Final Report Writing:** All findings, methodologies, results, and discussions from the preceding phases will be compiled into a comprehensive research report. This document will serve as the primary academic output of the project.
- **Task 4.2: Presentation and Demonstration:** A compelling presentation will be prepared, alongside a live demonstration of the functional prototype using the traffic simulator. This will showcase the system's capabilities and the project's innovations.

**Expected Output:** A complete and polished final project report, a professional presentation, and a compelling live demonstration.

**Estimated Duration:** 4 weeks.

The iterative development approach and careful scope management are critical for an undergraduate project of this ambition. The project should prioritize a Minimum Viable Product (MVP) that clearly demonstrates the core innovation—dynamic graph updates with GNNs for prediction and subsequent optimization—on a small, controlled scale within a simulated environment. This ensures successful completion within the academic timeline and provides a solid foundation for future, more complex expansions.

**Table 6: Proposed Project Milestones for a 4th-Year Undergraduate**

| Phase | Key Tasks | Expected Output | Estimated Duration |
|---|---|---|---|
| **Phase 1: Foundation & Data Modeling** | 1.1 Literature Review & Problem Refinement | Refined project scope, detailed problem statement | 2 weeks |
| | 1.2 Base Graph Construction (OSMnx, GraphML) | Initial static graph model of chosen area (.graphml) | 3 weeks |
| | 1.3 Initial Data Acquisition/Simulation Setup | Working traffic simulator setup, basic traffic flow data extraction | 3 weeks |
| **Phase 2: Dynamic Modeling & Prediction** | 2.1 Dynamic Edge Weight Implementation | Module for real-time travel time calculation based on simulated data | 4 weeks |
| | 2.2 GNN Model Selection & Implementation (TensorFlow/Keras) | Basic GNN architecture for traffic prediction (code) | 5 weeks |
| | 2.3 Real-time Data Pipeline with Redis | Redis instance with real-time traffic data, data ingestion/retrieval scripts | 3 weeks |
| | 2.4 Training and Initial Validation | Trained GNN model, initial MAE/RMSE/MAPE metrics | 4 weeks |

| | | | |
|---|---|---|---|
| **Phase 3: Optimization & Evaluation** | 3.1 Routing/Optimization Algorithm Integration | Shortest path algorithm or adaptive signal control logic using dynamic weights | 6 weeks |
| | 3.2 Performance Evaluation (KPIs) | Comprehensive performance report with TTI, queue length, travel time reduction | 2 weeks |
| | 3.3 Iterative Refinement | Improved model/algorithm based on evaluation feedback | (Ongoing) |
| **Phase 4: Documentation & Presentation** | 4.1 Final Report Writing | Complete, polished project report | 3 weeks |
| | 4.2 Presentation and Demonstration | Presentation slides, live demo of prototype in simulator | 1 week |

## IX. Performance Evaluation and Impact Assessment

The evaluation of an urban transportation optimization system must extend beyond simplistic measures of prediction accuracy to encompass a holistic assessment of its impact on network efficiency, congestion, and overall urban mobility. This requires the identification and measurement of a robust set of Key Performance Indicators (KPIs).[91] The project's title, "Urban Transportation Network Optimization," implies a multi-faceted objective that goes beyond merely finding the shortest path. It inherently includes goals such as reducing congestion, enhancing environmental sustainability, and improving the quality of life for commuters. Therefore, the evaluation metrics must reflect this multi-objective nature, demonstrating a comprehensive understanding of urban transportation challenges. The project can also discuss the inherent trade-offs between different optimization objectives, such as minimizing travel time versus minimizing fuel consumption.

### Identification of Key Performance Indicators (KPIs)

For the traffic prediction models, standard metrics provide a quantitative assessment of accuracy:

- **Mean Absolute Error (MAE):** This metric quantifies the average magnitude of the errors in a set of forecasts, without considering the direction of the errors. It offers a straightforward measure of prediction accuracy.[87]
- **Root Mean Square Error (RMSE):** RMSE measures the square root of the average of the square of all errors. This metric gives a relatively high weight to larger errors, making it sensitive to outliers and providing a good indicator of overall prediction performance.[87]
- **Mean Absolute Percentage Error (MAPE):** MAPE expresses accuracy as a percentage of the error relative to the actual value, making it easy to interpret and compare across different scales or datasets.[87]
- **Spatial KPIs:** Beyond average errors, metrics like Global Moran's I, Getis-Ord General G, and Adapted PageRank Algorithm Modified (APAM) can be employed to assess the spatial distribution of prediction errors. These indicators help identify whether errors are randomly distributed or clustered in specific areas, revealing potential hotspots or systematic biases in the model's spatial understanding.[91]

For the routing and optimization algorithms, particularly in terms of congestion reduction, the following KPIs are critical:

- **Average Travel Time Reduction:** This measures the percentage decrease in average trip duration across the simulated network or specific corridors, directly quantifying the efficiency gains.[93]
- **Travel Time Index (TTI):** TTI is the ratio of peak travel time to free-flow travel time. A TTI of 1.2, for example, indicates that travel speeds are 20 percent slower than free-flow conditions. This metric provides a clear indicator of congestion severity.[96]
- **Average Speed:** The overall mean speed of vehicles within the network provides a macroscopic view of traffic flow efficiency.[59]
- **Queue Length / Intersection Delay:** These metrics quantify the average waiting

time or the physical length of vehicle queues at intersections, directly reflecting localized congestion points.[94]

- **Throughput:** Throughput measures the number of vehicles passing through a specific road segment or intersection per unit of time. It indicates the efficiency of traffic movement through critical points in the network.[94]
- **Level of Service (LOS):** LOS is a qualitative measure that describes operational conditions within a traffic stream, ranging from A (free flow) to F (heavy congestion). While qualitative, it provides a standardized way to categorize traffic quality.[94]
- **Reliability:** This KPI assesses how dependable the traffic network is, often measured by the consistency of travel times for a given route over different periods. A highly reliable network experiences fewer unexpected delays.[95]
- **Emissions Reduction:** While challenging to measure directly in a simulated environment without detailed vehicle models, estimated reductions in fuel consumption and pollutants can be inferred from improved traffic flow and reduced idling times, aligning with environmental sustainability goals [Expected Outcomes and Impact].

### Discussion of Expected Practical and Research Contributions

The "Urban Transportation Network Optimization with Real-Time Graph Updates" project, even within a simulated environment, holds significant practical and research contributions.

- **Practical Impact:** Successful implementation is expected to demonstrate measurable reductions in average travel times and improved predictability of trip durations within the simulated urban area. This directly aligns with the objectives of smart city initiatives in India, which aim to enhance urban mobility and reduce congestion.[1] The prototype can serve as a proof-of-concept for city administrators, providing new tools to understand the cascading effects of infrastructure changes or policy decisions on network performance [Expected Outcomes and Impact].
- **Research Contributions:** The project will advance the understanding of how to effectively apply advanced machine learning techniques, specifically GNNs, to dynamic graph problems. The methodologies developed for handling real-time graph updates and predicting traffic flow could find broader applications in other domains characterized by evolving network structures, such as social network

analysis, supply chain optimization, or telecommunications network management [Expected Outcomes and Impact].

- **Environmental Impact:** By optimizing traffic flow and, if multimodal aspects are included, encouraging the efficient use of different transportation modes, the system can contribute to reduced fuel consumption and lower greenhouse gas emissions in a simulated context, supporting broader sustainability objectives [Expected Outcomes and Impact].

The project's potential for broader impact extends beyond its immediate technical outcomes. It serves as a foundational step towards truly intelligent urban infrastructure, systems that not only react to problems but actively work to prevent them, fostering more livable, efficient, and sustainable cities for all residents [Expected Outcomes and Impact].

The following table systematically presents the KPIs for evaluating the project's success:

**Table 7: Key Performance Indicators (KPIs) for Urban Traffic Optimization**

| KPI | Definition | Measurement Method (in simulation) | Relevance to Project Goals | Relevant Snippets |
|---|---|---|---|---|
| **For Traffic Prediction Models** | | | | |
| Mean Absolute Error (MAE) | Average magnitude of prediction errors | Sum of absolute differences between predicted and actual travel times, divided by number of samples | Core prediction accuracy, ease of interpretation | 87 |
| Root Mean Square Error (RMSE) | Square root of the average of squared errors | Square root of the sum of squared differences between | Sensitive to large errors, good for overall model fit | 87 |

| | | | | |
|---|---|---|---|---|
| | | predicted and actual travel times, divided by number of samples | | |
| Mean Absolute Percentage Error (MAPE) | Average percentage of prediction error relative to actual values | Sum of absolute percentage errors, divided by number of samples | Accuracy expressed as a percentage, useful for comparison across scales | 87 |
| Spatial Moran's I | Measures spatial autocorrelation of prediction errors | Statistical index calculated on prediction errors and adjacency matrix | Identifies spatial clustering of errors, reveals areas of poor performance | 91 |
| **For Routing and Optimization Algorithms (Congestion Reduction)** | | | | |
| Average Travel Time Reduction | Percentage decrease in average trip duration | ((Baseline Avg. Travel Time - Optimized Avg. Travel Time) / Baseline Avg. Travel Time) * 100% | Direct measure of efficiency gains, user benefit | 93 |
| Travel Time Index (TTI) | Ratio of peak travel time to free-flow travel time | (Peak Travel Time / Free-Flow Travel Time) | Overall congestion indicator, comparison to ideal conditions | 96 |
| Average Queue Length | Mean number of vehicles waiting at intersections | Average number of vehicles in queues at simulated | Operational efficiency at bottlenecks | 94 |

|  |  |  |  |  |
|---|---|---|---|---|
|  |  | intersections over time |  |  |
| Network Throughput | Number of vehicles passing through network/segment per unit time | Total vehicles successfully completing journeys within simulation period | Overall network capacity utilization | 94 |
| Level of Service (LOS) | Qualitative measure of traffic conditions (A-F) | Derived from simulated speed, density, and delay metrics | Holistic traffic quality assessment | 94 |
| Reliability | Consistency of travel times for a given route | Standard deviation of travel times for specific routes over multiple simulation runs | Predictability of journeys, user satisfaction | 95 |
| Emissions Reduction (Estimated) | Decrease in fuel consumption and pollutants | Estimated from reduced idling times and smoother flow (requires a basic emissions model in simulator) | Environmental sustainability impact | Expected Outcomes and Impact |

## X. Conclusion and Future Directions

The "Urban Transportation Network Optimization with Real-Time Graph Updates" project represents a significant and timely endeavor towards building intelligent urban infrastructure. By fundamentally shifting the paradigm from static traffic management to a dynamic, continuously evolving graph system, the project aims to address the pervasive challenges of congestion, prolonged commute times, and environmental impact that plague modern cities, particularly in rapidly urbanizing nations like India. The proposed methodology, integrating multi-layer graph representation, dynamic edge weight calculation powered by Graph Neural Networks, and sophisticated real-time optimization techniques, offers a robust framework for revolutionizing urban mobility. The careful selection of a Python-centric technology stack, including OSMnx, GraphML, TensorFlow, and Redis, provides a practical and powerful toolkit for implementation.

While the project's ambition is substantial, the outlined adaptations for a 4th-year CSE undergraduate in India ensure its feasibility. By strategically focusing on a smaller geographical area, prioritizing road network optimization, employing simplified GNN models, and critically, leveraging open-source traffic simulators for data generation, the project can effectively demonstrate its core innovations without being constrained by the complexities of real-world data acquisition or large-scale deployment. This simulation-based prototype serves as a powerful proof-of-concept, showcasing the potential for predictive, adaptive traffic management.

The project's success will be quantitatively evaluated through a comprehensive set of KPIs, including prediction accuracy metrics (MAE, RMSE, MAPE, Spatial Moran's I) and operational efficiency indicators (average travel time reduction, travel time index, queue length, network throughput, and level of service). These metrics will provide tangible evidence of the system's ability to reduce congestion, improve travel predictability, and contribute to a more efficient urban environment. Beyond immediate practical demonstrations, the project offers valuable research contributions, advancing the application of machine learning to dynamic graph problems and laying groundwork for future intelligent transportation systems.

Looking ahead, this project serves as a foundational stepping stone, opening numerous avenues for future research and expansion:

- **Advanced GNN Architectures:** Future work could explore more sophisticated Spatio-Temporal GNNs (ST-GNNs, DCRNN, T-GCN, STA-GCN) or dynamic GNNs (DGNNs) to capture even more intricate and subtle traffic patterns, enhancing prediction accuracy and adaptability.[19]
- **Reinforcement Learning for Traffic Signal Control:** A natural and impactful extension would be to integrate the dynamic graph and prediction capabilities with Reinforcement Learning (RL) agents. These agents could then learn and optimize traffic signal timings in real-time within the simulated environment, moving from predictive routing to active traffic control.[81]
- **Full Multimodal Integration:** The multi-layer graph could be expanded to include a wider array of transportation modes, such as ride-sharing services, micro-mobility options (e.g., e-scooters), and even freight logistics. Developing

sophisticated algorithms for truly seamless, real-time multimodal route recommendations, factoring in complex transfer costs and individual user preferences, would be a significant undertaking.

- **Integration of Diverse External Data Sources:** Incorporating a broader spectrum of real-time external factors, such as public event schedules, dynamic construction updates, and more granular social media sentiment analysis, could further refine traffic predictions and optimization strategies by capturing unexpected influences.[11]
- **Real-world Pilot Implementation:** For post-undergraduate research or industry collaboration, exploring opportunities for small-scale pilot implementations with city authorities or transportation departments in India would be invaluable. This would involve leveraging available sensor data, crowdsourced information, or even integrating with existing Intelligent Traffic Management Systems (ITMS).[2]
- **Explainable AI (XAI) for Traffic Decisions:** As AI models become more complex, understanding their decision-making processes is crucial for trust and adoption. Future research could focus on developing methods to make the GNN predictions and optimization decisions more interpretable for human traffic managers and urban planners.
- **Edge Computing Deployment:** Investigating how the real-time data processing and GNN inference could be deployed on edge devices closer to traffic sensors or intersections would significantly reduce latency and bandwidth requirements, pushing the system closer to true real-time operational capability.

This project, even in its undergraduate scope, serves as a vital conceptual and technical blueprint for addressing complex urban mobility challenges, demonstrating how cutting-edge computational science can contribute to building smarter, more efficient, and sustainable cities for the future.

## Works cited

1. Cities are getting Smarter, let's find tech behind it | by JAGRITI BANSAL | Jul, 2025 | Medium, accessed on August 1, 2025, https://medium.com/@jagriti.bansal/cities-are-getting-smarter-lets-find-tech-behind-it-bcebb1bd63d4
2. AI in Indian traffic management: Transforming urban mobility challenges - IndiaAI, accessed on August 1, 2025, https://indiaai.gov.in/article/ai-in-indian-traffic-management-transforming-urban-mobility-challenges
3. Why India Needs Intelligent Traffic Management Systems... - Trois Infotech, accessed on August 1, 2025, https://trois.in/blog/why-india-needs-intelligent-traffic-management-systems/

4. Multidimensional network - Wikipedia, accessed on August 1, 2025, https://en.wikipedia.org/wiki/Multidimensional_network
5. Home · MultilayerGraphs.jl - JuliaGraphs, accessed on August 1, 2025, https://juliagraphs.org/MultilayerGraphs.jl/
6. GraphML - Wikipedia, accessed on August 1, 2025, https://en.wikipedia.org/wiki/GraphML
7. GraphML - yFiles - Documentation - yWorks, accessed on August 1, 2025, https://docs.yworks.com/yfilesajax/yfiles-for-java/doc/developers-guide/graphml.html
8. Graph Database Schema for Multimodal Transportation in Semarang - ResearchGate, accessed on August 1, 2025, https://www.researchgate.net/publication/337104813_Graph_Database_Schema_for_Multimodal_Transportation_in_Semarang
9. Multimodal Travel Routing & Planning - Ex Machina Italia, accessed on August 1, 2025, https://exmitalia.it/en/tech/multimodal-routing/
10. Edge weight calculation based on real traffic street data - Stack Overflow, accessed on August 1, 2025, https://stackoverflow.com/questions/12880473/edge-weight-calculation-based-on-real-traffic-street-data
11. How Weather Influences Social Media Use and Your Ad Performance - LeadSync, accessed on August 1, 2025, https://leadsync.me/blog/weather-social-ads/
12. How weather-driven video ads enhance viewer engagement in the CTV ecosystem, accessed on August 1, 2025, https://www.weathercompany.com/blog/how-weather-driven-video-ads-enhance-viewer-engagement-in-the-ctv-ecosystem/
13. Real Time Traffic Prediction with Machine Learning - MoldStud, accessed on August 1, 2025, https://moldstud.com/articles/p-real-time-traffic-prediction-with-machine-learning
14. Road Traffic Forecast Based on Meteorological Information through Deep Learning Methods, accessed on August 1, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC9227396/
15. Traffic flow prediction for highways based on a multi-task spatiotemporal graph network | Transportation Safety and Environment | Oxford Academic, accessed on August 1, 2025, https://academic.oup.com/tse/article/7/1/tdaf003/7951731
16. W-edge: weighing the edges of the road network, accessed on August 1, 2025, https://www-users.cse.umn.edu/~mokbel/papers/gis18c.pdf
17. Graph Neural Network for Traffic Forecasting: The Research Progress, accessed on August 1, 2025, https://www.mdpi.com/2220-9964/12/3/100
18. Predicting Los Angeles Traffic with Graph Neural Networks | by Amelia Woodward - Medium, accessed on August 1, 2025, https://medium.com/stanford-cs224w/predicting-los-angeles-traffic-with-graph-neural-networks-52652bc643b1
19. Graph neural network for traffic forecasting: A survey - ResearchGate, accessed on August 1, 2025,

https://www.researchgate.net/publication/361517868_Graph_neural_network_for_traffic_forecasting_A_survey
20. Graph Neural Networks for Real-Time Traffic Flow Prediction: Applications in Urban Road Networks - ResearchGate, accessed on August 1, 2025, https://www.researchgate.net/publication/388007146_Graph_Neural_Networks_for_Real-Time_Traffic_Flow_Prediction_Applications_in_Urban_Road_Networks
21. A Survey on Graph Neural Networks in Intelligent Transportation Systems - arXiv, accessed on August 1, 2025, https://arxiv.org/html/2401.00713v2
22. A Survey of Spatio-Temporal Graph Neural Networks for Traffic Prediction - Karan Samel, accessed on August 1, 2025, https://karans.github.io/assets/pdf/Papers/STGNN_Review_Paper.pdf
23. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction - ResearchGate, accessed on August 1, 2025, https://www.researchgate.net/publication/335353434_T-GCN_A_Temporal_Graph_Convolutional_Network_for_Traffic_Prediction
24. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction - arXiv, accessed on August 1, 2025, http://arxiv.org/pdf/1811.05320
25. A Survey on Spatio-Temporal Graph Neural Networks for Traffic Forecasting - ResearchGate, accessed on August 1, 2025, https://www.researchgate.net/publication/378024964_A_Survey_on_Spatio-Temporal_Graph_Neural_Networks_for_Traffic_Forecasting
26. Spatio-Temporal Pivotal Graph Neural Networks for Traffic Flow Forecasting - AAAI Publications, accessed on August 1, 2025, https://ojs.aaai.org/index.php/AAAI/article/view/28707/29368
27. What is Spatial-Temporal Graph Convolutional Networks (ST-GCN) - Activeloop, accessed on August 1, 2025, https://www.activeloop.ai/resources/glossary/spatial-temporal-graph-convolutional-networks-st-gcn/
28. STA-GCN: Spatial-Temporal Self-Attention Graph Convolutional Networks for Traffic-Flow Prediction - MDPI, accessed on August 1, 2025, https://www.mdpi.com/2076-3417/13/11/6796
29. System architecture for the Diffusion Convolutional Recurrent Neural... - ResearchGate, accessed on August 1, 2025, https://www.researchgate.net/figure/System-architecture-for-the-Diffusion-Convolutional-Recurrent-Neural-Network-designed-for_fig2_318316069
30. Graph-Partitioning-Based Diffusion Convolutional Recurrent Neural Network for Large-Scale Traffic Forecasting - Tanwi Mallick, accessed on August 1, 2025, https://tanwimallick.github.io/papers/Graph-Partitioning-Based%20DCRNN.pdf
31. COOL: A Conjoint Perspective on Spatio-Temporal Graph Neural Network for Traffic Forecasting - arXiv, accessed on August 1, 2025, https://arxiv.org/pdf/2403.01091
32. A survey of dynamic graph neural networks - arXiv, accessed on August 1, 2025, https://arxiv.org/html/2404.18211v1
33. A Robust Comparative Analysis of Graph Neural Networks on Dynamic Link Prediction, accessed on August 1, 2025,

https://www.researchgate.net/publication/360705984_A_Robust_Comparative_Analysis_of_Graph_Neural_Networks_on_Dynamic_Link_Prediction
34. Optimizing Graph Partitioning for Real-World Problems - Number Analytics, accessed on August 1, 2025, https://www.numberanalytics.com/blog/optimizing-graph-partitioning-real-world-problems
35. (PDF) Grid Partition-Based Dynamic Spatial–Temporal Graph Convolutional Network for Large-Scale Traffic Flow Forecasting - ResearchGate, accessed on August 1, 2025, https://www.researchgate.net/publication/391894424_Grid_Partition-Based_Dynamic_Spatial-Temporal_Graph_Convolutional_Network_for_Large-Scale_Traffic_Flow_Forecasting
36. Redis + Local Cache: Implementation and Best Practices | by Max - Medium, accessed on August 1, 2025, https://medium.com/@max980203/redis-local-cache-implementation-and-best-practices-f63ddee2654a
37. How In-Memory Caching Works in Redis - freeCodeCamp, accessed on August 1, 2025, https://www.freecodecamp.org/news/how-in-memory-caching-works-in-redis/
38. Dynamic Travel Time Prediction with Real-Time and Historic Data ..., accessed on August 1, 2025, https://www.researchgate.net/publication/238180852_Dynamic_Travel_Time_Prediction_with_Real-Time_and_Historic_Data
39. Travel Time Prediction Using Machine Learning Algorithms: Focusing on k-NN, LSTM, and Transformer - ResearchGate, accessed on August 1, 2025, https://www.researchgate.net/publication/385854931_Travel_Time_Prediction_Using_Machine_Learning_Algorithms_Focusing_on_k-NN_LSTM_and_Transformer
40. Big Data Analytics for Smart Cities: Optimizing Urban Traffic Management Using Real-Time Data Processing | Journal of Computer Science and Technology Application, accessed on August 1, 2025, https://journal.corisinta.org/corisinta/article/view/56
41. Geometry, flows, and graph-partitioning algorithms - People @EECS, accessed on August 1, 2025, https://people.eecs.berkeley.edu/~vazirani/pubs/arvcacm.pdf
42. Efficient Partitioning of Road Networks - Google Research, accessed on August 1, 2025, https://research.google/blog/efficient-partitioning-of-road-networks/
43. Two-level hierarchical model-based predictive control for large ..., accessed on August 1, 2025, https://www.dcsc.tudelft.nl/~bdeschutter/pub/rep/16_010.pdf
44. Hierarchical Framework for Real-Time Traffic Control - Transportation Research Board, accessed on August 1, 2025, https://onlinepubs.trb.org/Onlinepubs/trr/1992/1360/1360-014.pdf
45. sas-network-analysis/applications/osm/Road_Network_TSP.ipynb at main - GitHub, accessed on August 1, 2025, https://github.com/sassoftware/sas-network-analysis/blob/main/applications/osm/Road_Network_TSP.ipynb
46. Modeling and Analyzing Urban Networks and Amenities with OSMnx - Geoff

Boeing, accessed on August 1, 2025, https://geoffboeing.com/share/osmnx-paper.pdf
47. Getting Started - OSMnx 2.0.5 documentation, accessed on August 1, 2025, https://osmnx.readthedocs.io/en/stable/getting-started.html
48. Get OSM Features Within a Distance of Address Using OSMnx Feature Module - GeeksforGeeks, accessed on August 1, 2025, https://www.geeksforgeeks.org/python/get-osm-features-within-a-distance-of-address-using-osmnx-feature-module/
49. TensorFlow, accessed on August 1, 2025, https://www.tensorflow.org/
50. TensorFlow-GNN: An End-To-End Guide For Graph Neural Networks, accessed on August 1, 2025, https://towardsdatascience.com/tensorflow-gnn-an-end-to-end-guide-for-graph-neural-networks-a66bfd237c8c/
51. Graph neural networks in TensorFlow, accessed on August 1, 2025, https://blog.tensorflow.org/2024/02/graph-neural-networks-in-tensorflow.html
52. MassimoPerini/online-gnn-learning - GitHub, accessed on August 1, 2025, https://github.com/MassimoPerini/online-gnn-learning
53. GNN 101: Visual Learning of Graph Neural Networks in Your Web Browser - arXiv, accessed on August 1, 2025, https://arxiv.org/html/2411.17849v1
54. Traffic forecasting using graph neural networks and LSTM - Keras, accessed on August 1, 2025, https://keras.io/examples/timeseries/timeseries_traffic_forecasting/
55. Python Redis: A Beginner's Guide - DataCamp, accessed on August 1, 2025, https://www.datacamp.com/tutorial/python-redis-beginner-guide
56. Redis - The Real-time Data Platform, accessed on August 1, 2025, https://redis.io/
57. RedisGraph - Medium, accessed on August 1, 2025, https://medium.com/@emmanueldavidson/redisgraph-1a13660b8cbd
58. India traffic report | TomTom Traffic Index, accessed on August 1, 2025, https://www.tomtom.com/traffic-index/india-country-traffic/
59. India | Road Traffic Data - xMap, accessed on August 1, 2025, https://www.xmap.ai/data-catalogs/india-road-traffic-data
60. Big Data Traffic Study Identifies India's Fastest and Slowest Cities - Haas News, accessed on August 1, 2025, https://newsroom.haas.berkeley.edu/research/big-data-traffic-study-identifies-indias-fastest-and-slowest-cities-2/
61. APIs - Open Government Data (OGD) Platform India, accessed on August 1, 2025, https://www.data.gov.in/apis/?sector=Transport
62. API Setu: Get access to thousands of APIs, accessed on August 1, 2025, https://apisetu.gov.in/
63. APIs | Open Government Data (OGD) Platform India, accessed on August 1, 2025, https://www.data.gov.in/apis
64. Traffic |Open Government Data (OGD) Platform India, accessed on August 1, 2025, https://www.data.gov.in/keywords/Traffic
65. API Monitoring: Metrics, Challenges and Best Practices - Catchpoint, accessed on August 1, 2025, https://www.catchpoint.com/api-monitoring-tools/api-monitoring

66. Refining Public and Anonymous API Traffic Management to Ensure Long-Term Sustainability - ORCID, accessed on August 1, 2025, https://info.orcid.org/refining-api-traffic-management/
67. Real-Time Traffic - HERE Technologies, accessed on August 1, 2025, https://www.here.com/docs/bundle/traffic-api-developer-guide-v7/page/topics/concepts/real-time-traffic.html
68. [2507.19912] DriveIndia: An Object Detection Dataset for Diverse Indian Traffic Scenes, accessed on August 1, 2025, https://www.arxiv.org/abs/2507.19912
69. DriveIndia: An Object Detection Dataset for Diverse Indian Traffic Scenes - arXiv, accessed on August 1, 2025, https://arxiv.org/html/2507.19912v1
70. CityFlow, accessed on August 1, 2025, https://cityflow-project.github.io/
71. CARLA Simulator, accessed on August 1, 2025, https://carla.org/
72. Traffic Modeling with SUMO: a Tutorial - arXiv, accessed on August 1, 2025, https://arxiv.org/html/2304.05982v2
73. Eclipse SUMO - Simulation of Urban MObility, accessed on August 1, 2025, https://eclipse.dev/sumo/about/
74. Sumo Logic APIs, accessed on August 1, 2025, https://help.sumologic.com/docs/api/
75. Sumo Logic API reference, accessed on August 1, 2025, https://api.sumologic.com/docs/
76. sumo/docs/web/docs/sumo.md at main - GitHub, accessed on August 1, 2025, https://github.com/eclipse/sumo/blob/main/docs/web/docs/sumo.md
77. SUMO Interface - Webots documentation, accessed on August 1, 2025, https://cyberbotics.com/doc/automobile/sumo-interface
78. Introduction — CityFlow 0.1 documentation, accessed on August 1, 2025, https://cityflow.readthedocs.io/en/latest/introduction.html
79. Welcome to CityFlow's documentation! — CityFlow 0.1 documentation, accessed on August 1, 2025, https://cityflow.readthedocs.io/en/latest/
80. Quick Start — CityFlow 0.1 documentation, accessed on August 1, 2025, https://cityflow.readthedocs.io/en/latest/start.html
81. Traffic Signal Control via Reinforcement Learning: A Review on Applications and Innovations - MDPI, accessed on August 1, 2025, https://www.mdpi.com/2412-3811/10/5/114
82. Traffic Manager - CARLA Simulator - Read the Docs, accessed on August 1, 2025, https://carla.readthedocs.io/en/latest/adv_traffic_manager/
83. carla/PythonAPI/examples/invertedai_traffic.py at ue5-dev - GitHub, accessed on August 1, 2025, https://github.com/carla-simulator/carla/blob/ue5-dev/PythonAPI/examples/invertedai_traffic.py
84. Integrating Machine Learning and Traffic Simulation for Enhanced Traffic Management and Optimization - IRJET, accessed on August 1, 2025, https://www.irjet.net/archives/V10/i9/IRJET-V10I9112.pdf
85. A Computationally Efficient Simulation-Based Optimization Algorithm for Large-Scale Urban Transportation Problems - PubsOnLine - INFORMS.org, accessed on August 1, 2025,

https://pubsonline.informs.org/doi/10.1287/trsc.2014.0550
86. How to asynchrony get a real time data from redis by using python flask socket.io and threads? - Stack Overflow, accessed on August 1, 2025, https://stackoverflow.com/questions/34736329/how-to-asynchrony-get-a-real-time-data-from-redis-by-using-python-flask-socket-i
87. Traffic flow prediction with a multi-dimensional feature input: A new method based on attention mechanisms - AIMS Press, accessed on August 1, 2025, https://aimspress.com/article/doi/10.3934/era.2024048?viewType=HTML
88. Adaptive Traffic Signal Control Based on Graph Neural Networks and Dynamic Entropy-Constrained Soft Actor–Critic - MDPI, accessed on August 1, 2025, https://www.mdpi.com/2079-9292/13/23/4794
89. A deep reinforcement learning model for large-scale traffic signal control based on graph meta-learning using local subgraphs, accessed on August 1, 2025, http://scis.scichina.com/en/2025/172203.pdf
90. (PDF) Traffic Signal Control via Reinforcement Learning: A Review on Applications and Innovations - ResearchGate, accessed on August 1, 2025, https://www.researchgate.net/publication/391508029_Traffic_Signal_Control_via_Reinforcement_Learning_A_Review_on_Applications_and_Innovations
91. (PDF) Spatial Performance Indicators for Traffic Flow Prediction - ResearchGate, accessed on August 1, 2025, https://www.researchgate.net/publication/387288427_Spatial_Performance_Indicators_for_Traffic_Flow_Prediction
92. Spatial Performance Indicators for Traffic Flow Prediction - MDPI, accessed on August 1, 2025, https://www.mdpi.com/2076-3417/14/24/11952
93. Smart Traffic Management Stats: Congestion Reduction & AI Integration - PatentPC, accessed on August 1, 2025, https://patentpc.com/blog/smart-traffic-management-stats-congestion-reduction-ai-integration
94. Performance Measurement in Road Traffic - Top 10 KPIs for Every Traffic Planner | Isarsoft, accessed on August 1, 2025, https://www.isarsoft.com/article/performance-measurement-in-road-traffic-top-10-kpis-for-every-traffic-planner
95. What Are The Key Performance Indicators (KPIs) For Smart Traffic? - Talking Tech Trends, accessed on August 1, 2025, https://www.youtube.com/watch?v=MbbCajrEzBA
96. Additional Performance Measures, accessed on August 1, 2025, https://performance.oki.org/mobility-congestion/additional-performance-measures/
97. SimST: A GNN-Free Spatio-Temporal Learning Framework for Traffic Forecasting, accessed on August 1, 2025, https://openreview.net/forum?id=2ppuWD3dkie
98. OSMnx: Python and OpenStreetMap - Talk Python to Me Ep. 495 - YouTube, accessed on August 1, 2025, https://www.youtube.com/watch?v=PGE2fwNc0zE
99. Traffic Congestion Level - KPI Depot, accessed on August 1, 2025, https://kpidepot.com/kpi/traffic-congestion-level