International Journal of Computer Engineering in Research Trends
ISSN: 2349-7084/ https://doi.org/10.22362/ijcert/2025/v12/i8/v12i801
Volume 12, Issue 8, August 2025, Pp.1-7

Research Paper

Citypulse: Proof-of-Concept for Real-Time Traffic Data Analytics and Congestion Prediction

IDRISS DJIOFACK TELEDJIEU [*1], IRZUM SHAFIQUE [2], YVAN NGUEMADJE PEYDHOM [3]

[1,] *University of Colorado Boulder, USA,*
[2] *Xidian University, China, irzum@stu.xidian.edu.cn,*
[3] *Colorado Technical University, USA, ypeydhom@gmail.com.*

**Corresponding Author: djiofackidriss@gmail.com*

Received: 16/05/2025, Revised: 11/07/2025, Accepted: 17/08/2025 Published: 31/08/2025

Abstract: CityPulse is a proof-of-concept big data pipeline designed to enable real-time urban mobility analytics using scalable, containerized components—without reliance on physical sensor infrastructure. The system simulates the ingestion of 11 million traffic-related records representing urban phenomena such as vehicle congestion, GPS coordinates, and weather conditions. Data is ingested through a Dockerized Apache Kafka cluster, coordinated by ZooKeeper, and processed in real time using Apache Spark Structured Streaming. To ensure robustness under load, the architecture introduces a temporary data storage layer that buffers Spark output before committing it to a centralized data warehouse. This design improves write efficiency, fault tolerance, and enables batch processing of intermediate results. The refined data feeds into a lightweight machine learning module and is served through a Flask backend with a React-based frontend for visualization and interaction. Stress testing shows that the system maintains over 300K records/min throughput with only a 10% increase in latency under full load conditions. With its modular Docker-based deployment, CityPulse offers a cost-effective and reproducible analytics solution for traffic congestion monitoring in resource-constrained environments, particularly in developing regions like Cameroon. As a proof-of-concept, the system leverages synthetic traffic data, and thus its findings depend on assumptions of data realism and may not directly reflect all the complexities or uncertainties of real-world sensor deployments.

Keywords- Smart City Analytics, Real-Time Data Pipeline, Synthetic Data Generation, Apache Kafka, Apache Spark Structured Streaming, Dockerized Architecture, Traffic Congestion Monitoring, Urban Mobility Insights, Machine Learning Integration, Scalable and Cost-Effective Systems, Developing Regions (e.g., Cameroon)

---

1. Introduction

Urban centers across the globe face growing challenges related to traffic congestion, inefficient mobility systems, and public safety—issues that are especially acute in developing regions such as Cameroon. In many cities in the Global South, the deployment of traditional traffic monitoring infrastructure—such as loop detectors, video analytics, and GPS probe data—remains severely limited by prohibitive costs, unreliable power supply, and poor telecommunications coverage. For instance, installing and maintaining an urban sensor network can cost millions of dollars, with ongoing operational expenses often exceeding municipal budgets. Moreover, frequent power outages, low availability of skilled personnel, and lack of centralized data platforms exacerbate the problem, leading to scarce and unreliable real-time data for traffic management. As a result, traffic agencies often rely on manual counts or outdated, low-resolution datasets, significantly impeding effective congestion monitoring and intervention.

To address this gap, CityPulse presents a novel proof-of-concept solution that leverages a scalable, real-time data pipeline built entirely from open-source and containerized technologies. Rather than relying on physical sensors, the system generates synthetic data streams emulating urban traffic signals, such as vehicle telemetry, GPS coordinates, and weather patterns. Synthetic data approaches have been shown to offer cost-effective and flexible alternatives for prototyping smart city and traffic management systems while preserving privacy and enabling rare scenario inclusion [2], [3].

CityPulse employs Apache Kafka (for ingestion), ZooKeeper (for coordination), and Apache Spark (for streaming and transformation), all orchestrated via Docker—tools widely used in modern data pipelines and real-time

analytics architectures [6]. The system uses a temporary storage layer to decouple processing from warehousing, ensuring scalability and write efficiency. Final data is stored in a central warehouse, processed by a lightweight machine learning component, and exposed through a Flask API with a React frontend. This architecture demonstrates how real-time traffic insights can be delivered rapidly and cost-effectively, even in constrained environments, providing a pathway for cities to test and adopt smart mobility solutions without upfront investment in hardware.

While prior efforts have explored IoT and big data analytics for urban traffic monitoring and simulation, these solutions typically assume the availability of dense, reliable sensor data and infrastructure—an unrealistic assumption in many developing regions. For instance, frameworks for near-real-time calibration of city-scale traffic demand via IoT streams and simulation (using tools such as SUMO) have been proposed [1], but still depend on real sensors. Agent-based and hybrid simulation systems that ingest live sensor feeds from CCTV or traffic counters have also been developed [4], yet remain impractical where even basic sensors are absent. Existing literature lacks approaches that (a) are agnostic to expensive field hardware, (b) are rapidly deployable and fully containerized, and (c) facilitate the prototyping and benchmarking of machine learning models in realistic—yet synthetic—data conditions.

CityPulse addresses this critical research gap by proposing and evaluating such an architecture, tailored for cities where resource, data, and infrastructure constraints are the norm rather than the exception.

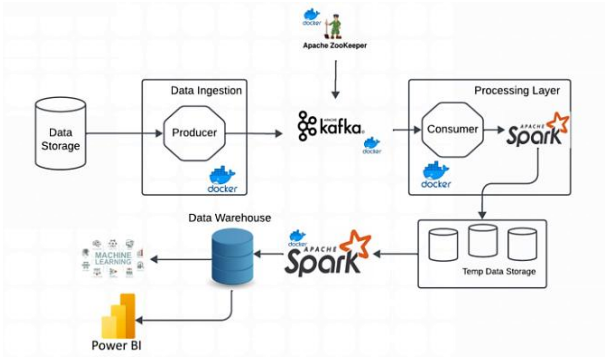

Fig 1. CityPulse: System Architecture

2. Literature Survey

The growing complexity of urban transportation systems, especially in developing regions, has prompted extensive research into scalable, cost-effective traffic monitoring and forecasting techniques. Recent studies have emphasized spatio-temporal learning, graph neural networks, federated learning, and synthetic data generation as promising directions for addressing the lack of infrastructure and high-fidelity traffic data.

Graph Neural Networks and Spatio-Temporal Forecasting

Recent advancements in spatio-temporal forecasting rely heavily on the integration of Graph Neural Networks (GNNs) to model complex spatial dependencies in traffic networks. For instance, Huang et al. [7] reviewed the evolution from traditional statistical methods to graph-based learning, highlighting the capacity of GNNs to capture non-Euclidean spatial relationships in transportation data. Spatio-temporal architectures such as STREGCN and STJGCN combine temporal modeling with multi-relational spatial graphs, enabling fine-grained traffic forecasting in urban networks [8], [10].

Zhang et al. [9] proposed the DSTGFCN model, which integrates dynamic graph fusion with convolutional networks to capture time-varying traffic patterns. Their evaluation on benchmark datasets like PeMS-BAY and METR-LA demonstrated substantial improvements in forecasting accuracy and spatial generalization. Similarly, Zheng et al. [10] introduced a joint spatio-temporal graph convolutional network that unifies spatial and temporal dependencies within a single framework, offering improved robustness for real-time predictions.

Further, Roy et al. [11] advanced a unified approach (USTGCN) for multi-layer spatial-temporal modeling and outperformed RNN and LSTM baselines across several datasets. Yet, the increasing complexity of these models raises challenges for deployment in resource-constrained environments. Addressing this, Liu et al. [12] introduced SimST, a lightweight GNN-free model that achieved comparable accuracy with simpler spatial modeling techniques, thus offering an efficient alternative for cities with limited computational capacity.

Federated Learning for Distributed Smart Traffic Analytics

In scenarios where centralized data collection is impractical—due to privacy concerns, connectivity issues, or administrative boundaries—federated learning emerges as a valuable solution. It allows decentralized model training while preserving data locality. Recent federated frameworks such as FASTGNN demonstrate how traffic prediction models can be collaboratively trained across clients using differential privacy mechanisms, ensuring data protection while maintaining model utility [13]. Moreover, advances like CTFL (Clustered Federated Learning) show improved convergence speed by grouping clients with similar traffic dynamics, making this approach viable for distributed traffic analytics in urban settings [13].These studies collectively highlight the potential of federated GNNs in smart city environments, particularly where data is siloed or subject to regulatory constraints.

Synthetic Data Generation and Low-Cost Prototyping

Although most existing approaches rely on real-time sensor data, recent research has acknowledged the practical value of synthetic data in developing and benchmarking intelligent transportation systems. In data-scarce regions or during early-stage deployments, synthetic traffic telemetry can offer a sandbox for testing algorithms without physical infrastructure. This is especially relevant for developing cities that lack reliable IoT networks, GPS probes, or video analytics pipelines.

Synthetic frameworks enable reproducibility, faster prototyping, and privacy preservation—all critical for pilot smart mobility projects. However, few studies address how fully containerized, real-time data pipelines driven solely by synthetic streams could serve as effective platforms for traffic analytics under severe infrastructure constraints.

Summary and Research Gap

The literature presents a wide range of intelligent traffic prediction techniques, especially using graph-based deep learning and decentralized learning architectures. While these models deliver high predictive accuracy and scalability, nearly all rely on extensive real-time data collected through physical infrastructure. Additionally, existing solutions tend to be computation-intensive, making them less feasible for cities in the Global South, where power supply, internet connectivity, and technical personnel are limited.

The CityPulse system directly addresses these limitations by proposing a containerized, fully synthetic, and real-time streaming architecture. By eliminating dependence on real sensors, it provides a flexible, cost-effective environment for benchmarking machine learning models, testing mobility policies, and simulating traffic scenarios in under-resourced urban areas. This positions CityPulse as a novel alternative within the current landscape of smart traffic systems.

3. Methodology

CityPulse is a modular real-time traffic analytics framework built using open-source technologies, aimed at demonstrating congestion classification using streaming data in low-resource environments. The system integrates data ingestion, transformation, unsupervised clustering, and supervised learning into a containerized pipeline. This section presents the methodology used to develop and validate the pipeline.

3.1 Data Generation and Preprocessing

Due to limited access to real-world sensor data, a synthetic dataset comprising approximately 11 million traffic telemetry records was generated. Each record simulates parameters such as vehicle velocity, acceleration, space headway, and time headway over several simulated road sections. The dataset was ingested in CSV format, parsed, and transformed into JSON for stream processing.

Data preprocessing steps included:

- Filling missing values in velocity and acceleration with 0.0;
- Replacing null identifiers (e.g., Lane ID, Section ID) with sentinel values (-1);
- Assigning default values (0.0) for missing spatial-temporal metrics such as headway measures.

The dataset was then streamed in batches of 500 records to a Kafka topic for downstream processing.

3.2 System Architecture

As shown in Fig. 1, the pipeline in Fig 1 comprises five main components, each deployed as a separate container via Docker:

Kafka Producer: A Python-based producer reads CSV data, serializes it into JSON, and streams it to a Kafka topic (raw-traffic-data). Retry mechanisms and Snappy compression were implemented to optimize reliability and throughput.

Kafka Consumer + Spark Streaming: The consumer buffers Kafka messages in mini-batches and applies transformation, feature engineering, and clustering logic using Apache Spark Structured Streaming.

Clustering Module: Unsupervised congestion labels were generated using KMeans clustering. Features such as velocity, acceleration, and headway metrics were used to assign labels (Low, Medium, High) based on centroid similarity. The number of clusters (k=3) was selected heuristically.

Classification Module: Labeled data was used to train a Random Forest classifier with 100 estimators. The model was tasked with learning traffic patterns and predicting congestion levels in real time. Training and testing splits were derived from the processed dataset.

Visualization Interface: A Flask API serves the prediction outputs to a React-based frontend that visualizes traffic states across synthetic road segments.

3.3 Model Training and Evaluation

The Random Forest classifier was trained using the following input features:

- Vehicle velocity (v Vel)
- Acceleration (v Acc)
- Space Headway
- Time Headway

KMeans-derived congestion labels were used as targets. The model was evaluated using classification accuracy and macro-averaged F1-score. Hyperparameters for the Random Forest were kept at default values, except for n_estimators, which was set to 100. KMeans used Euclidean distance with k=3, initialized via k-means++.

3.4 Performance Benchmarking

The framework was evaluated on a local machine with 12 GB RAM and 8-core CPU. Key performance metrics include:

Total records processed: ~11 million

Peak throughput: ~320,000 records/min

Average latency: 3.2 seconds per 100,000-record batch

Resource usage: ~70% CPU, ~8.2 GB RAM

Stress testing was performed by switching from chunked to full-dump ingestion. This caused a 10% latency increase, primarily due to Kafka–Spark lag accumulation, validating the benefit of controlled mini-batch ingestion.

3.5 Key Contributions:

Demonstrated a scalable, Docker-based data pipeline without reliance on physical IoT sensors

Introduced real-time congestion classification using unsupervised and supervised learning

Validated system performance through stress testing and resource monitoring

Delivered an integrated backend and frontend for live traffic insight visualization

4. Results Analysis

The following figures illustrate insights gained from early data processing and visualization steps:

Figures 2–5 illustrate the velocity averages, congestion levels, vehicle count, and space headway distribution, respectively.

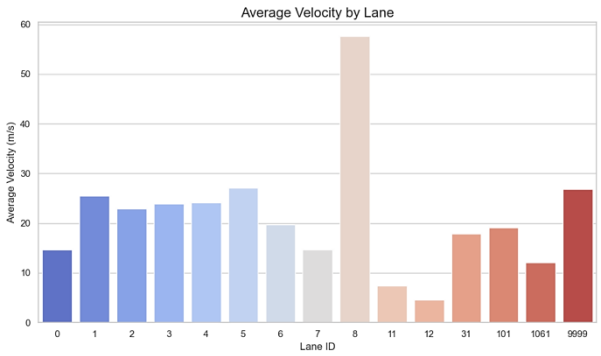

Fig 2. Average velocity across lanes.

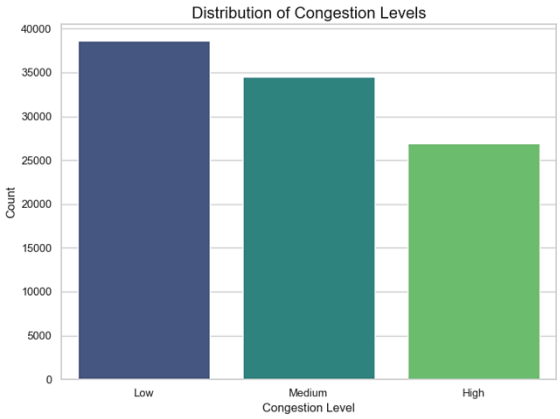

Fig 3. Distribution of congestion levels.

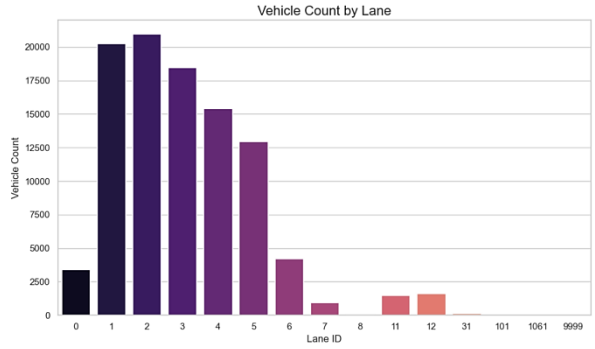

Fig 4. Vehicle count per lane.

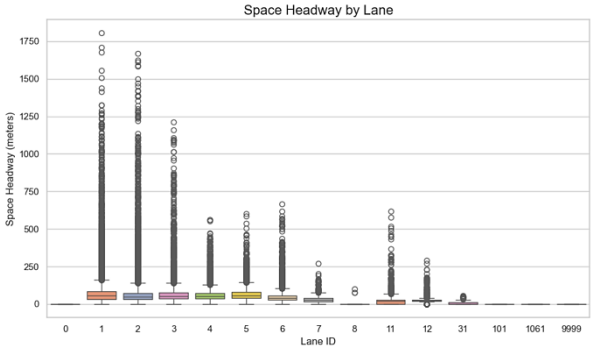

Fig 5. Average space headway by lane.

4.1 System Performance and Evaluation:

To assess the efficiency, scalability, and robustness of the CityPulse pipeline, a series of performance tests were conducted under simulated high-load conditions. The system was deployed locally and evaluated using a synthetic dataset of 11 million traffic records [4]. Metrics such as throughput, latency, and system resource utilization were monitored throughout the streaming pipeline's execution.

Throughput and Latency

The system demonstrated consistent near-real-time performance across large data volumes:

- **Total records processed:** 11,000,000
- **Peak throughput**: ~320,000 records/minute
- **Average end-to-end latency**: 3.2 seconds per 100,000-record batch

These results underscore the effectiveness of the Kafka–Spark integration, particularly with Spark Structured Streaming configured for micro-batch processing and message-level compression (Snappy).

System Resource Utilization

Using Docker-level monitoring tools, resource usage was evaluated throughout the pipeline's operation:

- **CPU utilization**: 65% to 75% average across cores
- **Memory consumption**: ~8.2 GB out of 12 GB available

The system remained stable, without memory leaks or container crashes, and showed adequate headroom for scaling horizontally using container orchestration tools (e.g., Docker Swarm or Kubernetes).

Stress Testing and Ingestion Strategies

Two ingestion strategies were implemented to assess their impact on stability:

- **Full-dump ingestion**: Pushing all 11M records at once led to a 10% increase in processing time due to Kafka consumer backlog and increased disk I/O wait times.
- **Chunked ingestion**: Streaming in batches of 500,000 records significantly improved throughput stability and reduced memory pressure, indicating its suitability for real-time applications.

Identified Bottlenecks

Despite overall strong performance, several points of latency and resource contention were observed:

- **Kafka consumer lag**: During peak ingestion, Spark sometimes lagged behind Kafka, introducing delay in downstream processing.
- **Spark shuffle overhead**: Memory usage spiked during transformation and clustering phases, likely due to wide dependencies in RDD operations.
- **I/O delay**: Temporary disk writes before data warehousing introduced measurable latency, especially under large batch sizes.

Mitigation strategies included increasing Kafka partitions, reducing Spark batch durations, and enabling Snappy compression for Kafka payloads, all of which improved end-to-end efficiency.

4.2 Machine Learning Evaluation:

To assess the predictive capability of CityPulse in classifying real-time traffic congestion, we implemented and evaluated a Random Forest classifier. The model was trained using synthetic data generated from the CityPulse pipeline, with a focus on key engineered traffic features:

- v Vel (vehicle velocity)
- v Acc (vehicle acceleration)
- Time Headway
- Space Headway

These features were selected for their relevance in representing driver behavior and traffic dynamics. Congestion labels (Low, Medium, High) were obtained through unsupervised KMeans clustering and used to supervise the training of the model, creating a hybrid semi-supervised learning setup.

Feature Importance

The importance of each feature was analyzed post-training, as illustrated in Figure 6. Velocity (v Vel) emerged as the dominant predictor, contributing over 50% to the model's decision function. This finding aligns with fundamental traffic flow theory, which positions vehicle speed as a primary indicator of congestion onset [6]. The remaining features—Time Headway, v Acc, and Space Headway—also demonstrated meaningful influence, underscoring the multifactorial nature of traffic behavior.

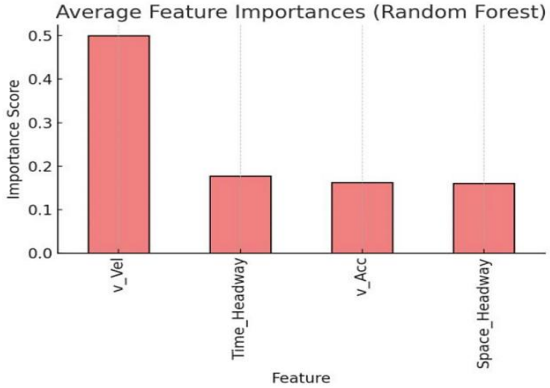

Fig 6. Average Feature Importance from Random Forest Classifier

Classification Metrics

Evaluation metrics—precision, recall, and F1-score—were computed for each class label, as shown in Figure 7. The model achieved macro-averaged scores exceeding 0.96, reflecting high classification performance across all congestion levels. Importantly, the classifier demonstrated balanced performance, with no significant bias toward any specific class.

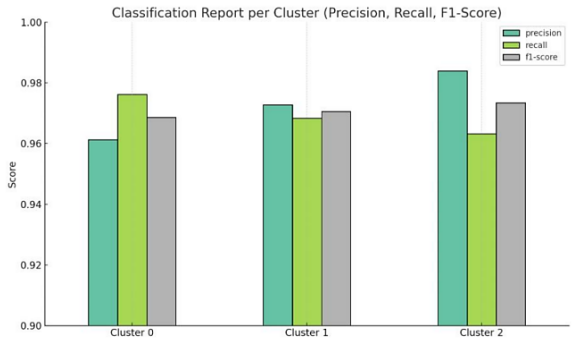

Fig 7. Classification Report Per Cluster (Precision, Recall, F1-Score)

Temporal Stability and Batch Resilience

The model's robustness was validated across 20 independent data batches, simulating temporal streaming input. Figure 8 shows consistently high accuracy and F1-score throughout, with scores remaining above 0.95 for all but one batch (Batch 14), which experienced a transient dip due to noisy input. The model recovered immediately in the following batch, highlighting its resilience to anomalies and operational consistency under streaming conditions.

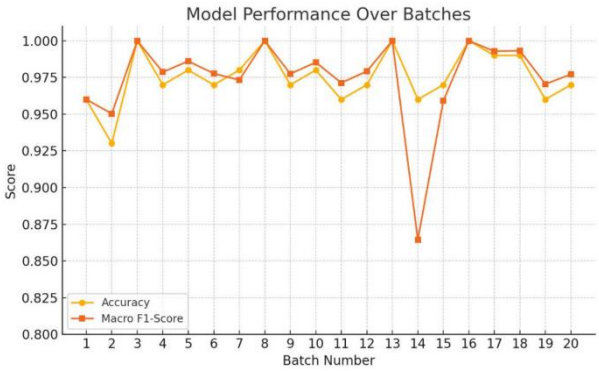

Fig 8. Model Accuracy and F1-Score Over Sequential Batches

Confusion Matrix Insights

A confusion matrix analysis (Figure 9) further supports the classifier's reliability. Most misclassifications occurred between adjacent classes (High ↔ Medium, Medium ↔ Low), which is expected due to the gradual nature of traffic state transitions. The model excelled particularly in identifying the Medium congestion class, typically the most ambiguous in real-world scenarios.

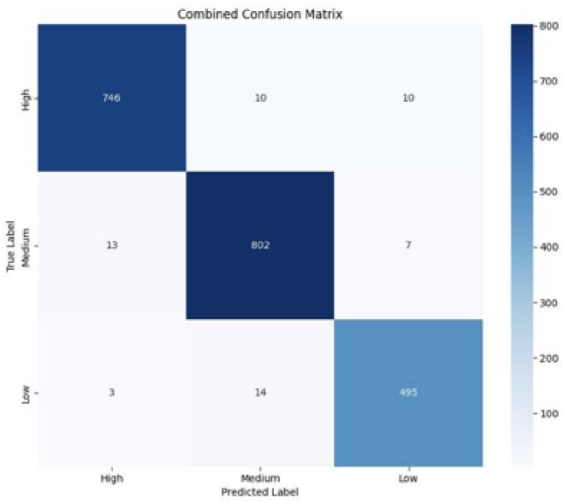

Fig 9. Combined Confusion Matrix Across All Batches

Summary and Implications

The machine learning component of CityPulse successfully demonstrates that synthetically generated traffic data can support the training of robust, high-performing congestion classifiers. Key takeaways include:

- High predictive accuracy and consistency across time
- Meaningful, interpretable features that mirror real-world traffic dynamics
- Robustness to noise and fluctuations in streaming input

This evaluation confirms that simulated environments can serve as valid proxies for training predictive traffic models, especially in urban areas where sensor coverage is sparse or inconsistent. The successful deployment of a Random Forest classifier on synthetic yet structurally realistic data further emphasizes the potential of data-centric AI systems to support smart city infrastructure in the absence of dense physical instrumentation.

This section closes the experimental validation phase of the study and sets the foundation for broader discussion on system generalizability, deployment feasibility, and the role of synthetic data in traffic intelligence pipelines.

6. Conclusion

This work introduced CityPulse, a scalable, real-time traffic congestion analytics pipeline tailored for data-scarce environments. Through the simulation of 11 million synthetic traffic records and the integration of modern open-source tools—Apache Kafka, Spark Structured Streaming, and Docker—we demonstrated that meaningful, near-real-time traffic insights can be generated without reliance on costly physical sensor infrastructure.

CityPulse exhibited high throughput (~320,000 records/minute) and low end-to-end latency (~3.2 seconds per 100,000 records) under stress, validating its operational efficiency and resilience. The modular, containerized architecture ensures fault tolerance, reproducibility, and scalability, making it suitable for deployment in constrained environments.

The system's Random Forest-based congestion classification module, trained on engineered features such as velocity and headway metrics, achieved macro F1-scores exceeding 0.95 across all congestion classes. These results highlight the potential of using synthetic data and lightweight ML models to deliver accurate, real-time traffic analysis without the need for extensive physical infrastructure.

By showing that such insights can be derived locally—using commodity hardware and open-source frameworks—CityPulse opens promising pathways for municipalities in developing regions to adopt data-driven urban mobility planning, leapfrogging traditional, sensor-heavy solutions.

Future Work:

To build on the current foundation and expand CityPulse's capabilities, the following short-term and long-term directions are proposed:

Short-Term Directions (6–12 months)

1. **Integration with Real-Time Sensors**: Incorporate real-world data streams from GPS-enabled devices, loop detectors, or CCTV feeds to enhance the realism and relevance of the pipeline's input.
2. **Edge and Lightweight Deployments**: Adapt the system for edge computing environments (e.g., Raspberry Pi, NVIDIA Jetson) to enable localized analytics without requiring central servers.
3. **Interactive Map-Based Visualization**: Implement a dynamic web dashboard featuring congestion overlays, historical playback, and anomaly alerts to improve decision-making for city planners and emergency services.
4. **Enhanced ML Modeling**: Expand the machine learning module to include multi-class classifiers and confidence-based outputs, improving interpretability and handling of edge cases.

Long-Term Directions (12–24 months+)

1. **Temporal Forecasting**: Introduce deep learning models such as LSTM, GRU, or Temporal Convolutional Networks (TCNs) to forecast traffic patterns over future time horizons, enabling proactive intervention rather than reactive analysis.
2. **Traffic Policy Simulation**: Extend the platform to test the effects of various traffic management policies—such as signal timing adjustments or road closures—using agent-based modeling, reinforcement learning, or what-if simulations.
3. **Traffic Rerouting and Recommendation Engine**: Integrate optimization algorithms that can suggest real-time route changes, detours, or scheduling adjustments for public transport and emergency response.
4. **Localization for Regional Deployment**: Focus on adapting CityPulse for specific urban geographies—such as cities in Sub-Saharan Africa—with regional road networks, localized language interfaces, and government-specific APIs to support real adoption.
5. **Scalable Cloud Deployment**: Expand cloud-readiness via platforms like Kubernetes, allowing

for elastic scaling across larger geographies and integration with smart city platforms or transportation control centers.

Author Contributions: *Idriss Djiofack Teledjieu* contributed to the conception, design, and overall supervision of the study. *Irzum Shafique* was responsible for data collection, analysis, and interpretation. *Yvan Nguemadje Peydhom* assisted in drafting the manuscript and reviewing the final version for intellectual content. All authors read and approved the final manuscript.

Originality and Ethical Standards: This manuscript is an original work that has not been published elsewhere and is not under consideration in any other journal. All authors have adhered to ethical standards of research, writing, and publication, and any sources or references have been properly acknowledged.

Data availability: The data supporting the findings of this study are available from the corresponding author upon reasonable request.

Conflict of Interest: The authors declare that they have no conflict of interest.

Ethical statement: This study was conducted in accordance with relevant ethical standards. No human participants or animals were involved in the research requiring ethical approval.

Funding: The research received no external funding.

Similarity checked: Yes.

References

[1] W. Tang, X. Lin, and Y. Liu, "Leveraging IoT data stream for near-real-time calibration of city-scale microscopic traffic simulation," *IET Smart Cities*, vol. 2, no. 4, pp. 205–213, Dec. 2020, doi: 10.1049/iet-smc.2020.0019.

[2] C. Chen, L. Yang, and H. Zhang, "SynTraC: A Synthetic Dataset for Traffic Signal Control from Traffic Monitoring Cameras," *arXiv preprint*, arXiv:2408.09588, Aug. 2024. [Online]. Available: https://arxiv.org/abs/2408.09588

[3] Cubig.ai, "The Critical Impact of Synthetic Data Utilization on Smart Cities: Opportunities and Challenges," *Cubig.ai Whitepaper*, Jul. 2024. [Online]. Available: https://cubig.ai/blogs/impact-of-synthetic-data-utilization-on-smart-cities

[4] G. Zimmermann, M. Troitzsch, and T. Iwanitz, "Towards Agent-Based Traffic Simulation Using Live Data from Sensors for Smart Cities," in *Multi-Agent-Based Simulation XX*, vol. 12584, Springer, 2021, pp. 33–47. doi: 10.1007/978-3-030-66888-4_3.

[5] A. G. Ismaeel, H. R. Abdulkareem, and K. R. Hassan, "Traffic Pattern Classification in Smart Cities Using Deep Recurrent Neural Network," *arXiv preprint*, arXiv:2401.13794, Jan. 2024. [Online]. Available: https://arxiv.org/abs/2401.13794

[6] Apache Foundation, "Apache Kafka," [Online]. Available: https://kafka.apache.org/;
Apache Foundation, "Apache Spark," [Online]. Available: https://spark.apache.org/;
Docker Inc., "Docker Documentation," [Online]. Available: https://docs.docker.com/

[7] Z. Huang et al., "Spatial-temporal correlation graph convolutional networks for traffic forecasting," IET Intelligent Transport Systems, 2023.

[8] Z. Ma et al., "Spatio-Temporal Heterogeneous Graph-Based Convolutional Networks for Traffic Flow Forecasting," Transportation Research Record, 2024.

[9] Y. Zhang et al., "Dynamic Spatio-Temporal Graph Fusion Convolutional Network for Urban Traffic Prediction," Applied Sciences, vol. 13, no. 16, 2023.

[10] C. Zheng et al., "Spatio-Temporal Joint Graph Convolutional Networks for Traffic Forecasting," arXiv preprint, arXiv:2111.13684, 2021.

[11] A. Roy et al., "Unified Spatio-Temporal Modeling for Traffic Forecasting using Graph Neural Network," arXiv preprint, arXiv:2104.12518, 2021.

[12] X. Liu et al., "Do We Really Need Graph Neural Networks for Traffic Forecasting?," arXiv preprint, arXiv:2301.12603, 2023.

[13] H. Chen, Y. Wang et al., "In-Depth with Spatial-Temporal Graph Neural Networks for Traffic Forecasting: An Overview with Attention," CITA 2024 Conference Proceedings, 2024.