Machine Learning-Based Anomaly Detection in Smart City Traffic: Performance Comparison and Insights

Mohammad Bawaneh[a] and Vilmos Simon[b]

Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary

Keywords: Anomaly Detection, Machine Learning, Traffic Congestion, Intelligent Transportation Systems, Smart City, Sustainability.

Abstract: In recent years, urban roads have suffered from substantial traffic congestion due to the rapidly increasing number of road users and vehicles. Some traffic congestion patterns on specific roadways, such as the recurring congestion during morning and evening rush hours, can be foreseen. However, unexpected events, such as incidents, may also cause traffic congestion. Monitoring traffic status poses vital importance for city traffic operators. They can leverage the monitoring system for resource allocation, traffic lights adjusting, and adapting the public transport schedules to alleviate traffic congestion. Machine learning-based methods for anomaly detection are valuable tools for monitoring traffic status and promptly detecting congestion on city roads. In this paper, we comprehensively study the performance of the common machine learning methods for anomaly detection in the traffic congestion detection use case. In addition, we provide methods usage insights based on the study findings by examining the accuracy, detection speed, and computation overhead of the methods to guide the researchers and city operators toward a suitable method based on their needs.

1 INTRODUCTION

In recent years, metropolitan cities have witnessed a significant increase in traffic congestion due to the increasing population in urban cities (González-Aliste et al., 2023). Therefore, the burden of traffic congestion has made a direct negative impact on the drivers and commuters' daily lives. In addition, traffic congestion significantly slows down the development of urban cities and weakens the efforts of achieving sustainable smart transport (Nguyen and Jung, 2021). A potential solution to reduce traffic congestion is to change the roads' infrastructure, such as opening new roads and widening the existing roads that witness high daily traffic. However, this solution is costly and sometimes inefficient (Zhu et al., 2021). In addition, it is not adaptable to different traffic situations where some roads are susceptible to frequent changes in traffic status. Therefore, Intelligent Transportation Systems (ITS) intend to find smart solutions to reduce traffic congestion without the need to change the roads' infrastructures to avoid the aforementioned shortcomings (Zhang et al., 2011).

ITS integrates information, data analytics, machine learning, and advanced communications technologies into one system that can monitor and manage the city's traffic in an automated way (Cheng et al., 2020). In order to monitor the traffic status and take actions to prevent or mitigate traffic congestion, the data-driven system must be capable of analyzing and understanding the underlying traffic patterns. The traffic data can be provided by vehicles' trajectories or by measurements collected by roadside sensors. The vehicles' trajectories can be collected using the Global Positioning System (GPS) technology, which collects the movement data of the equipped vehicle, such as the vehicle's speed and location per time (Sousa et al., 2020). On the other hand, roadside sensors, such as inductive loops and cameras, collect aggregated traffic data such as the traffic flow (vehicles' count), speed, and occupancy in close vicinity of the sensor's location (Tasgaonkar et al., 2020). By collecting this data, various artificial intelligence (AI) algorithms could be utilized to learn the patterns, which can help the city operators manage the traffic and introduce smart transportation services. Some of the algorithms that have already discovered its path to ITS are: Convolutional Neural Network (CNN) (Li

[a] https://orcid.org/0000-0003-4402-9254
[b] https://orcid.org/0000-0002-7627-3676

Bawaneh, M. and Simon, V.
Machine Learning-Based Anomaly Detection in Smart City Traffic: Performance Comparison and Insights.
DOI: 10.5220/0013141100003941
In *Proceedings of the 11th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2025)*, pages 309-318
ISBN: 978-989-758-745-0; ISSN: 2184-495X

et al., 2017), Graph Convolutional Neural Networks (GCNs) (Zheng et al., 2023), Long Short-Term Memory (LSTM) (Zhao et al., 2017), Gated Recurrent Units (GRUs) (Fu et al., 2016), Bidirectional-Long Short-Term Memory (Bi-LSTM) (Redhu et al., 2023), K-Means Clustering (Rao et al., 2019; Pustokhina et al., 2020), Autoencoders (Li et al., 2022), and Generative Adversarial Network (GAN) (Shi et al., 2024). AI algorithms can provide traffic prediction (Nagy and Simon, 2018; Lee et al., 2021), and traffic congestion detection and prediction services (Djenouri et al., 2019; Akhtar and Moridpour, 2021). This paper sheds light on automatizing the process of monitoring traffic status by employing the most common machine learning-based anomaly detection methods to detect unusual behavior in road traffic.

Numerous research papers have put forth algorithms for the detection of traffic congestion. The first category of these research papers relies on traffic data derived from the trajectories of vehicles, as presented by references (Pang et al., 2011; Pan et al., 2013; Kuang et al., 2015; Yu et al., 2017; Kong et al., 2018; Kohan and Ale, 2020; Zhang et al., 2021; Qin et al., 2021; He et al., 2023; Makara et al., 2023). However, it is worth noting that trajectory data, while easier and cheaper to collect compared to the costs of installing and operating sensor stations throughout the city network, comes with certain limitations. This data is typically collected by public transportation vehicles that have GPS devices, which operate on fixed routes and schedules, making them less precise in capturing citywide traffic congestion. Additionally, data from connected cars could offer more benefits, but the penetration of such vehicles remains low when compared to the overall vehicle population, and access to these datasets is often restricted due to their proprietary nature and high market prices. Navigation mobile apps also collect this kind of data; however, access to such datasets is restricted.

In the subsequent category of research papers from the field, aggregated data from roadside sensors is utilized. Various congestion detection definitions have been developed, including Travel Time Index (TTI) (Hasanzadeh et al., 2017), Traffic Condition (TC) (Fouladgar et al., 2017), and the Congestion Level (CL) (Chen et al., 2018). These definitions check and compare the current traffic state (i.e., speed or travel time) with a free-flow state and trigger a congestion alert if a significant deviation is observed. Other statistical-based algorithms have also been proposed, such as the ones mentioned in (Lam et al., 2017; Nguyen et al., 2016; Kalair and Connaughton, 2021; Bhardwaj et al., 2023; Xu and Li, 2024), which classify traffic states as congested when they deviate from a predefined null hypothesis. While statistical-based algorithms are efficient and straightforward to implement, they do not acquire correlations between different traffic states, potentially leading to more false congestion alerts. Several algorithms have introduced distance-based approaches, such as the K-Nearest Neighbors Outlier Detection (KNN OD) algorithm (Dang et al., 2015), the Bounded Local Outlier Factor (BLOF) algorithm (Tang and Ngan, 2016), and the Piecewise Switched Linear Traffic-Kalman Filter based K-Nearest Neighbors (PWSL-KF-based KNN) algorithm (Harrou et al., 2020). These methods employ similarity measures to detect dissimilar traffic states as congestion. However, KNN OD and PWSL-KF-based KNN may miss certain instances of congestion when the traffic data exhibits varying distribution densities. On the other hand, BLOF can handle traffic data with varying densities, thanks to its density-based outlier detection nature, but at the expense of higher computational complexity. Deep learning techniques have also been applied for traffic congestion detection, as seen in references (Zhu et al., 2019; Li et al., 2020; Liu et al., 2023). However, these algorithms often need more computational complexity, which hinders fast congestion detection, as they require multiple scans of the traffic data.

Our goal in this paper is to examine the performance of the most common machine learning methods for anomaly detection in the traffic congestion detection use case, as researchers, ITS developers, and city traffic operators could get easily lost in the abundance of these methods. We could not find a paper where the most essential methods are compared on the same dataset, using the same use case scenario and parameter settings, measuring the comprehensive performance metrics to show a thorough picture of the methods' performance. The papers from the literature usually validate only a few congestion detection methods in their performance testing, which are based on similar principles. Therefore, we have chosen methods that originate from different solution areas: the Modified Z-Score Method (MZ), the Isolation Forest (IF), the Local Outlier Factor (LOF), the Support Vector Machine (SVM), and the Long Short Term Memory (LSTM).

Many aspects are taken into account to provide a comprehensive overview of each method's performance. The methods are tested for their ability to provide reliable and accurate congestion detection. In addition, the speed of detecting the congestion is measured to verify the methods' effectiveness for the traffic management process. In other words, the congestion detection alarm must be triggered in the early phases of the congestion to be helpful if the traffic

management center wants to take actions to prevent or mitigate the propagation of the detected congestion to the neighboring roads (Nagy and Simon, 2021). Moreover, the computation overhead of running the methods is measured to test their suitability for edge computing, where the sensors collect the data and run the machine learning methods simultaneously to monitor the traffic.

The remainder of this research paper is organized as follows. Section 2 presents the anomaly detection methods used in this research, while Section 3 presents the experimental results, performance comparison, and insights. Finally, the paper is concluded in Section 4.

2 ANOMALY DETECTION METHODS OVERVIEW

This paper follows the second category of research papers in the literature, which utilizes aggregated traffic data (flow, speed, and occupancy) from roadside sensors to detect traffic congestion. In particular, this paper detects congestion by detecting anomalies in traffic time series data. The speed and occupancy traffic features are utilized as their behavior during the congestion period reflects an anomaly compared to the free-flow period. In the free-flow period (i.e., the normal behavior of the traffic), the speed values will be high. However, when the congestion starts (i.e., the abnormal traffic behavior), the drivers have to slow down their vehicles, which will be reflected in the low-speed values in the speed time series data. When the congestion ends, the speed values will return to the high values as the free-flow period. Consequently, abnormal traffic behavior can be found by finding the anomaly in the speed time series data, and hence, traffic congestion can be detected. Similarly, the occupancy values will be low during the free-flow traffic and high during the congestion period. Therefore, as the free-flow traffic is the normal behavior and the congestion is typically the abnormal behavior, detecting the anomaly in the occupancy time series leads to detecting the congestion. On the contrary, this is not the case for the flow traffic feature, as the anomaly in the flow time series does not ensure a congestion behavior. We refer the reader to the articles (Chakroborty and Das, 2017; Garber and Hoel, 2019; Bawaneh, 2023) for detailed information on the fundamental behavior of traffic flow data. Based on the explanation above of the behavior of traffic flow data, traffic congestion can be detected by studying the data and detecting the anomalies in the speed and occupancy time series.

This section provides an overview of the anomaly detection algorithms used in this paper to detect traffic congestion. They have been chosen in this paper because of their popularity and effectiveness for anomaly detection problems accompanied by easy and free access to code implementations, which facilitates the city operators' work of automating city traffic monitoring (Nassif et al., 2021; Scikit-learn, 2024; TensorFlow, 2024). Moreover, they represent different solution domains (statistical, machine learning, and deep learning), which leads to a more comprehensive study.

2.1 Modified Z-Score Method

The modified Z-score (MZ) method is a simple yet effective statistical technique for finding outliers or anomalies within a dataset (Blaise et al., 2020). It represents a variation of the traditional Z-score method. The traditional z-score method measures how many standard deviations a particular data point deviates from the mean of the dataset. It is computed by subtracting the mean from the data point's value and dividing the result by the value of the standard deviation. The computed z-score is then thresholded to classify the data point as an outlier or normal value. Its variation (i.e., the modified z-score method) works with a similar principle. However, the modified version uses the median and the median absolute deviation values instead of the mean and the standard deviation. This modification provides the method with more robustness against extreme values, as the mean and the standard deviation can be significantly affected by the extreme values. In addition, the modified z-score method does not require a particular data distribution, in contrast to the traditional one, which assumes a normal distribution for the data.

2.2 Isolation Forest

Isolation Forest is a machine learning algorithm used for anomaly detection (Liu et al., 2008; Togbe et al., 2020; Chen et al., 2020; Liu et al., 2021). Isolation Forest works by recursively randomly partitioning the dataset. It selects a random split point at each step. This randomness in partitioning provides the algorithm with more effectiveness. The algorithm constructs a binary tree collection called Isolation Trees. Each tree is constructed by recursively partitioning the data until it isolates individual data points. The isolation process involves randomly selecting split points to create splits in the data. Data points that are isolated quickly (with fewer splits) are likely to be anomalies, while those that require many splits are

considered normal. The anomaly score for each data point is determined by the path length required to isolate that point across all the trees in the forest. Data points that have shorter-than-average path lengths are more likely to be anomalies, as they were easier to isolate. A threshold is set to identify anomalies. Data points with path lengths less than this threshold are considered anomalies, while those over the threshold are considered normal.

2.3 Local Outlier Factor

The Local Outlier Factor (LOF) is a machine learning algorithm used for identifying local anomalies or outliers within a dataset (Breunig et al., 2000; Djenouri et al., 2018; Šabić et al., 2021). LOF assesses the relative density of data points with respect to their local neighborhoods to detect regions of the data where points are significantly less dense than their neighbors. LOF starts by estimating the local density of each data point within its neighborhood. A neighborhood is defined around each data point by considering its k nearest neighbors, where k is a pre-defined parameter. The density of a data point is measured by considering the average density of its k nearest neighbors. Data points in densely populated regions will have a higher local density, while those in sparser regions will have a lower local density. For each data point, LOF calculates its local outlier factor by comparing its local density to the local densities of its neighbors. The LOF of a point is defined as the ratio of its own local density to the average local density of its k nearest neighbors. LOF values provide a measure of how much a data point deviates from its local neighborhood's density. Data points with high LOF values are considered local outliers because they have significantly lower densities than their neighbors. Setting a threshold can determine which points are considered outliers based on their LOF values. Points with LOF values above the threshold are identified as outliers.

2.4 Support Vector Machine

A One-Class Support Vector Machine (One-Class SVM) is a machine learning algorithm used for anomaly detection. It is a special use case of the Support Vector Machine (Hearst et al., 1998; Pisner and Schnyer, 2020; Wang et al., 2020; Ioannou and Vassiliou, 2021). It is designed to detect anomalies by learning the characteristics of the majority class (normal instances) and then flagging any instances that deviate significantly from this majority class. One-Class SVM trains on a dataset that contains mostly normal instances. The algorithm's goal is to create a decision boundary (or hyperplane) that covers the majority of the normal data points while minimizing the number of anomalies within this boundary. The hyperplane is positioned in a way that maximizes the margin between the hyperplane and the normal data points. This margin represents the region within which most normal instances are expected to lie. During training, the algorithm identifies a subset of data points known as support vectors. These are the data points closest to the hyperplane and are essential for defining the margin. The margin is the distance between the hyperplane and the closest support vectors. One-Class SVM aims to maximize this margin while minimizing the number of support vectors. Once trained, One-Class SVM can classify new data points into normal and anomaly. Normal data points fall within the margin, while anomaly data points fall outside the margin.

2.5 Long Short Term Memory

Long Short Term Memory (LSTM) (a type of Recurrent Neural Network (RNN)) is a deep learning method that is well-suited for modeling sequential data due to its ability to capture long-term dependencies (i.e., avoiding the vanishing gradient problem), which was a shortcoming in the traditional RNN (Hochreiter and Schmidhuber, 1997; Tan et al., 2020; Markovic et al., 2023). The LSTM model works by integrating units. The unit consists of a cell state, forget gate, input gate, and output gate. The cell state represents the long-term memory, which can be controlled by the forget, input, and output gates. The forget gate controls the percentage of the memory in the cell state that should be kept or removed. The input gate controls the addition or the update of the cell state based on the new information. The output gate controls which information should be the output of the unit. The units are connected sequentially, forming a chain of LSTM units. For each unit, an input is received from the actual current input value in addition to the previous output value of the previous unit. These connections provide an effective process of sequential data, where the dependencies are captured over time. Due to the recurrent nature of such a structure, the information is propagated across the network. Consequently, LSTM can learn complex temporal patterns and dependencies. LSTM does not detect the anomalies in the data directly. It can be used to train a model to forecast time series data. Once the LSTM model is trained, it can be used to make predictions on new unseen data. Anomalies are typically detected by comparing the model's predictions to the actual data points. A high deviation be-

tween the predicted and the actual values indicates an anomaly. Therefore, a threshold must be trained to detect the anomalies points.

Considering the space limitation of the paper, a brief overview of the anomaly detection methods utilized has been presented. We refer the readers to the original papers for more details and mathematical descriptions of the methods. As mentioned previously, prior studies typically focus on validating a limited subset of anomaly detection methods, often based on similar principles (for example, statistical-based methods). However, in this paper, a diverse array of methods spanning different solution domains has been chosen for this study to assess their efficiency for traffic congestion detection. We have chosen popular statistical-based, machine learning-based, and deep learning-based methods to cover widespread solution domains. Despite the spread of such methods for solving other research problems, existing literature lacks a comprehensive comparison conducted under standardized conditions (i.e., using the same dataset, scenario, and parameter settings) and covering the overall performance points of view, enabling a thorough evaluation of their performance for traffic congestion detection. The next section of this research not only offers a comprehensive understanding of their comparative performance but also provides effective practical recommendations for optimizing congestion detection strategies. This study thus serves as a pivotal resource for enhancing the effectiveness of traffic management systems, which contributes to achieving safer, sustainable, and more efficient urban mobility.

3 EXPERIMENTAL RESULTS, PERFORMANCE COMPARISON, AND INSIGHTS

In the previous section, several methods for anomaly detection were presented briefly. These methods have been successfully used for various domains and have shown superior performance (Blaise et al., 2020; Chen et al., 2020; Šabić et al., 2021; Wang et al., 2020; Markovic et al., 2023). However, their performance for traffic congestion detection tasks was not previously compared with each other. Therefore, taking into account the diversity in the solution domains, a comprehensive comparison that shows the entire performance picture under unified and standardized conditions is presented in this section.

As mentioned previously, this paper provides an experimental verification of employing these methods for the traffic congestion detection use case. The methods are tested for accuracy, detection speed, and computation complexity. Consequently, this paper provides the needed comprehensive evaluation to present the complete picture of the performance of each method based on the traffic management operators' needs. The following subsection presents the real-life acquired dataset utilized in this paper to conduct the experiment.

3.1 The Utilized Dataset

The California Department of Transportation provides the researchers with the Caltrans Performance Measurement System (PeMS) real-time collected traffic dataset (PeMS, 2020). The PeMS dataset also provides the incident information. This is an advantage of utilizing it to conduct this paper's experiment because the incident usually causes congestion.

A separate traffic incident dataset has been extracted by Nagy et al. (Nagy et al., 2021; Nagy and Simon, 2020) from the PeMS dataset. The extracted data covers one year of the traffic dataset (i.e., from January 1, 2016, to December 31, 2016). It contains incidents that happened in different locations in District 3. The aggregation period is 5 minutes, which means that every sample shows the traffic information (speed, occupancy, or flow) for 5 minutes. The total traffic information is collected in a PARQUET file with a size of 183 MB. The traffic incident dataset has been extracted to contain only the incidents that caused congestion on the roads, according to (Nagy et al., 2021). This makes it suitable to conduct the experiment on traffic congestion detection.

This paper is experimenting with 100 incidents from the dataset that occurred in 2016, where 50% of the dataset was used for training and 50% for testing. The incidents have occurred in different locations, which helps to verify the robustness of the tested anomaly detection methods. We have tested the methods on 12 hours of traffic for each incident, where the congestion happened in the middle of this period. Python programming language has been used to implement all the methods. As explained in section 2, and based on the fundamental behavior of traffic congestion, the speed and occupancy traffic features have been used in this paper. All the methods have been tested by using both of them in order to evaluate their performance in both cases.

3.2 Experimental Results

As an initial step, common to all these algorithms, we perform data cleaning and preprocessing, a necessary

step as the data often contains noise and may have some missing values for certain samples. In the data cleaning process, we employ the forward-fill method to fill in any missing sample values. Subsequently, we apply the exponential moving average technique to mitigate noise and to smooth the traffic data. To illustrate, for a specific traffic feature such as road speed, flow, or occupancy, we define a time series denoted as X with n observations in the following manner:

$$X = (X_1, X_2, ..., X_n) \quad (1)$$

where the time sequence in which the observations (speed or occupancy) were captured by a certain sensor is indicated by the indices $(1, 2, ..., n)$. The definition of the exponential moving average is:

$$T_t = \begin{cases} X_1 & t = 1 \\ \alpha \cdot X_t + (1-\alpha) \cdot T_{t-1} & t > 1 \end{cases} \quad (2)$$

Each of the utilized methods has its own parameters that must be configured. A common example of these parameters between the utilized methods is the threshold that classifies the sample point as normal or anomaly point by thresholding the anomaly score obtained by the anomaly detection method. The outcome can be significantly affected by the choice of these parameters. Hence, in order to conduct a fair comparison, all methods must be tuned to achieve their best performance, which can be achieved by applying the method with the best parameter configuration. Consequently, to find the optimal parameters for each method, the dataset under the study is split into a training dataset and a testing dataset, where the training dataset is used to find the optimal parameters that can achieve the highest performance of the method, which can be measured by the used performance metrics in this paper. Grid search has been used to obtain the optimal parameters for all the utilized methods. The determined optimal parameters are after applied to the testing dataset to measure and conclude the method's performance. This ensures that all the methods used have been fairly tuned to achieve the best possible results, which in turn ensures a fair comparison. In addition, we note that separate learning for the parameters has been done for each traffic feature (i.e., speed and flow).

The experiment has been conducted by employing the testing dataset to evaluate the efficiency of each anomaly detection method for the traffic congestion detection use case. The standard performance evaluation metrics in the literature for evaluating traffic congestion detection methods have been used in this study (AlDhanhani et al., 2019; Kalair and Connaughton, 2021). In addition to these metrics, we have compared the run time of each method to compare their computation complexity and suitability for edge computing. The used metrics are defined as follows:

- Detection Rate (DR): The percentage of congestions that have been correctly detected.
- False Alarm Rate (FAR): The percentage of false congestion alarms to the number of congestion-free instances.
- Mean Time to Detection (MTTD): The average time needed to detect the congestions (i.e., the detection speed). It is measured based on the difference between the timestamp where the congestion occurred and the timestamp where the congestion has been detected by the method.
- Runtime: The computation run time needed by the method to conduct the experiment.

Table 1 shows the experimental results of this study for all the utilized methods.

Table 1: Performance Comparison: the outcome of the experiments of this study in terms of accuracy and time.

Method	Traffic Feature	DR	FAR	MTTD (minutes)	Runtime (seconds)
MZ	Speed	94%	6.8%	9.3	1.8
MZ	Occupancy	96%	17%	7.5	1
IF	Speed	94%	7.4%	5.6	13.9
IF	Occupancy	90%	5.5%	4	13.6
LOF	Speed	98%	21%	2.8	1.45
LOF	Occupancy	88%	10%	8	1.2
SVM	Speed	98%	6.5%	5.6	1.5
SVM	Occupancy	94%	4%	5.2	5.1
LSTM	Speed	96%	11.7%	14	10397
LSTM	Occupancy	96%	18%	7.9	11284

3.3 Discussion and Recommendations

Based on Table 1, it can be generalized that high DR can be achieved with most of the methods. Therefore, the anomaly detection methods (MZ, IF, LOF, SVM, and LSTM) have verified their ability to achieve a reliable system for detecting congestion. Typically, the speed readings during the congestion period will reach a significant drop compared to the free flow readings, which makes it easier for the anomaly detection methods to capture the congestion. As a consequence, the methods have shown better performance using the speed traffic feature than the occupancy in terms of DR. The best performance has been achieved by LOF and SVM using the speed traffic feature with a DR of 98%.

In terms of FAR, which reflects the robustness of the method to the noise and false alarms, SVM and IF have shown better performance than the rest of the competing methods.

The speed of the detection has been found to vary between the methods. The fastest method to detect the congestion was LOF using the speed traffic feature, achieving only 2.8 minutes of delay. However, a high delay occurred when LOF used the occupancy traffic feature. Such behavior may happen as each traffic feature explains the traffic from a different point of view. Therefore, one traffic feature may reflect the congestion behavior in the data before the other traffic feature (Bawaneh and Simon, 2023). In general, SVM and IF have achieved acceptable MTTD using both features (the speed and occupancy traffic features). On the other hand, LSTM has shown the highest delay among the anomaly detection methods when using the speed traffic feature.

In this study, we added another important metric to the comparison: the runtime of the algorithms on the utilized dataset. This metric measures the computation complexity of the methods and indicates the computation resources needed to utilize these methods. It can be noticed in Table 1 that all methods required roughly low time to conduct the experiment except for LSTM. LSTM required around 3 hours (more or less) to run the calculations, while the highest runtime needed by the rest of the methods was 13.9 seconds.

Traffic control operators may utilize traffic congestion detection for different needs, such as utilizing the congestion alarm information to adjust traffic lights to reduce the effect of the congestion on the neighboring areas (Cao et al., 2016; Tseng and Ferng, 2021). These needs require different performance standards. Thus, we suggest in this paper three main scenarios to provide guidance for traffic control operators according to the efficiency of each method:

- Scenario I: Traffic management center operators intervene to reduce the effect of the congestion by adjusting traffic lights.
- Scenario II: Traffic management center operators provide a city mobility application to the city-dwellers to suggest routes to avoid congestions.
- Scenario III: Traffic management center operators apply edge computing (i.e., the roadside sensors collect the data and run the traffic congestion detection locally in parallel).

The traffic congestion detection system can be considered reliable when it can achieve high DR. This means that the system is more likely not to miss detecting any congestion. Consequently, traffic management centers can safely deploy and rely on such an automated system. The three scenarios require high DR to provide system reliability. However, if this system suffers from a high number of false alarms, actions that are not needed might be taken with the intention to mitigate the wrongly detected congestion effect on the roads network (for example, wrong resource allocation). The effect of this is highly dependent on how the traffic congestion detection system is utilized. For example, suppose the traffic congestion detection system is needed only for monitoring purposes, such as providing a city mobility application to provide road users with traffic status information (Scenario II). In that case, the negative impact of a high FAR can be dealt with. On the other hand, if the system is meant to be utilized to intervene to mitigate the effect of congestion, the cost of a high FAR will be significant. On one side, it will overload the work of the traffic management centers with unnecessary actions, and on the other side, it might lead to undesired events, such as causing real traffic congestion or resource wasting. An example of that is if wrong decisions have been made to adjust the traffic lights (Scenario I) while there was actually no congestion, congestion might occur as a result of such wrong adjusting. Therefore, low FAR is recommended for the chosen system to prevent such effects.

From another point of view, the detection speed is a crucial metric for all Scenarios. Fast detection (i.e., detection with low delay) eases the congestion effects mitigation work of the traffic operators by providing them with sufficient time to take proper actions. These actions may include adapting the traffic lights, sending suggestions for lighter traffic routes for the drivers, or increasing the operated public transport services.

Last but not least, if edge computing is to be employed in smart city infrastructure (Scenario III), the computation complexity of the used method is a key factor. Even without adopting edge computing and parallelizing the computations, traffic control centers in major cities with thousands of measurement points can become overloaded with these calculations. For example, deep learning-based methods (such as LSTM) are time-consuming, and hence, they cannot fulfill a low computation complexity to support edge computing. This can be seen in the performance of LSTM, which confirms the time-consuming limitation of deep learning methods in general, even if they can achieve powerful results. The recorded runtime shows that the other methods can support the new era of Tiny Machine Learning (TinyML), where machine learning models are deployed on devices with limited computational resources (such as roadside units). TinyML allows to make intelligent decisions locally without relying on cloud infrastructure (or traffic control centers) by running lightweight and efficient machine learning algorithms directly on

the roadside units, which enables better real-time processing, lower latency, increased privacy, and reduced reliance on cloud infrastructure. Table 2 summarizes the ability of each method to support each Scenario.

Table 2: A summary of the ability of the anomaly detection methods to support each Scenario.

Method	Traffic Feature	Scenario I	Scenario II	Scenario III
MZ	Speed		✓	✓
MZ	Occupancy			✓
IF	Speed	✓	✓	✓
IF	Occupancy	✓	✓	✓
LOF	Speed			✓
LOF	Occupancy		✓	✓
SVM	Speed	✓	✓	✓
SVM	Occupancy	✓	✓	✓
LSTM	Speed			
LSTM	Occupancy			

4 CONCLUSION

In this research paper, we have studied the performance of the most commonly used techniques for identifying anomalies in the context of traffic congestion detection. Statistical-based, machine-learning-based, and deep learning-based techniques have been chosen to cover diverse solution domains. In addition, various factors have been considered to offer a comprehensive assessment of each method's performance. Conducted under standardized conditions, the study outcomes have shown that the methods have varying efficiency for different needs. Therefore, this study can help researchers and city operators choose the optimal method based on their needs, such as the ability to intervene to reduce the effect of congestion by adjusting traffic lights and the ability to support edge computing. Three different scenarios for traffic management centers have been proposed in this paper. The experimental results have shown that IF and SVM can support all three scenarios with varying performance. However, the city operators can also use the results to assess the methods for other needs by studying their comprehensive performance in terms of DR, FAR, MTTD, and runtime. This allows for an assessment from all possible points of view. In general, the experimental results have shown that all the utilized methods can achieve relatively high DR. However, regarding FAR, MTTD, and runtime, varying efficiency has been found. Hence, the latter metrics are key factors in choosing the suitable method for particular requirements.

REFERENCES

Akhtar, M. and Moridpour, S. (2021). A review of traffic congestion prediction using artificial intelligence. *Journal of Advanced Transportation*, 2021:1–18.

AlDhanhani, A., Damiani, E., Mizouni, R., and Wang, D. (2019). Framework for traffic event detection using shapelet transform. *Engineering Applications of Artificial Intelligence*, 82:226–235.

Bawaneh, M. (2023). *Public Service Optimization in Automated Cities*. PhD thesis, Budapest University of Technology and Economics.

Bawaneh, M. and Simon, V. (2023). Novel traffic congestion detection algorithms for smart city applications. *Concurrency and Computation: Practice and Experience*, 35(5):e7563.

Bhardwaj, A., Iyer, S. R., Ramesh, S., White, J., and Subramanian, L. (2023). Understanding sudden traffic jams: From emergence to impact. *Development Engineering*, 8:100105.

Blaise, A., Bouet, M., Conan, V., and Secci, S. (2020). Detection of zero-day attacks: An unsupervised port-based approach. *Computer Networks*, 180:107391.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104.

Cao, Z., Jiang, S., Zhang, J., and Guo, H. (2016). A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion. *IEEE transactions on intelligent transportation systems*, 18(7):1958–1973.

Chakroborty, P. and Das, A. (2017). *Principles of transportation engineering*. PHI Learning Pvt. Ltd.

Chen, H., Ma, H., Chu, X., and Xue, D. (2020). Anomaly detection and critical attributes identification for products with multiple operating conditions based on isolation forest. *Advanced Engineering Informatics*, 46:101139.

Chen, M., Yu, X., and Liu, Y. (2018). Pcnn: Deep convolutional networks for short-term traffic congestion prediction. *IEEE Transactions on Intelligent Transportation Systems*, 19(11):3550–3559.

Cheng, Z., Pang, M.-S., and Pavlou, P. A. (2020). Mitigating traffic congestion: The role of intelligent transportation systems. *Information Systems Research*, 31(3):653–674.

Dang, T. T., Ngan, H. Y., and Liu, W. (2015). Distance-based k-nearest neighbors outlier detection method in large-scale traffic data. In *2015 IEEE International Conference on Digital Signal Processing (DSP)*, pages 507–510. IEEE.

Djenouri, Y., Belhadi, A., Lin, J. C.-W., Djenouri, D., and Cano, A. (2019). A survey on urban traffic anomalies detection algorithms. *IEEE Access*, 7:12192–12205.

Djenouri, Y., Zimek, A., and Chiarandini, M. (2018). Outlier detection in urban traffic flow distributions. In *2018 IEEE international conference on data mining (ICDM)*, pages 935–940. IEEE.

Fouladgar, M., Parchami, M., Elmasri, R., and Ghaderi, A. (2017). Scalable deep traffic flow neural networks for urban traffic congestion prediction. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2251–2258. IEEE.

Fu, R., Zhang, Z., and Li, L. (2016). Using lstm and gru neural network methods for traffic flow prediction. In *2016 31st Youth academic annual conference of Chinese association of automation (YAC)*, pages 324–328. IEEE.

Garber, N. J. and Hoel, L. A. (2019). *Traffic and highway engineering*. Cengage Learning.

González-Aliste, P., Derpich, I., and López, M. (2023). Reducing urban traffic congestion via charging price. *Sustainability*, 15(3):2086.

Harrou, F., Zeroual, A., and Sun, Y. (2020). Traffic congestion monitoring using an improved knn strategy. *Measurement*, 156:107534.

Hasanzadeh, A., Liu, X., Duffield, N., Narayanan, K. R., and Chigoy, B. (2017). A graph signal processing approach for real-time traffic prediction in transportation networks. *arXiv preprint arXiv:1711.06954*.

He, Y., Hofer, B., Sheng, Y., Yin, Y., and Lin, H. (2023). Processes and events in the center: a taxi trajectory-based approach to detecting traffic congestion and analyzing its causes. *International Journal of Digital Earth*, 16(1):509–531.

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ioannou, C. and Vassiliou, V. (2021). Network attack classification in iot using support vector machines. *Journal of sensor and actuator networks*, 10(3):58.

Kalair, K. and Connaughton, C. (2021). Anomaly detection and classification in traffic flow data from fluctuations in the flow–density relationship. *Transportation Research Part C: Emerging Technologies*, 127:103178.

Kohan, M. and Ale, J. M. (2020). Discovering traffic congestion through traffic flow patterns generated by moving object trajectories. *Computers, Environment and Urban Systems*, 80:101426.

Kong, X., Song, X., Xia, F., Guo, H., Wang, J., and Tolba, A. (2018). Lotad: Long-term traffic anomaly detection based on crowdsourced bus trajectory data. *World Wide Web*, 21(3):825–847.

Kuang, W., An, S., and Jiang, H. (2015). Detecting traffic anomalies in urban areas using taxi gps data. *Mathematical Problems in Engineering*, 2015.

Lam, P., Wang, L., Ngan, H. Y., Yung, N. H., and Yeh, A. G. (2017). Outlier detection in large-scale traffic data by naïve bayes method and gaussian mixture model method. *Electronic Imaging*, 2017(9):73–78.

Lee, K., Eo, M., Jung, E., Yoon, Y., and Rhee, W. (2021). Short-term traffic prediction with deep neural networks: A survey. *IEEE Access*, 9:54739–54756.

Li, L., Lin, Y., Du, B., Yang, F., and Ran, B. (2020). Real-time traffic incident detection based on a hybrid deep learning model. *Transportmetrica A: Transport Science*, pages 1–21.

Li, L., Lin, Y., Du, B., Yang, F., and Ran, B. (2022). Real-time traffic incident detection based on a hybrid deep learning model. *Transportmetrica A: transport science*, 18(1):78–98.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

Liu, D., Zhen, H., Kong, D., Chen, X., Zhang, L., Yuan, M., and Wang, H. (2021). Sensors anomaly detection of industrial internet of things based on isolated forest algorithm and data compression. *Scientific Programming*, 2021:1–9.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE.

Liu, Y., Cai, Z., and Dou, H. (2023). Highway traffic congestion detection and evaluation based on deep learning techniques. *Soft Computing*, 27(17):12249–12265.

Makara, L. A., Maric, P., and Pekar, A. (2023). Public transport congestion detection using incremental learning. *Pervasive and Mobile Computing*, 91:101769.

Markovic, T., Dehlaghi-Ghadim, A., Leon, M., Balador, A., and Punnekkat, S. (2023). Time-series anomaly detection and classification with long short-term memory network on industrial manufacturing systems. In *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*, pages 171–181. IEEE.

Nagy, A. M. and Simon, V. (2018). Survey on traffic prediction in smart cities. *Pervasive and Mobile Computing*, 50:148–163.

Nagy, A. M. and Simon, V. (2020). Traffic incident dataset. https://gitlab.medianets.hu/anagy/incident_dataset. Accessed: 2024-08-30.

Nagy, A. M. and Simon, V. (2021). A novel congestion propagation modeling algorithm for smart cities. *Pervasive and Mobile Computing*, 73:101387.

Nagy, A. M., Wiandt, B., and Simon, V. (2021). Transient-based automatic incident detection method for intelligent transport systems. *Infocommunications Journal*, 13(3):2–13.

Nassif, A. B., Talib, M. A., Nasir, Q., and Dakalbab, F. M. (2021). Machine learning for anomaly detection: A systematic review. *Ieee Access*, 9:78658–78700.

Nguyen, H., Liu, W., and Chen, F. (2016). Discovering congestion propagation patterns in spatio-temporal traffic data. *IEEE Transactions on Big Data*, 3(2):169–180.

Nguyen, T.-H. and Jung, J. J. (2021). Swarm intelligence-based green optimization framework for sustainable transportation. *Sustainable Cities and Society*, 71:102947.

Pan, B., Zheng, Y., Wilkie, D., and Shahabi, C. (2013). Crowd sensing of traffic anomalies based on human mobility and social media. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 344–353. ACM.

Pang, L. X., Chawla, S., Liu, W., and Zheng, Y. (2011). On mining anomalous patterns in road traffic streams. In *International conference on advanced data mining and applications*, pages 237–251. Springer.

PeMS (2020). Caltrans Performance Measurement System. https://pems.dot.ca.gov/. Accessed: 2024-9-15.

Pisner, D. A. and Schnyer, D. M. (2020). Support vector machine. In *Machine learning*, pages 101–121. Elsevier.

Pustokhina, I. V., Pustokhin, D. A., Rodrigues, J. J., Gupta, D., Khanna, A., Shankar, K., Seo, C., and Joshi, G. P. (2020). Automatic vehicle license plate recognition using optimal k-means with convolutional neural network for intelligent transportation systems. *Ieee Access*, 8:92907–92917.

Qin, J., Mei, G., and Xiao, L. (2021). Building the traffic flow network with taxi gps trajectories and its application to identify urban congestion areas for traffic planning. *Sustainability*, 13(1):266.

Rao, W., Xia, J., Lyu, W., and Lu, Z. (2019). Interval data-based k-means clustering method for traffic state identification at urban intersections. *IET Intelligent Transport Systems*, 13(7):1106–1115.

Redhu, P., Kumar, K., et al. (2023). Short-term traffic flow prediction based on optimized deep learning neural network: Pso-bi-lstm. *Physica A: Statistical Mechanics and its Applications*, 625:129001.

Šabić, E., Keeley, D., Henderson, B., and Nannemann, S. (2021). Healthcare and anomaly detection: using machine learning to predict anomalies in heart rate data. *AI & SOCIETY*, 36(1):149–158.

Scikit-learn (2024). Scikit-Learn (Machine Learning in Python). https://scikit-learn.org/. Accessed: 2024-07-27.

Shi, H., Dong, S., Wu, Y., Nie, Q., Zhou, Y., and Ran, B. (2024). Generative adversarial network for car following trajectory generation and anomaly detection. *Journal of Intelligent Transportation Systems*, pages 1–14.

Sousa, R. S. D., Boukerche, A., and Loureiro, A. A. (2020). Vehicle trajectory similarity: models, methods, and applications. *ACM Computing Surveys (CSUR)*, 53(5):1–32.

Tan, Y., Hu, C., Zhang, K., Zheng, K., Davis, E. A., and Park, J. S. (2020). Lstm-based anomaly detection for non-linear dynamical system. *IEEE access*, 8:103301–103308.

Tang, J. and Ngan, H. Y. (2016). Traffic outlier detection by density-based bounded local outlier factors. *Information Technology in Industry*, 4(1):6.

Tasgaonkar, P. P., Garg, R. D., and Garg, P. K. (2020). Vehicle detection and traffic estimation with sensors technologies for intelligent transportation systems. *Sensing and Imaging*, 21:1–28.

TensorFlow (2024). TensorFlow: An end-to-end platform for machine learning. https://www.tensorflow.org/. Accessed: 2024-07-27.

Togbe, M. U., Barry, M., Boly, A., Chabchoub, Y., Chiky, R., Montiel, J., and Tran, V.-T. (2020). Anomaly detection for data streams based on isolation forest using scikit-multiflow. In *Computational Science and Its Applications–ICCSA 2020: 20th International Conference, Cagliari, Italy, July 1–4, 2020, Proceedings, Part IV 20*, pages 15–30. Springer.

Tseng, Y.-T. and Ferng, H.-W. (2021). An improved traffic rerouting strategy using real-time traffic information and decisive weights. *IEEE Transactions on Vehicular Technology*, 70(10):9741–9751.

Wang, Y., Masoud, N., and Khojandi, A. (2020). Real-time sensor anomaly detection and recovery in connected automated vehicle sensors. *IEEE transactions on intelligent transportation systems*, 22(3):1411–1421.

Xu, M. and Li, J. (2024). High-efficiency anomaly detection of traffic data stream using sequential bi-iteration svd. *IAENG International Journal of Computer Science*, 51(2).

Yu, Y., Cao, L., Rundensteiner, E. A., and Wang, Q. (2017). Outlier detection over massive-scale trajectory streams. *ACM Trans. Database Syst.*, 42(2).

Zhang, J., Wang, F.-Y., Wang, K., Lin, W.-H., Xu, X., and Chen, C. (2011). Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639.

Zhang, X., Zheng, Y., Zhao, Z., Liu, Y., Blumenstein, M., and Li, J. (2021). Deep learning detection of anomalous patterns from bus trajectories for traffic insight analysis. *Knowledge-Based Systems*, 217:106833.

Zhao, Z., Chen, W., Wu, X., Chen, P. C., and Liu, J. (2017). Lstm network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2):68–75.

Zheng, G., Chai, W. K., Duanmu, J.-L., and Katos, V. (2023). Hybrid deep learning models for traffic prediction in large-scale road networks. *Information Fusion*, 92:93–114.

Zhu, L., Krishnan, R., Sivakumar, A., Guo, F., and Polak, J. W. (2019). Traffic monitoring and anomaly detection based on simulation of luxembourg road network. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 382–387. IEEE.

Zhu, Q., Liu, Y., Liu, M., Zhang, S., Chen, G., and Meng, H. (2021). Intelligent planning and research on urban traffic congestion. *Future Internet*, 13(11):284.