Noname manuscript No.
(will be inserted by the editor)

Detection and Classification of Sensor Anomalies for Simulating Urban Traffic Scenarios

Chiara Bachechi · Federica Rollo · Laura Po

Received: date / Accepted: date

Abstract Sensor network infrastructures are widely used in smart cities to monitor and analyze urban traffic flow. Starting from punctual information coming from traffic sensor data, traffic simulation tools are used to create the digital twin" mobility data model that helps local authorities to better understand urban mobility. However, sensors can be faulty and errors in sensor data can be propagated to the traffic simulations, leading to erroneous analysis of the traffic scenarios. Providing real-time anomaly detection for time series data streams is highly valuable since it enables to automatically recognize and discard or repair sensor faults in time-sensitive processes.

In this paper, we implement a data cleaning process that detects and classifies traffic anomalies distinguishing between sensor faults and unusual traffic conditions, and removes sensor faults from the input of the traffic simulation model, improving its performance. Experiments conducted on a real scenario for 30 days have demonstrated that anomaly detection coupled with anomaly classification boosts the performance of the traffic model in emulating real urban traffic.

1 Introduction

Anomaly investigation is an important element in many application domains such as fault detection, privacy and cybersecurity, communication networks, and social media. In a complex system, regular behavior allows us to investigate the general characteristics of the system, while anomalies let us discover and analyze unexpected behavior, as significant events and patterns, that otherwise could not be discovered. For this reason, anomaly detection and classification are universally recognized and have become very important in data analysis [1].

In the smart city context, through deploying Internet of Things (IoT) technologies, many aspects of the urban environment can be monitored in real-time such as mobility, pollution, parking, waste, lighting. Although the technology has evolved greatly and has allowed for a drastic drop in costs and a wide variety of sensors available, sensors are prone to malfunctions. Therefore, the big sensor data streams generated in real-time need to be handled with appropriate techniques to detect erroneous measurements instantly.

Analysis of traffic data is an essential component of intelligent transportation system applications crucial for smart cities. Traffic data collected through sensors such as induction loop detectors often contain anomalies, e.g. due to malfunctioning detectors or anomalous traffic conditions. Such anomalies can heavily affect the results of the subsequent analysis like traffic flow analysis, monitoring, and prediction. There are several challenges regarding anomaly detection, including the absence of a universal definition of anomaly, change of traffic pattern over time, as well as unavailability of labeled data. Traffic models are essential tools for road traffic analysis and simulation of urban mobility and the deployment of intelligent transportation system applications. Through a traffic model, real-time simulations can be performed using as input the traffic sensors data to emulate the traffic flow in the entire urban context. However, the delay between the acquisition of sensor data and the generation of the model output should be reduced as much as possible. Therefore, the detection of anomalies in the input has to be done in a short time.

'Enzo Ferrari' Engineering Department
University of Modena and Reggio Emilia
Modena, Italy
E-mail: name.surname@unimore.it

In this paper, a methodology of anomaly detection and classification on traffic time series data streams is proposed. This method does not assume labeled historical data and can be applied in real-time. Since observations coming from traffic sensors are often used as input of customized traffic models that simulate urban traffic flows in real-time, this paper also examines and discusses the improvement provided by the anomaly detection and classification method by comparing the traffic model outputs, considering or excluding sensor faults. The method is exemplified and evaluated by applying it to real traffic data collected through loop detectors installed in a medium city road network. Employing the proposed methodology can increase the accuracy of sensor observations, as well as ease the learning of different traffic patterns, and improve the performance of traffic simulations.

The rest of this document is structured as follows. Related works are introduced in Section 2. While Section 3 outlines the context of the use cases. The methodology used, introduced in Section 4, is composed of: filtering and detection of anomalies (Section 5), classification of anomalies in sensor faults and unusual traffic conditions (Section 6) and the creation of a traffic model for running simulations (Section 7) that take advantage of the anomalies identified and classified in the previous steps. The experimentation on a real scenario is reported and discussed in Section 8. Section 9 sketches conclusion and future directions.

This work extends the conference paper [5] that introduced a novel data cleaning process to detect anomalies in real-time traffic data streams, exploiting a traffic sensor network. In this paper, the data cleaning process, that previously contained only anomaly detection, is ameliorated with an anomaly classification phase where anomalies are classified into sensor faults and unusual traffic conditions; then, only observations classified as sensor faults are removed from the input of the traffic model. Several changes and extensions have been made to the article w.r.t. the previous conference paper. In particular, Section 3 is enriched with two new subsections (3.3 and 3.4) to help the reader to better understand the features of traffic data on which the anomaly detection and classification method is applied; in Section 4, the proposed methodology is ameliorated with an anomaly classification phase where anomalies detected by an anomaly detection algorithm based on STL are classified. In subsection 5.3, a new version of the anomaly detection process, that applies RobustSTL and the inverse function of the logarithm to the residual values and finally normalizes the obtained values using the Robust Scaler, is provided and compared to the original one presented in [5]. A new Section 6 is inserted to describe the classification of anomalies into sensor faults and unusual traffic conditions. In the end, Section 8 has been extensively enriched and restructured into several subsections; new experiments have been conducted to test the anomaly classification (subsection 8.3), and new traffic simulations have been run to demonstrate that eliminating anomalies after having classified them and distinguished among sensor faults and unusual traffic conditions further improves the results of the traffic model (subsection 8.4).

2 Related Work

With IoT devices pervading our everyday life, we see an exponential increase in the availability of data streams. As known, these devices may have abnormal behavior; thus, different techniques to perform data cleaning on sensors' data have been discussed, classified, and compared.

In the literature, we find supervised [12], unsupervised [20], and semi-supervised algorithms. However, several anomaly detection methods are formerly created for processing data in batches, and unsuitable for real-time streaming applications.

In [25], constraint-based algorithms are used with a focus on speed constraints. Then, time series anomaly detection algorithms are proposed: ARIMA (Autoregressive Integrated Moving Average), LSTM (Long Short-Term Memory), DBSCAN (Density-Based Spatial Clustering), and GANs (Generative Adversarial Networks). time series anomaly detection can be performed with several techniques: statistical approaches, machine learning algorithms, and deep neural networks. In [8], 20 different univariate anomaly detection methods from the above-mentioned three categories are compared and evaluated on publicly available datasets. Also, Seasonal-Trend decomposition using Loess (STL) is exploited for anomaly detection combined with other methods, such as Interquartile Range (IQR). In [15], a two-layer outlier detection approach is proposed: firstly, the non-stationarity and periodic variation of the time series are studied, then observable variables in the environment are used to explain any additional signal variation. The authors of [14] combine STL decomposition with the SARIMA model to detect anomalies in non-periodic time series. In the end, the approach described in [23] combines STL decomposition with extended isolation forest. Our method integrates the use of STL decomposition with a constraint-based filter on the data coming from the sensors. This approach is missing in the aforementioned methods. The filter identifies anomalous values that would not be recognized as anomalies by the STL decomposition technique.

Explaining the causes of anomalous sensor readings is a tricky issue. An interesting approach that tries to classify sensor faults is the one proposed in ClariSense+ [11]. ClariSense+ is an automated anomaly clarification service with the ability to explain sensor anomalies using social network feeds from Twitter. Authors have demonstrated the feasibility of explaining the causes of traffic anomalies by identifying unusual social network feeds that are correlated with each anomaly in time and space. In [30], an accurate and transferable accident detection approach is described. The detection methodology is based on the relationship between traffic variables and observed traffic accidents. Authors employ a deep learning-based method calibrated using part of the collected traffic variables and the pre-assigned traffic accidents. This methodology is not able to detect sensor faults but is focused on the discovery of traffic accidents.

Correlated anomaly detection is a cutting-edge topic. Especially from streaming data, it is an essential task in several real-time data mining applications. In [9], a framework is introduced for better detection of correlated anomalies from large streaming data of various correlation strengths. The experiment shows balanced recall and estimated accuracy of the framework that was applied to the U.S. stock daily price data set. In the traffic context, the correlation among traffic sensors to detect anomalies has been analyzed in [24]. The authors exploited a statistical baseline method, along with a sensor correlation analysis. They evaluated the approach by comparing the detected anomalies against traffic alerts, which are emitted by Traffic Agents on Twitter.

Another possible approach is described in [31]; in this study, faulty readings from traffic sensors are identified by examining the correlations among them and by taking advantage of the ubiquitous citizens through crowd-sourced data. The authors evaluate cross-correlation between sensors using, firstly, the Pearson metric, and then employing a multivariate ARIMA model to detect anomalies considering correlated sensors. In our work, we use a different correlation metric and employ correlation to perform anomaly classification once the anomalies have already been detected with our combined anomaly detection methodology.

3 Context

This section is devoted to describing the urban context and the traffic sensor network in the city of Modena. Moreover, a statistical overview of the traffic data is provided, including an analysis of the stationarity of the traffic sensors' time series.

The discussed use case consists of the traffic sensor network of Modena, a medium Italian city of $184,727$ inhabitants with a population density of $1,017$ $inhabitants/km^2$ and more than 900 km of public roads. Data coming from around 400 traffic sensors are given as input to a customized traffic model that simulates traffic flows on the whole road network in real-time. The improvement provided by the data cleaning process has been evaluated through a comparison between the model output, considering or excluding sensor faults.

3.1 Sensor network

Smart Cities employ sensors to share information with the public, businesses, city managers, and other smart systems. For traffic management, there are plenty of sensors, that exploit different technologies, able to determine the number of vehicles traversing the city streets (e.g. induction loop, preformed loop, pressure sensor, radar, video). Induction loops are the most commonly used sensors, introduced 50 years ago, that are still present in a lot of cities. An induction loop sensor consists of wire "coiled" to form a loop that is installed into or under the surface of the roadway. In Modena, induction loop detectors are spread in different locations, usually near traffic lights. These sensors collect traffic data (i.e. the number of vehicles and the average speed) with different frequencies according to the provider of the data. In Modena, we have two data providers for traffic information: the sensors located in the urban area are managed by the city council and send data every 1 minute; while the sensors in provincial and regional roads are owned by the Region and their data are distributed by a regional company with a frequency of 15 minutes.

Sensors data are collected and exploited to emulate real routes of vehicles in a traffic model [2,17,18]. Modena sensor map[1] displays all the traffic sensors available in the city of Modena, the ones in green are used as input to the traffic model. The others, excluded from the input of the traffic model, might be unreliable sensors (i.e. they obtain only a few measurements in a day interval, or most of their measurements are zero values) or sensors located in streets that are outside the urban area where we run the traffic model.

3.2 Sensor data collection

The sensors' data coming from the two data providers are collected in real-time into a PostgreSQL database

[1] Modena Sensor Map: `https://trafair.eu/modenasensormap/`

[17]. The database exploits two extensions: PostGIS to handle geospatial data and Timescale to perform better with a huge amount of data. From September 2018 till now (June 2021) the database collected more than 457 million observations recorded by urban traffic sensors in Modena. For each observation, the database stores a record composed of the identifier of the sensor, the flow measured by the sensor, the speed, the type of the vehicles (only for provincial and regional sensors), and the time slot of the measurement. The process to parse data coming from the traffic sensors and store them has been implemented in Python.

3.3 Statistical overview of traffic sensors observations

In order to detect anomalies in data streams, a statistical analysis of the data distribution can help to determine the best methodology for the use case. Considering all the measurements from October 2018 till May 2020, we discover that 33 out of 400 sensors did not provide any measurement. Then, the observations provided by the remaining sensors were aggregated every 15 minutes and for each sensor median, mean, standard deviation, and interquartile range (IQR) were evaluated. As a result of this evaluation, 35 sensors were considering malfunctioning since their mean, median, and IQR of traffic flow were zeros. Observing the value of IQR of traffic flow, some sensors have an IQR equal to zero. This means that there is no difference between first and third percentile, and all the measurements have a very similar value. This is not the regular behavior of a traffic sensor, where measurements are supposed to vary during the day. Thus, the 6 sensors with IQR equal to zero were excluded. Moreover, for 8 sensors the rate of measurements with a flow value equal to zero was above the 90%; thus, they cannot be considered reliable. To summarize, the total number of untrustworthy traffic sensors is 49 in addition to 33 not working sensors, as reported in Table 1. These sensors were not included in the data cleaning procedure and their data were not given as input to the traffic model. The number of reliable sensors is 318.

Traffic sensors installed in Modena	400
Not working sensors (no measurements)	33
Sensors with zero mean, median, and IQR	35
Sensors with zero IQR	6
Untrustworthy sensors (90% rate of zero values)	8
Reliable sensors	318

Table 1: Analysis of the reliability of traffic sensors.

The Table in Figure 1 shows some statistical evaluation for the distribution of IQR, median, and standard deviation considering only the 318 reliable traffic sensors. The median of the traffic flow values has an average value of 30 vehicles in 15 minutes; however, there are sensors with a median equal to zero and sensors with a median equal to 132. Observing the graph showing the distribution of the median, the majority of sensors have a traffic flow median value between 0 and 25. This happens because the time series of measurements of traffic flow also includes the night period where the traffic flow values are all near zero and the standard deviation and the IQR are significantly reduced. Besides, the graphs show that the distribution of median, IQR, and standard deviation values is right-skewed: mean is higher than mode. Thus, there are few sensors with high values and more sensors with lower values.

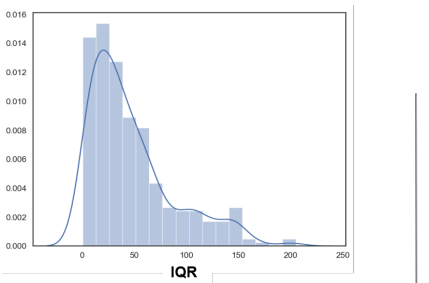
3.4 Stationary study

If we think about vehicle traffic in a city, many trends come to mind: a daily trend, a weekly trend, a similar trend in working days that is different in the weekend, and also seasonal trends. Here, we want to introduce the concept of stationary and study the stationary of the time series provided by the traffic sensors in Modena.

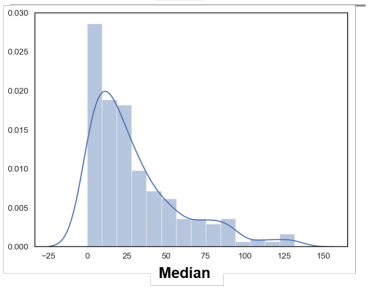
In a stationary time series, each time series measurement reflects a system in a steady state. A series X_t is called "stationary" if, loosely speaking, its statistical properties (mean, variance and covariance) do not change over time [19]. A stationary time series does not exhibit trends or patterns based on time, or periodic fluctuations (seasonality). There are three types of stationary: (1) a *strict stationary* series satisfies the definition as mentioned above of stationary; (2) a *trend stationary* series exhibits a trend, that, if removed, makes the resulting series strict stationary; in the end, (3) a *difference stationary* series can be transformed into a strict stationary series by differencing. Before applying any prediction model to a non-stationary time series, the time series has to be converted into a strict stationary series. For this reason, we tried to study our time series stationary. The easiest way to do this is by differencing, which means to compute the difference of consecutive terms in the series, following the formula $Y_i = X_i - X_{i-1}$, where i is the time instant and X_i is the value of the time series at instant t.

The most popular tests to check if a time series is stationary or non-stationary are:

- Augmented Dickey Fuller test (ADF);
- Kwiatkowski Phillips Schmidt Shin test (KPSS).



	IQR	Median	standard deviation
mean	47,37	30,27	29,82
standard deviation	40,67	29,12	24,68
min	1,00	0,00	0,97
first quartile	17,00	9,00	12,63
median	35,00	21,00	23,01
third quartile	63,00	42,00	39,11
max	205,00	132,00	155,39



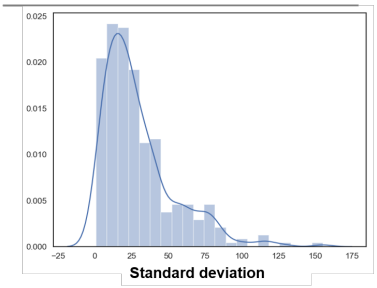


Fig. 1: Statistics of IQR, median and standard deviation considering 318 trustworthy traffic sensors.

ADF test is a statistical test and, in particular, a unit root test that aims at determining how strongly a time series is defined by a trend. A unit root is a characteristic element of a time series that makes it non-stationary; Besides, the number of unit roots in a time series corresponds to the number of differencing operations required to make the series stationary. Since the ADF test is a statistical test, there are two hypotheses to be tested: a null hypothesis (H0) and an alternate hypothesis (H1). The null hypothesis is that the time series is not stationary and has some time-dependent structure that can be represented by a unit root. The alternate hypothesis instead is that the time series is stationary. The p-value obtained by the ADF test is exploited to interpret the result of the test, i.e., whether to reject the null hypothesis or not. In statistical, the p-value is a probability score that establishes the significance of an observed effect. In other words, it is the probability of seeing the effect E when the null hypothesis is true ($p-value = P(E|H0)$). If the p-value is lower than a threshold, the time series is stationary (H0 rejected); otherwise, it is non-stationary (H0 not rejected). Typically, the threshold is set to 0.05.

Also, the KPSS test is a unit root test to study the stationary of the time series around a deterministic trend. A time series exhibits a deterministic trend if the slope of the trend does not change permanently; even if the series goes through a shock, it tends to regain its original path. The difference from the ADF test is that the null hypothesis of the KPSS test is the stationary of the series. Therefore, the p-value must be interpreted

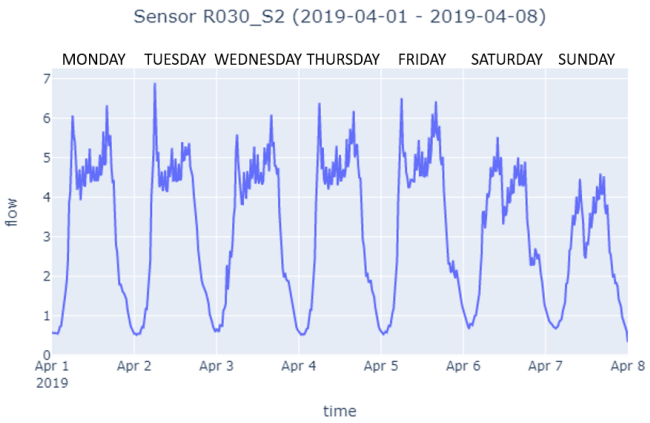


Fig. 2: Traffic flow of sensor "R030_S2" in the first week of April 2019.

oppositely: if the p-value is lower than the threshold, the series is non-stationary.

We decided to apply both the tests to be sure of the stationary of the time series. In some cases, the results of the two tests could conflict with each other. In particular, if the result of KPSS test is "stationary" and the one of ADF test is "non-stationary", then the series will be trend stationary; on the other hand, if the result of KPSS test is "non-stationary" and the one of ADF test is "stationary", the series will be difference stationary.

Our goal is to study the stationary of the time series of traffic data in Modena to find the best way to look for anomalies. By way of example, Figure 2 shows the number of vehicles measured by one traffic sensor; data

is related to the period from 1st to 8th April 2019 and is grouped over 15 minutes. Looking at the curve, it is straightforward to identify the weekly and daily trends. To check the stationary of these time series, we applied both the ADF and the KPSS tests to the 15-minutes aggregated measurements of April 2019; we exploited the statsmodels package in Python. Then, we compared the p-values of each daily time series to the commonly used significance threshold of 0.05. All the p-values are far less than the threshold. Therefore the time series are classified as stationary, as expected.

4 Proposed methodology

Our goal is to perform data cleaning on traffic observations by identifying anomalies among the data coming from the traffic sensors to exclude them from the input of the traffic simulation model. As represented in Figure 3, the methodology consists of several steps: (1) filtering observations with an anomalous flow - speed correlation (these observations are stored as "filtered" in the database) (described in more details in Section 5.1), (2) anomaly repairing by replacing "filtered" observations with average values, (3) anomaly detection through STL decomposition following two different approaches, (4) classification of anomalies in sensor faults and unusual traffic conditions, (5) generation of the input for the traffic simulation model by removing sensor faults, and (6) running the model.

The effects derived from the data cleaning will be investigated, comparing the performance of the traffic model excluding anomalies from the input data or including them.

5 Filtering and anomaly detection

In this Section, we describe the first three steps of the proposed methodology, namely filtering, anomaly repairing, and anomaly detection of Figure 3.

5.1 Flow - speed correlation filter

The values of flow and speed provided by the traffic sensors are strongly correlated with each other. Assuming that a sensor is located in a single lane, as in our case, there is a maximum number of vehicles that can pass over a sensor with a certain average speed in a fixed time interval. This number is provided by the following formula:

$$num\_vehicles = \frac{speed[Km/h] * 1000}{vehicle\_length + safe\_distance[m]}$$

where *speed* is the average speed provided by the sensor, *vehicle_length* is the average length of different types of vehicles, and *safe_distance* is the safe driving distance that should be kept between every couple of vehicles based on the vehicle speed (calculated as *speed*/3.6). We set the value of *vehicle_length* to 4.

The value of *num_vehicles* represents the maximum number of vehicles in one hour based on the average speed of the vehicles. The upper bound for allowed values of flow is obtained by dividing the value of *num_vehicles* by 60 or by 4 based on the time interval which the observation is related to (1 minute or 15 minutes, respectively). If the flow provided by the sensor is higher than this number, the observation is considered an anomaly and marked as "filtered", therefore it will not be considered in the following steps.

We apply this filter in real-time, that is, every time an observation is stored in our database we add a mark "filtered" or "non-filtered" on it.

5.2 Anomaly repairing

The majority of sensors provide measurements every minute, and the filtering is applied to this time interval. Then, data need to be aggregated every 15 minutes to be used by the traffic model. For this reason, once sensor measurements have been labeled as filtered, they cannot be simply removed from the traffic sensor observations. To evaluate the aggregated flow in a bigger time interval, flow values are summed up, thus removing the observation is like considering that zero vehicles are passing in that time interval. This assumption is not correct, thus an alternative solution needs to be found. Since the measured value in the filtered observation is not reliable, it is substituted considering the values observed in the proximal time interval by the same sensor. We decide to consider a time interval of 15 minutes. The flow of filtered observation is thus replaced with the average of the reliable (non-filtered) flows measured by the sensor in the same time interval.

The aggregated speed is more difficult to evaluate; since the measure provided by the sensor is an average speed, we assume that filtered observations have a speed equal to the weighted averaged speed evaluated considering only the non-filtered measurements in the 15 minutes time interval. This anomaly repairing technique cannot be applied when less than 2 reliable observations are available for a sensor in the 15 minutes time interval. In this case, the aggregated flow and speed of that 15 minutes time interval are classified as anomalies, and they are not used in the following steps.

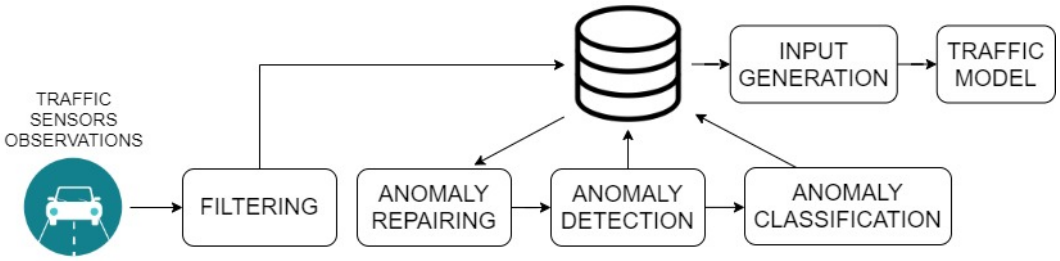


Fig. 3: Workflow of the proposed methodology.

5.3 Anomaly detection through STL decomposition

The Seasonal-Trend Decomposition using Loess (STL) is a filtering procedure for decomposing a time series into three components: the trend, the seasonal, and the remainder (also called residual) [10]. Splitting the time series into components can help in identifying anomalies.

The additive decomposition considers the time series model described by the following formula:

$$y_t = \tau_t + s_t + r_t, \qquad t = 1, 2, ..., N$$

where y_t represents the observation at time t, τ_t is the trend in time series, s_t is the seasonal signal with period T, and r_t is the remainder component.

The trend component consists of the low-frequency variations in the data with non-stationary, long-term changes, i. e. continuous increase or decrease. The seasonality instead is composed of the variations (periodic patterns) in the data near a baseline. Typically, trend changes faster than seasonal. The remaining variations are included in the residual.

The decomposition of the time series is obtained by applying the locally-weighted regression (Loess smoother) several times, iteratively. The result of these applications is a curve representing a smoothing of the original time series, computed taking into account the value of a variable number of observations in the neighborhood.

A variant of STL is the RobustSTL [26], defined as a robust and generic seasonal-trend decomposition method able to extract seasonality from data with a long seasonality period and high noises. The method applied by the RobustSTL consists of four steps:

1. noise removal by applying bilateral filtering and using neighbors in a window of $2H + 1$ observations with similar values to smooth the time series;
2. trend extraction by using the least absolute deviations (LAD) loss with sparse regularization;
3. seasonality extraction through non-local seasonal filtering which takes into consideration K seasonal neighborhoods of $2H + 1$ observations with different weights according to their distance in the time dimension and their seasonality values;
4. final adjustment by calculating the mean of seasonality, which is added to trend and removed from seasonality.

Several configuration parameters must be provided to the algorithm: the period T, that is the number of observations in each cycle of seasonality, the hyper-parameters of the bilateral filter of step 1 ($dn1$ and $dn2$), the number of the neighborhood (H), the regularization parameters for trend extraction ($reg1$ and $reg2$), the number of past season samples (K), and the hyper-parameters of the bilateral filter in seasonality extraction step ($ds1$ and $ds2$).

After each step, the input signal is updated by removing the component extracted in that specific step. After step 4, the steps are repeated until you get convergence between the remainder of the current iteration and the one of the previous iteration. The convergence is computed by the following formula:

$$convergence = \sqrt{(r_i - r_{i-1})^2}, \qquad i = 1, 2, ..., N$$

where r_i is the remainder at iteration i (current iteration) and r_{i-1} is the remainder at iteration $i - 1$ (previous iteration). If the convergence is higher than a threshold, the process will continue with another iteration, otherwise, the decomposition obtained is considered definitive. In the latter case, the results are 3 time series representing trend, seasonal, and residual.

Once the decomposition is concluded, the curve of residual can be analyzed to detect anomalies by using different methods. A solution could be the application of the interquartile range (IQR), which allows the calculation of the two fences to define reliable values. More details are provided in the following.

We evaluated two versions of the anomaly detection process: ADP1 and ADP2. The operations of each version are explained in Figure 4. In ADP1, we apply the logarithm to the flow values obtained after anomaly

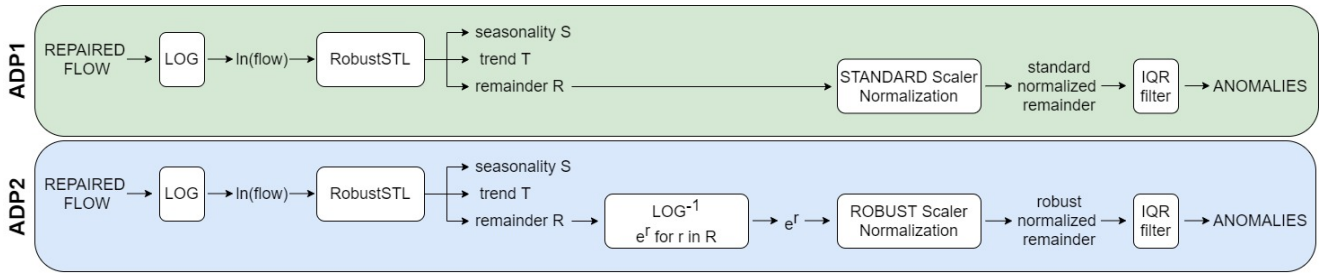


Fig. 4: The two versions of anomaly detection process: ADP1 and ADP2.

repairing and provide it to RobustSTL. Then, remainder values are normalized by using mean and standard deviation, as follows:

$$r\_normalized_t = \frac{r_t - mean}{standard\ deviation}$$

where $mean$ and $standarddeviation$ are computed on the remainder. Thus, the first ($Q1$) and third ($Q3$) quartiles are calculated on the normalized residual to find lower and upper fences with the following formulas:

$$lowerfence = Q1 - k * IQR$$

$$upperfence = Q3 + k * IQR$$

where IQR is the difference between third and first quartiles, and k is a multiplier. The observations with a residual value lower than $lowerfence$ or higher than $upperfence$ are outliers. The higher the multiplier value, the fewer outliers are found. The value of k depends on which type of outliers we want to detect; high values of k are used for extreme outliers.

In ADP2, after the application of RobustSTL, the inverse function of the logarithm is applied to the residual values. The obtained values are normalized using the Robust Scaler, as follows:

$$r\_normalized_t = \frac{r_t - median}{IQR}$$

where $median$ and IQR are evaluated on the remainder after the inverse logarithm function. Then, the lower fence and upper fence are calculated on the normalized remainder values and the IQR filter is applied to them. Again, the observations with residual value out of the range between the lower fence and the upper fence are outliers.

We set up n processes to apply STL decomposition to sensors data in real-time, where n is the number of sensors in our dataset. Indeed the decomposition of the time series of one sensor is independent of the others. We exploited an implementation of RobustSTL available online.[2]

[2] https://github.com/LeeDoYup/RobustSTL

In both anomaly detection process versions, the decomposition is performed every 15 minutes. The time series, used as input, is generated grouping by 15 minutes the logarithm of one-week flow measurements of a specific sensor. After the application of IQR, only the anomalies of the last 15 minutes are taken into account since the anomalies on the previous time slots have already been extracted by the previous applications of the decomposition. We choose to aggregate observations every 15 minutes since the traffic model takes in input this type of aggregation, as described in Section 7. The choice of using one week observations is a trade-off between the need to provide context for decomposition and the requirement of having results in a short time. Indeed, the time required for one month decomposition (on average 12 minutes) is far greater than the one for week decomposition (on average 15 seconds). Therefore, we set the period T to the number of observations in the week divided by 7. After several experiments, we decided to set the values of the other configuration parameters in this way: $dn1$=1, $dn2$=1, H=3, $reg1$=10, $reg2$=0.5, K=1, $ds1$=50, $ds2$=1. In the end, we initialized the value of k in the IQR method to 3 since we want to avoid that real unusual traffic conditions are labeled as anomalies. Using a lower value for k in the IQR method to 3 since we want to exclude those observations that are real anomalies. Using a lower value for k, we noticed that a lot of observations were considered anomalies and, checking the values of flow and speed, they did not seem like real anomalies. Probably this is due to the fact that our sensors are located near the traffic lights, therefore, the presence of peaks is possible and could be related to the turned green of the traffic light.

6 Anomaly Classification

In this section, we introduce the methodology we applied to classify anomalies and distinguish between sensor faults and unusual traffic conditions.

Fig. 5: Examples of correlated sensors: in a junction (on the left), and on a road (on the right).

Firstly, the correlation among traffic sensors was studied: in this phase, groups of sensors that have similar trends are identified. Anomalies are classified as unusual traffic conditions when they occurred contemporary in correlated sensors, otherwise, they are labeled as sensor faults. In the road traffic environment, an anomaly can be caused by traffic accidents, road closures, and road works that create an unusual traffic condition, or by faulty sensors, i.e. sensors that are not able to measure correctly, vehicles in the streets. In the case of unusual traffic condition, the measurement collected by the sensor is a real value; on the other hand, in the case of a sensor fault, the measured value is an error and do not correspond to the real number of vehicles crossing the street in the sensor's position. In [13], a description of how these faults may occur is given.

Induction loop sensors are composed of inductive and capacitive elements that may encounter open or short circuit faults during measurements. Such faults lead to erroneous interpretation of data acquired from the loops. The idea behind the classification of anomalies is the following: if an anomaly occurs only in the time series of a sensor and, at the same time, it is not recorded by other sensors correlated to the sensor itself, then it is a sensor fault. Given a sensor A, B is a correlated sensor of A if its observations of traffic flow are in relation with the observations recorded by A. Usually, B can be placed in proximity of A. For example, sensors placed in the same junctions are often correlated or sensors on the same road and lane are correlated sensors (as shown in Figure 5). Otherwise, if the anomaly recorded by the sensor occurs in the same period as the anomalies recorded by the correlated sensors, then it is an unusual traffic condition.

In order to distinguish between sensor faults and unusual traffic conditions, the correlation evaluated with DCCA (described in Section 6.1) was taken into account. Each anomaly was associated with anomalies that happened in an adjacent time interval. The amplitude of this time interval is a parameter that should be defined considering the frequency of observations. For each anomaly, the number of traffic sensors that had simultaneous anomalies (ns) was calculated, then the number of correlated sensors was counted (nc). If an anomaly happened in adjacent time intervals for correlated sensors, the anomaly is considered an unusual traffic condition.

The anomalies observed for sensors with a low number of correlated sensors are more likely to be classified as sensor faults, even if ns is high. For this reason, we take into account not only correlated sensors but also proximal sensors. For each anomaly classified as sensor fault, the distance between the sensor it belongs to and each traffic sensor showing a simultaneous anomaly was evaluated; if there are at least two other sensors experiencing an anomaly in a radius of 1500 meters from the sensor, the anomaly is classified as an unusual traffic condition. The distance was evaluated directly, querying the PostgreSQL database where sensor locations are collected. This allows us to identify additional spatial correlations between traffic sensors located nearby that may appear in specific traffic conditions (i.e., traffic accidents that caused re-routing of vehicles) but were not detected by the DCCA coefficient that mainly explores the temporal dimension. An overview of the classification process is shown in Figure 6.

6.1 Correlation

If sensors are placed nearby one another, an unusual traffic condition will be recorded by the majority of sensors in the area. Instead, if the anomaly is caused by sensor fault, the majority of sensors located nearby will not observe any anomaly. The relation between sensors is not completely dependent on their spatial distance. The road network has a complex structure, and two sensors can be distant but placed on two connected roads. For this reason, a way to deeply understand the relation between two or more sensors is to take into account historical data and evaluate their correlation with a representative metric.

Considering measurements referring to one month (April 2019), the correlation between sensors was investigated. Sensors' observations are collected every minute at different timestamps, and to compare them, we need a common timestamp. Thus, we aggregate values every 15 minutes. Moreover, the traffic sensors available in Modena are located near traffic lights, and their measurements are affected by the traffic light logic and the presence of the red light. When the traffic light is red, vehicles cannot move; thus the number of vehicles that can be counted is one (if there is a vehicle that stops on the sensor) or zero. Aggregating data every 15 minutes

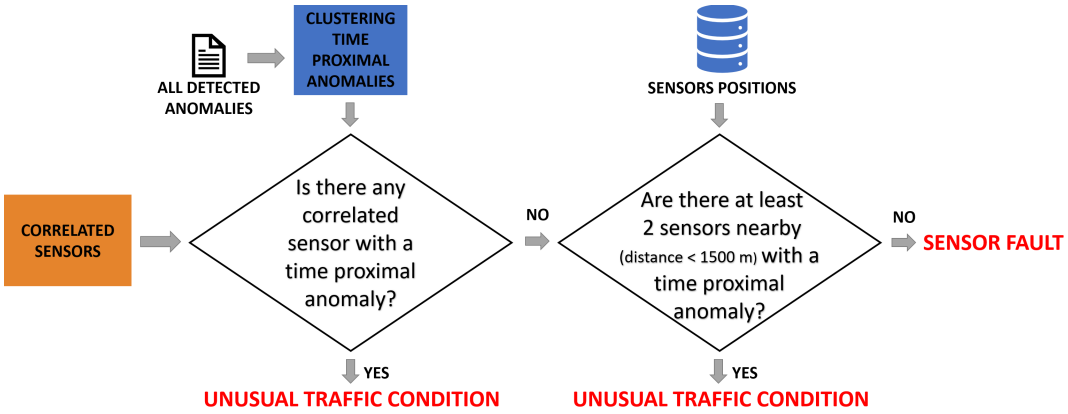


Fig. 6: Classification process.

reduces this effect and allows us to better understand the real traffic flow.

Firstly, the correlation between sensors was studied on the 15-minutes aggregated time series with Pearson coefficient, as described in Section 6.2. Then, since Pearson correlation is not suitable for skewed distributions, in order to ameliorate the correlation study and obtain a more reliable result, DCCA was studied (Section 6.3).

6.2 Pearson Correlation

Firstly, we evaluate the correlation between sensors using Pearson correlation metrics. The Pearson coefficient measures the strength and direction of linear relationships between pairs of continuous variables. As described in [6], the Pearson coefficient is easily computed, dividing the correlation between the two variables for the product of the square of their variance. The correlation was firstly evaluated considering whole traffic observations of April 2019. Only traffic sensors with a correlation coefficient higher or equal to 0.8 and with a distance between them lower than 2000 m are considered correlated; these thresholds were selected considering the necessity of a strong correlation and the mutual distance of the sensors in our urban area. There were a lot of high correlation scores. Observing Figure 7 showing the correlation between two highly correlated sensors, the comparison between correlation studied considering night period or not shows that the high number of zero values registered during the night strongly influences the distribution of each sensor. For this reason, the correlation was evaluated by removing night hours (between midnight and five in the morning). For the two sensors in Figure 7, the exclusion of night hours strongly decreases the observed correlation to 0.74, which is lower than 0.8. Thus the two sensors will no more be considered correlated. The sensors correlated with sensor "R013_SM21" are represented in Figure 8. For each of the traffic sensors that collect at least one measure every 15 minutes in the reference period (they were 310 out of the 318 reliable sensors), the correlated sensors have been evaluated with the presented methodology. For each sensor, the average number of correlated sensors is 21. In Table 2, can be observed that sensor showing at least three correlated sensors are 242, 196 of them considering a threshold for the Person coefficient of 0.8 and 46 relaxing this threshold to 0.7. The sensors that have less than three correlated sensors, but at least one correlated sensor, are 30. Finally, the isolated sensors are 38.

Pearson correlation assumes that the two variables are individually normally distributed [22], thus it is not suitable for skewed distributions. Since we aim to find outliers, we can assume outliers are present in our data. The Pearson correlation coefficient is also very sensitive to outliers.

6.3 Detrending Cross-Correlation Analysis Coefficient (DCCA)

In [28], a new coefficient is proposed to quantify the level of cross-correlation between non-stationary time series but can be employed to study traffic flow fluctuations in the presence of outliers also in stationary time series [27]. This cross-correlation metric is called DCCA (Detrending Cross-Correlation Coefficient). Since traffic sensors data have nonlinear behavior, their correlation can be studied by taking into account the properties of fractals. DFA (Detrended Fluctuation Analysis) is a method for determining the statistical autocorrelation of a trend-stationary signal. DCCA is a generalization of DFA which measures cross-correlation between two time series. The presence of outliers can in-

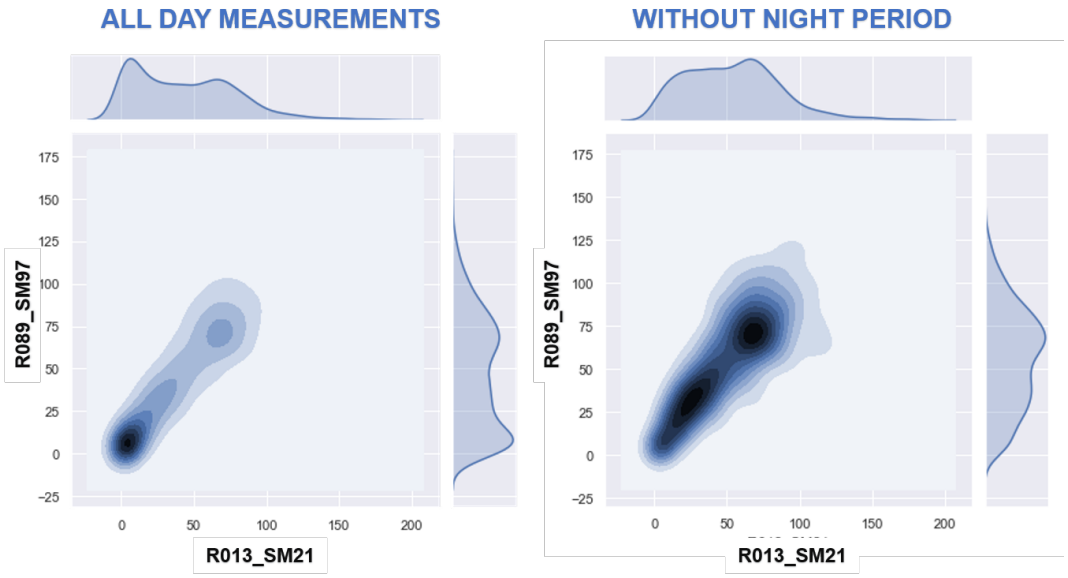


Fig. 7: Correlation between two traffic sensors, considering traffic sensor data of April 2019: the results are evaluated on the 24 hours of the day (on the left) and on daylight hours only (on the right).

	Pearson C.		DCCA	
Number of traffic sensors analyzed		310		310
Correlated sensors		242		266
Strongly correlated sensors (corr. thr. = 0.8)	196			
Weakly correlated sensors (corr. thr. = 0.7)	46			
Loosely correlated sensors (3>correlated sensors>1)		30		21
Isolated sensors		38		23
Strongly isolated sensors			17	

Table 2: Pearson correlation and DCCA correlations evaluated among the reliable traffic sensors in April 2019.

troduce noise in a stationary time series; thus, DCCA is preferred for the traffic flow correlation analysis. To evaluate the DCCA coefficient between two time series of equal length, four steps are necessary:

- integrating each time series;
- dividing the integrated sequence into non-overlapping segments and finding a polynomial function representing the trend of each segment for each time series;
- evaluating the local detrended covariance between the two time series in each segment;
- calculating the mean between the fluctuation evaluated for each segment to obtain a unique value.

In the integration step, for each time series, the total mean is subtracted from each value of the sequence, and then a cumulative sum is performed. This step is performed to obtain from each time series its profile. The two resulting profiles of length N (the same as the two original time series) are then divided into sequences of n values obtaining $N_n = N/n$ segments for each profile. The obtained segments are enumerated as s=1,..N_n. Then, each segment s is taken singularly to evaluate its trend and define a polynomial function (T) that approximates its sequence of values. The degree of the polynomial function is a parameter that can be fitted. Each sequence s starts at element $(s-1)*n+1$ of the profile and ends at $s*n$. The polynomial function has a value for each element of s and each profile. In the third step, the covariance of each segment is evaluated as follows [29]:

$$f(s,n)^2 = 1/n \sum_{i=1}^{n} \Delta_1 \times \Delta_2$$

$$\Delta_1 = {Y_{(s-1)*n+i}}^{(1)} - {T_{(s-1)*n+i}}^{(1)}$$

$$\Delta_2 = {Y_{(s-1)*n+i}}^{(2)} - {T_{(s-1)*n+i}}^{(2)}$$

where $T_{(s-1)*n+i}$ is the value of the polynomial function in position $(s-1)*n+1$ of the s segment, and $Y_{(s-1)*n+i}$ is the value in the sequence. Δ_1 and Δ_2 are the values of the detrended signal respectively in

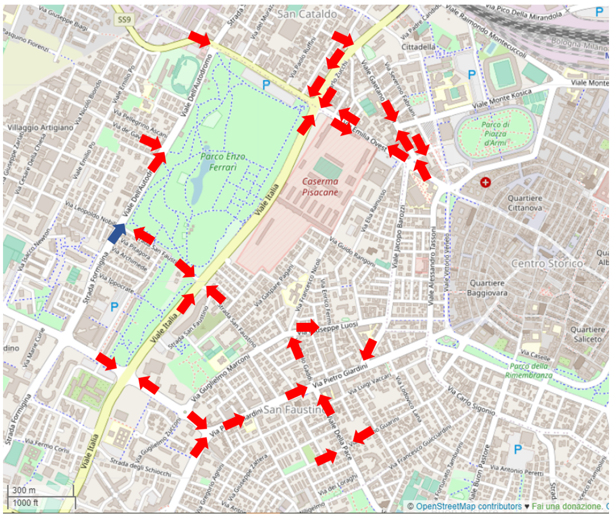


Fig. 8: Location of the sensors that are considered correlated to sensor "R013_SM21" (blue arrow) taking into account Pearson correlation. The red arrows indicate the position of the sensors and the traffic direction.

the first and second profile. In the last step, the average between all the evaluated fluctuations (one for each segment) is calculated to obtain a unique covariance that represents the correlation between the two time series. To quantify the level of cross-correlations, a dimensionless measure, the DCCA cross-correlation coefficient, needs to be evaluated. This coefficient is the ratio between the previously calculated covariance and the product of two detrended standard deviations. The DCCA method allows us to select a value of n. Thus, it is possible to evaluate the covariance considering segments of a given length. This is important for our use case, sensor measurements can be correlated daily but their correlation can be less evident considering a shorter period. We are mostly interested in the short-term correlation between sensors. The daily trend can be similar (peak hours in the morning, fewer vehicles during the evening), even weekly trend can be easily detected (fewer vehicles during the weekend and more during working days) but to identify sensor faults we need to find sensors that are correlated considering a time interval of maximum one hour. For this reason, the DCCA correlation coefficient was evaluated considering segments of length $n = 2$ (corresponding to 30 minutes). Two sensors are considered correlated if their DCCA correlation coefficient is higher or equal to 0.7, and their spatial distance is lower than 2000 m. The choice of these thresholds is the result of several empirical tests. The mean number of correlated sensors for each sensor is 42, a higher value than the one obtained with the Pearson correlation. The number of isolated sensors decreases in comparison with the Pearson correlation to 23. Comparing the isolated sensors detected by Pearson and DCCA, we found that 18 of them are in common. The DCCA methodology reduced the number of isolated sensors and better investigated the correlation between sensors. For the 23 isolated sensors, we decided to change the segment length $n = 4$ to cover one hour and to set the condition on the distance between correlated sensors at 2500 m. In this way, the isolated sensors became 17 (indicated as strongly isolated sensors in Table 2); 9 of them are located far away from any other sensors, one of them is a sensor that counts only busses and appears to have different behavior. For these sensors, we were not able to find correlated sensors.

A comparison of the results of correlation analysis conducted with Pearson Correlation and DCCA is shown in Table 2. It can be observed that DCCA methodology not only better fits our data but also allows us to discover more correlation patterns and reduce the number of isolated sensors. For these reasons, the DCCA correlation cohefficient is the one employed in our data cleaning process to classify anomalies.

7 Traffic Model

Traffic simulation models are mathematical tools that help to plan, manage and analyze urban mobility. Dynamic traffic models create a detailed evolution over time of traffic situations. The model employed is a dynamic microscopic model. In microscopic models, the movement of each vehicle is the result of individual choices depending on: interactions with other vehicles, road environments, and traffic signals. The vehicle's movements do not depend on macroscopic or probabilistic laws. Every single vehicle in the model is a unique entity with its own goals and behavioral characteristics; each possessing the ability to interact with other entities in the model. We employed the open-source micro-simulation model SUMO[3] [16], configured to generate the routes of the vehicles starting from traffic sensors data as described in [2]. This model produces data about vehicle counts and their average speed in every road of the city of Modena taking as input the measurements of the traffic sensors.

7.1 Input Generation

Vehicles routes are created taking into account real traffic sensors data aggregated into 15-minutes time slots.

[3] https://sumo.dlr.de

The model includes objects called calibrators. Calibrators are part of the SUMO suite and are like virtual traffic sensors calibrated considering the real measurements of the on-road sensors. Each calibrator can produce the aspired traffic flow, i.e. the number of vehicles counted by the sensor associated with that calibrator. For each real traffic sensor in the city, a correspondent calibrator is inserted in the simulation. Vehicle counts and speed from the traffic sensor are the values the calibrator aspires to reach. If no information about the traffic flow in a specific time interval is provided to the calibrator, vehicles are not added or subtracted. Instead, if any information about the vehicle flow to be reached is given, the calibrator acts adding or removing vehicles to reach the given traffic flow. For this reason, removing sensors faults from the input data is of fundamental importance to obtain a more realistic simulation.

We use OSM (OpenStreetMap) as the source of geographical data to define our road network and store it in our database. Every road section is a line object composed of several points. The road section is then divided into smaller segments that are the piece of road between two consecutive points.

7.2 Output

Simulations are performed on an HPC resource. Once a simulation is finished, its output is stored in the database. The values of flow (veh/h) and average speed (m/s) are collected for each lane segment (more than 17000 in our use case) every 15 minutes of simulation. An analysis of the traffic data obtained from several simulations is described in [3]. An additional output is provided for every sensor location: a time series of 15 minutes aggregated vehicle counts and speeds observed during the simulation at that point.

7.3 Traffic model evaluation

The main goal of a traffic model is to simulate a traffic flow close enough to real values observed in urban streets. The model provides as output the number of vehicles counted in the exact position where the traffic sensor is located. The vehicles number simulated in a point where a traffic sensor is located must be compared with real vehicle counts. Even if traffic sensor data are used to feed the model, calibrators are not always able to insert the required number of vehicles: the vehicles inserted in the simulation and the vehicles counted by real sensors can be different. This mainly happens when a sensor measures a very high flow and its calibrator inserts many vehicles causing a jam in the simulation and avoiding other calibrators, placed nearby, to eventually add additional vehicles. This very high flow could be caused by a sensor fault that affects the performance of the entire simulation.

The evaluation method of the presented traffic model is described in [4]. For each sensor, whose measurements are employed as input for the model, the distance between the two time series, i.e. real flows observed by the sensors and simulated flow, is calculated with three different metrics: the fast Dynamic Time Warping (DTW), the PointWise Distance (PWD), and the Count Time slots Distance (CTD).

The DTW is evaluated using FastDTW, a less complex version of DTW described in [21]. This metric allows sequences to be stretched along the time axis, is able to find corresponding regions in time series, and can tolerate noise, time shifts, and scaling in the y axis [7]. PWD is evaluated by summing all the differences between the measured flow and the simulated flow in all the time steps of the simulation. We do not use absolute distance, but we sum distances considering their sign since a subsequent time slot can compensate for the difference observed in a previous one. The CTD is the number of time slots in which the absolute difference between the measured and simulated flow is higher than 2 vehicles per minute.

Calibrators have been classified considering the presented metrics in “aligned” and “not aligned”. A calibrator is considered “not aligned” if the DTW distance between real measurements and simulated flow is higher than 1200, the PWD is higher than 30, and the CTD is higher than half of the total time steps of the simulation. A calibrator classified as “not aligned” cannot correctly insert the expected number of vehicles as required by the given input.

To evaluate the performances of the simulation, we need to produce metrics that can summarize with a unique value the distances observed in each sensor position. We defined 5 metrics that help us to compare simulations performed considering or excluding anomalies referring to the whole simulation: percentage of aligned calibrators, mean Root Mean Squared Error (RMSE), mean DTW reduction, mean PWD reduction, and mean CTD reduction. For each simulation, the number of calibrators classified as aligned has been divided by the total number of calibrators to obtain the percentage of aligned calibrators. A value of the percentage of aligned calibrators is calculated for the simulation performed including anomalies and the simulation of the same period excluding anomalies from the input of the model. A higher percentage means a higher number of aligned calibrators, thus an optimization of model performance. The RMSE between the real mea-

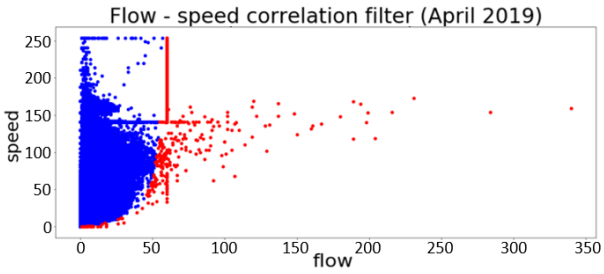
surements and the simulated flows is evaluated for each sensor position and averaged. For each simulation with and excluding anomalies, a value of mean RMSE is calculated. A higher mean RMSE means a bigger error and thus a decrease in the performance of the model. Mean DTW reduction, mean PWD reduction, and mean CTD reduction are obtained comparing the performances of two simulations: the simulations with all the available measurements (we will refer to this simulation as "Standard"), and the simulation excluding anomalies (we will refer to this simulation as "Cleaned"). The DTW distance between real measurements and simulated flow time series is evaluated in each sensor position for the two simulations as follows:

$$\frac{1}{N}\sum_{i=1}^{N}(DTW(m_{i,1}, s_{i,1}) - DTW(m_{i,2}, s_{i,2}))$$

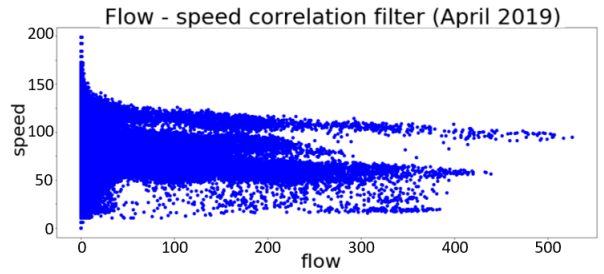
where $m_{i,1}$ is the real measurements time series of sensor i, $s_{i,1}$ is the simulated flow in the sensor position, and N is the total number of sensors. The difference between the DTW distances observed by the two simulations in each location is evaluated and the mean of these differences is called mean DTW reduction. If the value is positive, the distance has been reduced, thus the cleaned simulation performed better; otherwise, the standard simulation has better performances. Similarly, mean PWD reduction and mean CTD reduction are obtained by evaluating the difference between values of PWD distances or CTD distances of the two simulations referring to the same period for each sensor location and averaging these differences to obtain a unique value. If the value is positive, the distance was reduced and the cleaned simulation performed better than the standard one.

8 Experiment and result

To demonstrate the effectiveness of our methodology, the traffic model performances without a data cleaning procedure (standard simulation) and excluding anomalies (cleaned simulation) are compared for each day of April 2019. In Section 8.1, we show the results of flow-speed correlation filter. While in Section 8.2, the results of ADP1 without anomaly classification are discussed. Moreover, in Section 8.3, the results of the classification applied to both ADP1 and ADP2 will be compared and examined. Finally, in Section 8.4, the performances of the simulations obtained excluding anomalies detected by ADP2 and classified as sensor faults are presented.



(a) Traffic sensor observations in the urban area (1 minute time interval).



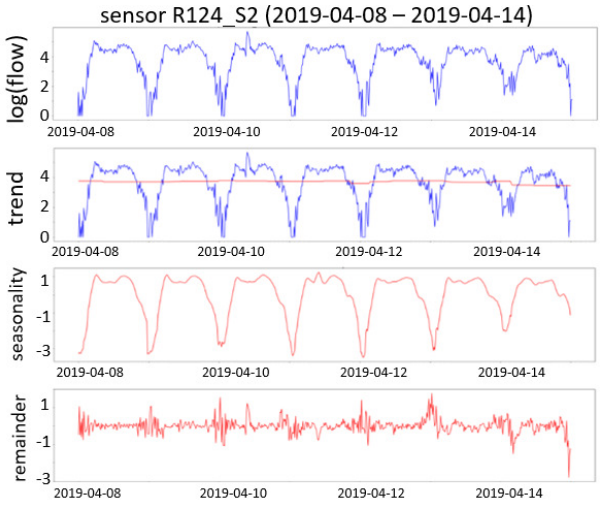
(b) Traffic sensor observations in provincial and regional roads (15 minutes time interval).

Fig. 9: Flow - speed scatter plots representing the observations of traffic sensors in April 2019 with filtered observations in red.

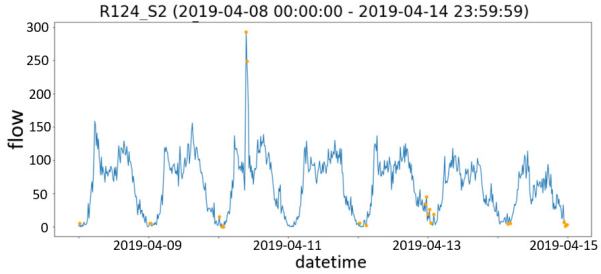
8.1 Flow-speed correlation filter

In April 2019, the number of observations coming from the traffic sensors was 13 millions, and they were produced by338 sensors.

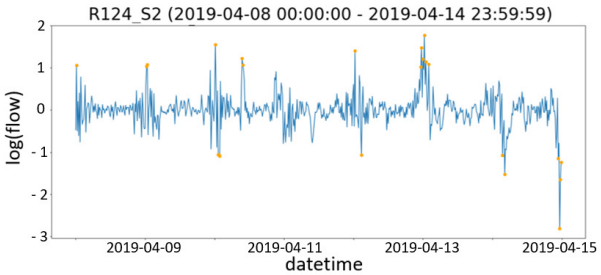
Using the flow-speed correlation filter, 450845 observations are filtered out (3% of the total number of observations). These filtered observations are related to 259 sensors. The scatter plots in Figure 9 show the values of flow and speed of the observations of urban sensors (Figure 9a) and sensors outside the urban area, in provincial or regional roads (Figure 9b) in April 2019; the red points are the filtered observations. As can be seen, no filtered observations are found among data coming from provincial and regional sensors. In Figure 9a, we can notice that very high values of speed are considered "non-anomalous" for low values of flow. Indeed, if there is no traffic (low flow), it could be possible that vehicles move at high speed, especially at night. However, for higher values of flow, an observation with a very high speed is considered "anomalous" and filtered. Then, the filtered flow values are replaced as described in Section 5.2.



(a) Decomposition of the time series.



(b) time series with anomalies (i.e. the orange dots).



(c) Remainder with anomalies (i.e. the orange dots).

Fig. 10: The STL decomposition of the time series related to the observations of the sensor "R124_S2" from April 8th, 2019 to April 14th, 2019 and anomalies detected by ADP1.

8.2 Improvement in traffic simulation with anomaly detection (ADP1)

In this subsection, we discuss the anomalies identified by ADP1 and the performance of traffic simulation obtained excluding these anomalies (ADP1 SIM).

After the anomaly repairing phase is applied to one minute filtered data, all sensor observations are aggregated every 15 minutes. The STL decomposition, applied to aggregated traffic data by using the IQR method, detects 13932 anomalies (less than 0.1% of the observations) related to 310 sensors. Figure 10a shows an example of decomposition which refers to the observations of one sensor from April 8th to April 14th, 2019. Time series decomposition involves thinking of a series as a combination of trend, seasonality, and remainder (also called noise or residual) components. Figure 10b draws anomalies, highlighted in orange, on the time series of all observations; while Figure 10c shows anomalies on the remainder component of the time series. This last Figure highlights that the anomalies are detected in positive and negative peaks on the remainder component of the time series.

Analyzing the time distribution of the anomalies, most of them are detected at night, as can be seen in Figure 11. Figure 12 shows the percentage of anomalies for every day of April 2019: in one day (April 6th, 2019) the percentage exceeds 2%.

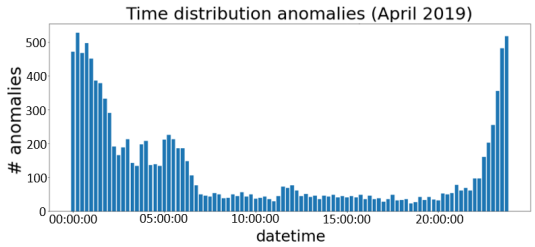


Fig. 11: Time distribution of anomalies in April 2019.

Once the anomalies have been detected by ADP1, they are stored in the database. Each anomaly is linked to the 15-minutes time interval of the aggregated observations and the sensor it belongs to.

For each day of April 2019, two simulations have been performed. The STD SIM uses as input all the available sensors observations (no data cleaning is performed on them). The cleaned simulation ADP1 SIM, instead, is obtained applying ADP1 to traffic data and removing anomalous observations from the input. We

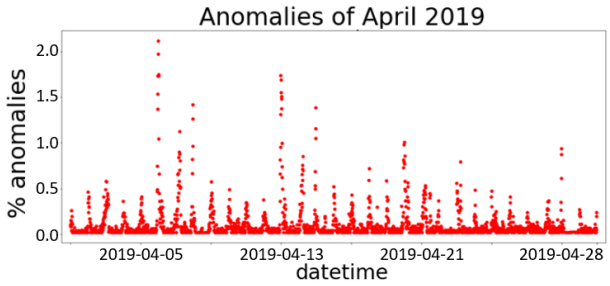


Fig. 12: Percentage of anomalies above the total number of observations for every day of April 2019.

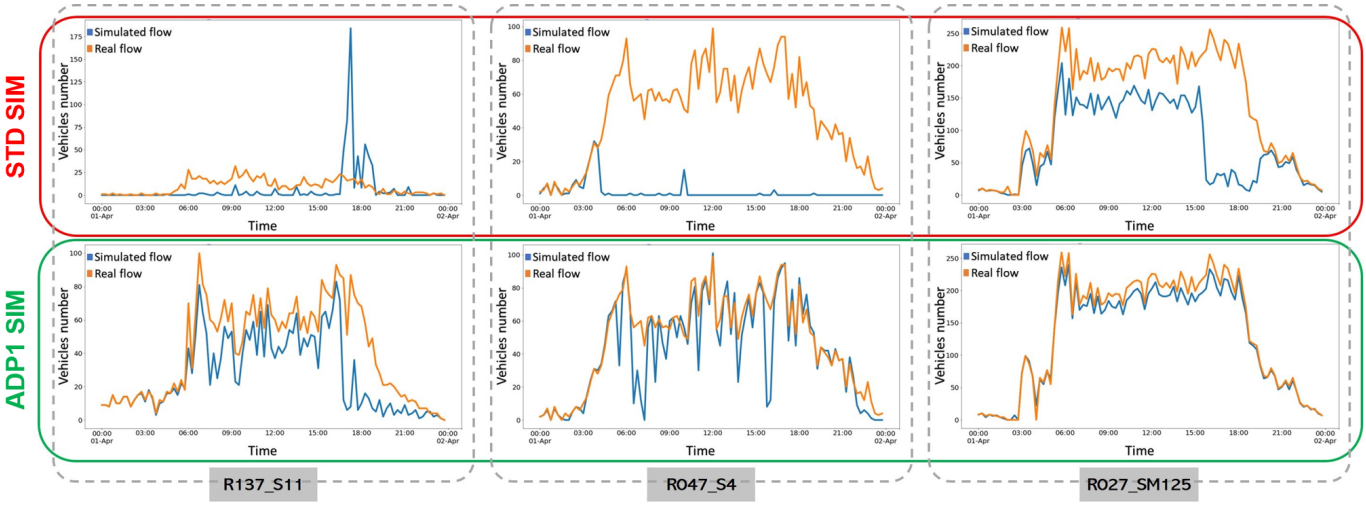


Fig. 13: Comparison between observed time series (orange) and simulated flows (blue) in three locations for the 1st of April. Standard traffic simulation (STD SIM - on the top) is compared to the traffic simulation that takes advantage of the data cleaning process with ADP1 (ADP1 SIM - on the bottom).

DAY	% Aligned STD SIM	% Aligned ADP1 SIM	Mean RMSE STD SIM	Mean RMSE ADP1 SIM	Mean DTW reduction	Mean PWD reduction	Mean CTD reduction	N. FILTERED	N. ANOMALIES
1	0.86	0.88	28.79	27.50	3.59	1.30	2	9821	404
2	0.85	0.85	28.96	27.28	88.03	2.41	2	9512	589
3	0.83	0.86	30.86	29.37	142.05	0.46	2	9979	603
4	0.84	0.82	29.19	30.15	-131.43	-0.98	4	9931	624
5	0.79	0.81	33.07	30.84	114.21	1.90	1	10188	1026
6	0.82	0.83	27.26	27.08	10.05	2.05	2	11185	681
7	0.91	0.94	19.41	19.12	-14.50	0.85	3	10914	784
8	0.83	0.85	27.83	30.38	6.26	-1.21	5	9678	116
9	0.83	0.81	28.35	28.56	-59.62	-0.39	2	9770	254
10	0.83	0.85	29.28	29.21	66.02	0.84	3	9908	309
11	0.81	0.84	30.04	29.76	73.66	-0.02	3	9758	212
12	0.81	0.79	29.73	32.06	-114.46	-1.19	5	10393	449
13	0.83	0.83	25.84	26.63	-16.12	0.13	3	11092	677
14	0.92	0.94	18.52	18.55	87.98	1.00	2	10664	723
15	0.82	0.84	29.21	30.05	-118.95	0.41	4	9514	114
16	0.83	0.83	30.95	30.15	33.93	1.15	3	9500	274
17	0.82	0.83	31.87	33.02	-67.18	-0.33	3	10001	225
18	0.84	0.84	30.33	28.40	-294.32	1.47	-1	9950	243
19	0.85	0.87	29.83	24.01	-29.10	-1.13	1	10364	336
20	0.84	0.89	23.76	19.87	50.86	0.22	1	10753	600
21	0.93	0.96	15.53	13.00	-42.23	-1.08	1	10262	543
22	0.93	0.96	17.69	13.88	37.32	0.37	0	9980	35
23	0.84	0.89	27.67	20.46	78.73	1.32	-2	10182	371
24	0.80	0.90	29.56	21.98	171.81	1.75	0	10099	176
25	0.91	0.94	18.11	14.35	7.75	0.21	0	10432	250
26	0.82	0.85	29.22	23.17	7.01	0.16	-1	10048	178
27	0.84	0.85	25.28	21.78	-135.00	-0.61	1	11026	139
28	0.86	0.90	29.69	27.63	19.99	-0.46	1	10587	360
29	0.81	0.85	31.49	25.35	28.37	-1.66	1	9694	42
30	0.82	0.84	32.57	25.85	-89.27	-0.50	1	9517	173

Table 3: Comparison of traffic model evaluation metrics between standard simulations (STD SIM) and simulations after removing anomalies of ADP1 (ADP1 SIM).

simply remove the anomalous observations from the input of the traffic model and calibrators simulate vehicles considering previous and next observations. Classification of anomalies is not applied in this case. The evaluation of both the standard simulation (STD SIM) and the ADP1 SIM are performed considering only the non-anomalous points. This because we assume that the real measurements labeled as anomalous are not reliable and cannot be used to estimate the error.

In Table 3, all the evaluated metrics are displayed for each day of April 2019. Comparison metrics described in Section 7.3 are evaluated, considering as the first simulation the standard simulation (STD SIM), and as the second simulation the simulation without anomalies detected by ADP1 (ADP1 SIM). In the 77% of daily simulations, the percentage of aligned calibrators increased excluding anomalies (as can be seen in the third column of Table 3), only on 3 days the number of aligned calibrators decreases. In the 73% of cases, the mean RMSE error decreases (as can be seen in the fifth column of Table 3). If mean DTW reduction has a positive value, the exclusion of anomalies reduces mean DTW distance. In only the 60% of daily simulations, the mean DTW reduction is positive (as can be seen in the sixth column of Table 3). Moreover, the mean PWD reduction is positive in the 60% of daily simulations (as can be seen in the seventh column of Table 3), this means that in the majority of days, the mean PWD was reduced through the data cleaning process. Finally, the mean CTD reduction is positive in the 90% of daily simulations (as can be seen in the eighth column of Table 3), thus the mean number of time slots in which the distance between the simulated flow and the

real measurements was higher than 2 ($veh/minute$) is significantly reduced. Overall, the 83% of days shows an improvement on at least 3 out of 5 metrics in the ADP1 SIM. For each day, the number of filtered one-minute data and the number of anomalies detected using STL is calculated. The days with the highest number of filtered values and anomalies are the ones with better performances (see for example 5th, 6th, and 20th April in Table 3).

The reason for this enhancement of performances was further investigated, observing the time series of measurements and simulation' flows in the more affected locations. In Figure 13, the comparison between the observed and the simulated flow on the 1st April, without excluding anomalies (STD SIM) and removing anomalies detected by ADP1 (ADP1 SIM) shows that the performances in 3 sensors locations are significantly affected by the anomaly detection process. The calibrators located in the position of sensors "R137_S11", "R047_S4" and "R027_SM125" before the data cleaning process were not able to correctly follow real measurements: "R137_S11" had some time slot with very high flow, "R047_S4" had zero flow for the most part of the simulation, and "R027_SM125" was not able to follow the real flow in the second half of the STD SIM. For these reasons, they were all classified as not aligned calibrators. After the data cleaning process, the similarity between the observed and the simulated flow rises, and the calibrators were classified as aligned. Finally, the outputs (flow and speed in every road section) of the simulations were compared. Since we removed some high flow values detected as anomalies, the expected result was a decrease in the total number of vehicles. Yet, the number of vehicles increased globally and the total number of vehicles per day increased on average in the 57% of road sections. The daily mean speed has a difference with an absolute value higher than 10 km/h for only 11% of road sections. Calculating the sum of the values of speed increment and speed reduction due to anomaly detection, the final result is that the data cleaning process speeds up the vehicles in the simulation. The road sections whose flow is more affected by the data cleaning process are displayed in the flow variation map in Figure 14. Roads with an increase in the mean daily flow for at least 20 days on 30 are colored in red; while roads with a decrease are colored in blue.

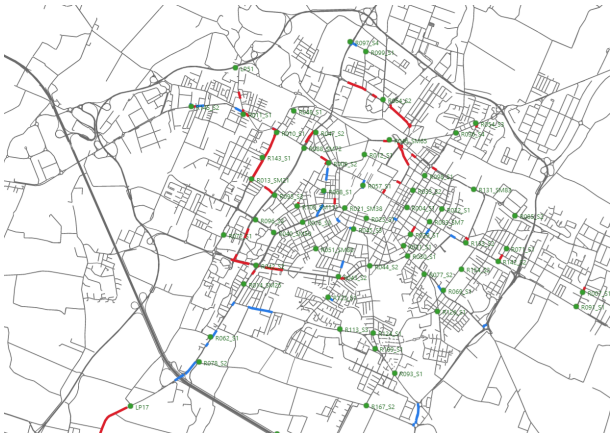


Fig. 14: The flow variation map built comparing the flow simulated by STD SIM and the one obtained by ADP1 SIM. The map can be seen in more detail at `https://trafair.eu/flowvariationmap/`.

8.3 Anomaly classification for tuning the anomaly detection process

Anomalies detected by ADP1 in April 2019 were classified. Firstly, the DCCA correlation coefficient was employed to identify for each sensor a group of correlated sensors referring to the measurements collected in April 2019, as described in Section 6.1. Then, anomalies of April 2019 were classified as described in Section 6. Several experiments have been conducted changing the amplitude of the time interval to define the best setup, considering that anomalies are detected on time series with a data rate of 15 minutes; the final choice was to consider an interval from 30 minutes before to 30 minutes after the detected anomaly. Section 8.2 pointed out that ADP1 detects a lot of anomalies during night hours. When these anomalies were classified, the majority of them (70%) are labeled as unusual traffic conditions. Moreover, 82% of unusual traffic conditions were during night hours. However, only 49% of anomalies classified as sensor faults were at night.

Furthermore, the corresponding value of flow in detected anomalies is very low during the night, suggesting that the unusual traffic conditions can be acceptable values that should not be excluded from the input of the traffic model.

The results of the classification on the anomalies detected by ADP1 underline the necessity of some improvement; hence, a new version of the anomaly detection algorithm that includes classification was generated: ADP2.

In Figure 15, the results of the classification for the two versions of the anomaly detection process can be compared; the value of traffic flow is very low for both sensor faults and unusual traffic conditions in ADP1. In ADP2 instead, the traffic flow values are significantly higher for sensor faults and remain lower for unusual traffic conditions. Besides, in ADP2 the percentage of unusual traffic conditions detected during night hours is reduced to 68% and 44% for sensor faults.

In Figure 16, the time series of measurements of sensor "R124_S2" is displayed, with sensor faults and unusual traffic conditions.

When unusual traffic conditions appear during night hours, they generally are a consequence of the inability of the STL decomposition to correctly extract the seasonality of night hours; thus, they are not significant for traffic evaluation. Instead, when unusual traffic conditions are detected during the day they may indicate traffic congestion or a real traffic event (e.g a road accident).

In Figure 17, the number of unusual traffic conditions during daylight hours for each day of April 2019 is displayed. The highest number of unusual traffic conditions is detected on Fridays (the 5th and the 12th) and Saturdays (the 6th and the 20th). The 23rd is a Tuesday, and it was the day after Easter Monday in Italy. Therefore, when a day is different from the days before, a high number of unusual traffic conditions is detected. This is a consequence of the STL parameters (the number of seasonality used by this version is 3), thus, if the day curve is different from the seasonality of the three days before, the remainder will be larger and more anomalies will be detected. As can be seen in Figure 18, this effect is not present in sensor faults; thus, the classification discriminates between anomalies that are a consequence of the adopted algorithm (classified as unusual traffic conditions) and real anomalies.

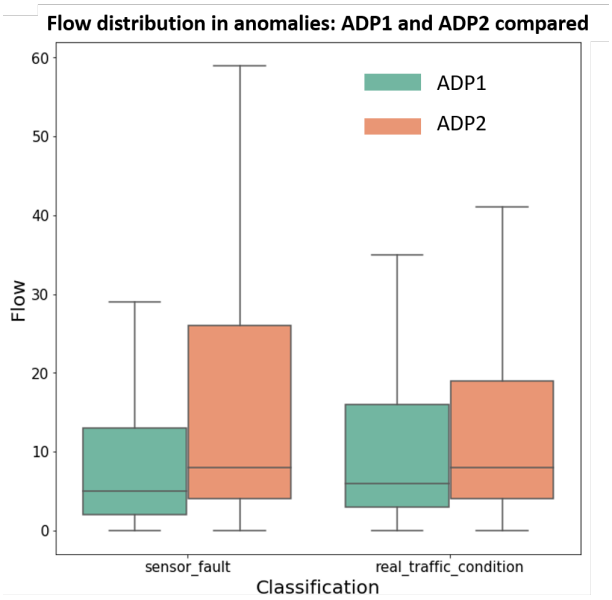


Fig. 15: Boxplots with the distribution of sensor faults (on the left) and unusual traffic conditions (on the right) with ADP1 (in green) and ADP2 (in orange).

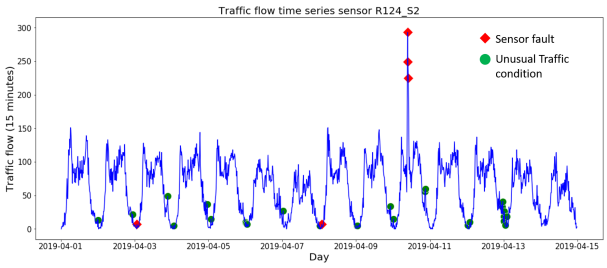


Fig. 16: "R124_S2" measurements time series with sensor faults in red and unusual traffic conditions in green. The depicted anomalies have been detected by ADP2.

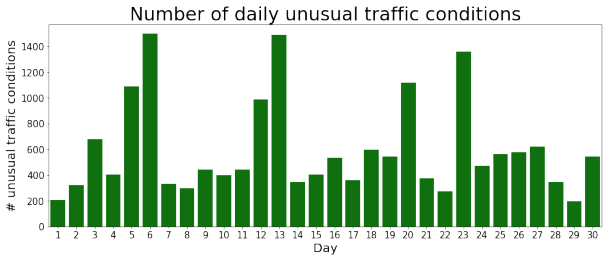


Fig. 17: Number of unusual traffic conditions during daylight hours for each day of April 2019.

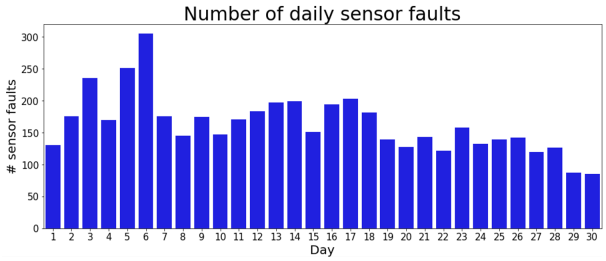


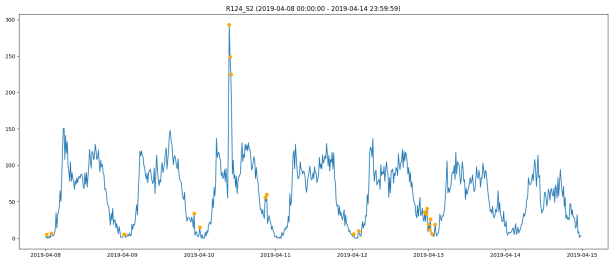
Fig. 18: Number of sensor faults during daylight hours for each day of April 2019.

8.4 Improvement in traffic simulation with anomaly detection and classification (ADP2+CLASS)

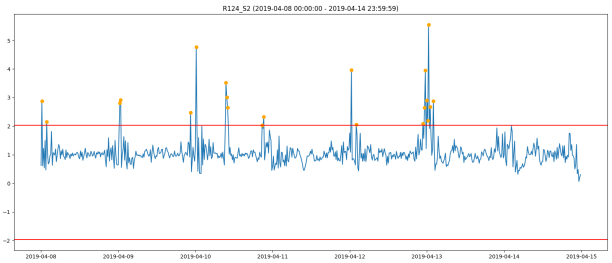
Different from experiments in subsection 8.2, in this part, we test the application of anomaly detection and classification in order to exclude from the input of the traffic simulation not all the anomalies but only the sensor faults.

ADP2 was used to detect anomalies in April 2019. Then, the detected anomalies were classified and only the ones labeled as sensor faults were removed from the traffic model input. Thus, in this case, we combine ADP2 with the anomaly classification.

ADP2 detected 26792 anomalies, which are related to 333 sensors. Compared to the number of anomalies detected by ADP1in the same period (as reported in subsection 8.2), ADP2 finds twice as many anomalies.



(a) Anomalies on the time series (i.e. the orange dots).



(b) Anomalies on the remainder of the time series (i.e. the orange dots).

Fig. 19: Anomalies found by ADP2 on sensor "R124_S2" observations from April 8th, 2019 to April 14th, 2019.

Figure 19.a shows anomalies detected by ADP2 on the time series observations of the sensor "R124_S2" from April 8th to April 14th, 2019, while Figure 19.b reports the same anomalies on the remainder component after the application of the inverse logarithm function (the red lines represent the lower fence and the upper fence).

These figures can be compared with the ones of Figure 10. Obviously, the remainder values of Figure 19.b and Figure 10.c are different since they are calculated in different ways, as described in Section 5.3. We can notice that anomalies on the high values of April 10th are identified by both anomaly detection processes; in addition, ADP2 finds another very high value in April 10th. Not all the anomalies on low flow values are detected by ADP2; indeed, we can notice that the low flow observations on the nights between 13th and 14th, and 14th and 15th are not highlighted as anomalies. Besides, at the end of April 10th ADP2 manages to identify anomalies on two positive peaks.

After anomaly detection, the traffic of each day of April 2019 was simulated using the traffic model described in Section 7. We compared the results of the simulation considering anomalous values (STD SIM) and the simulation excluding them (ADP2+CLASS SIM) for all days of April 2019. As displayed in Table 5, the average value of Mean DTW reduction is very high: 76.71. This is a significant improvement, considering that the same value was negative (-2.82) using ADP1.

DAY	% Aligned STD SIM	% Aligned ADP2+CLASS SIM	Mean RMSE STD SIM	Mean RMSE ADP2+CLASS SIM	Mean DTW reduction	Mean PWD reduction	Mean CTD reduction
4	0.83	0.82	31.19	29.66	-2.73	0.62	1
9	0.82	0.85	31.19	29.18	138	0.22	-1
12	0.80	0.77	32.14	31.93	-104.73	-2.07	4
17	0.81	0.82	35.70	32.00	111.12	1.19	0

Table 4: Comparison of traffic model evaluation metrics between traffic simulation including anomalies (STD SIM) and traffic simulation excluding sensors faults (ADP2+CLASS SIM).

Moreover, the average value of mean PWD reduction is 0.68 (it was 0.28 with ADP1); however, the average value of mean CTD reduction is 1 and is lower than the one in ADP1 (2).

Concerning the days with the worst performances using ADP1 (red values in Table 4), the values of the metrics described in Section 7.3 are evaluated considering only non-anomalous measurements for both the standard simulation (STD SIM) and the simulation performed removing sensor faults detected with ADP2 (ADP2+CLASS SIM). Since the anomalies detected by the two versions of the anomaly detection algorithm are different and the performances are evaluated only on non-anomalous values, the performances of the STD SIMs are different from the ones in Table 3 even if the simulation is the same.

The results show that ADP2 combined with classification significantly improves the performances of 4th, 9th, and 17th of April; however, removing anomalies still reduces the performance of April 12th, only the mean RMSE error is reduced.

9 Conclusion and future work

In this paper, we have presented a methodology to detect and classify anomalies in traffic sensor data streams. The proposed data cleaning process consists of three main components. The first one is a flow-speed correlation filter that removes unrealistic observations where the number of counted vehicles in a certain time interval is not related to the corresponding average speed. The second one is the anomaly detection algorithm that is based on the Seasonal-Trend Decomposition using Loess (STL). Two versions of the anomaly detection algorithm are defined and tested: ADP1 and ADP2. Anomaly classification performed investigating the cor-

	% Aligned STD SIM	% Aligned	Mean RMSE STD SIM	Mean RMSE	Mean DTW reduction	Mean PWD reduction	Mean CTD reduction
ADP1 SIM	0.84	0.87	27.33	25.32	-2.82	0.28	2
ADP2+CLASS SIM	0.86	0.86	29.83	24.99	76.71	0.68	1

Table 5: Comparison of average metrics on all April 2019 for STD SIM, ADP1 SIM and ADP2+CLASS SIM.

relation between sensors is the third component. The combination of ADP2 and anomaly classification has been proved to be the most effective in identifying the anomalies, as confirmed by experimental results.

Experiments proved that the classification of anomalies between sensor faults and unusual traffic conditions allows removing sensor faults from the input of the traffic simulation model, improving its performance, and ensuring that it can better emulate real urban traffic conditions.

The proposed solution that employs ADP1 is currently employed in real-time to detect anomalies on traffic sensor data. In the future, also ADP2 will be employed in real-time on our dataset of traffic sensor measurements. Moreover, we plan to experiment with different solutions for anomaly detection in multivariate time series considering both flow and speed.

Acknowledgements Research reported in this paper was partially supported by the TRAFAIR project (Grant Agreement: 2017-EU-IA-0167), co-financed by the Connecting Europe Facility of the European Union. Conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of EU Commission. The authors would like to thank those partners that contribute to the collection and management of traffic sensor data: the City of Modena and Lepida S.c.p.A..

References

1. Ahmed, M., Mahmood, A.: Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection. Annals of Data Science **2**(1), 111–130 (2015). DOI 10.1007/s40745-015-0035-y. URL `https://www.scopus.com/inward/record.uri?eid=2-s2.0-84950287149&doi=10.1007%2fs40745-015-0035-y&partnerID=40&md5=6bea39d13fd3d2713d20043c005cad50`. Cited By 31
2. Bachechi, C., Po, L.: Implementing an urban dynamic traffic model. In: 2019 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2019, Thessaloniki, Greece, October 14-17, 2019, pp. 312–316. ACM (2019). DOI 10.1145/3350546.3352537. URL `https://doi.org/10.1145/3350546.3352537`
3. Bachechi, C., Po, L.: Traffic analysis in a smart city. In: Web4City, International IEEE/WIC/ACM Smart City Workshop: Web for Smart Cities - In conjunction with IEEE/WIC/ACM International Conference on Web Intelligence, WI'19, Thessaloniki, Greece, Oct. 14-17, 2019 (2019). To appear
4. Bachechi, C., Rollo, F., Desimoni, F., Po, L.: Using real sensors data to calibrate a traffic model for the city of modena. In: Proceedings of the 3rd International Conference on Intelligent Human Systems Integration (IHSI 2020), February 19-21, 2020, Modena, Italy, pp. 468–473 (2020). DOI 10.1007/978-3-030-39512-4_73. URL `https://doi.org/10.1007/978-3-030-39512-4_73`
5. Bachechi, C., Rollo, F., Po, L.: Real-time data cleaning in traffic sensor networks. In: 17th IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2020, Antalya, Turkey, November 2-5, 2020, pp. 1–8. IEEE (2020). DOI 10.1109/AICCSA50499.2020.9316534. URL `https://doi.org/10.1109/AICCSA50499.2020.9316534`
6. Benesty, J., Chen, J., Huang, Y., Cohen, I.: Pearson Correlation Coefficient. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). DOI 10.1007/978-3-642-00296-0_5. URL `https://doi.org/10.1007/978-3-642-00296-0_5`
7. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03, pp. 359–370. Seattle, Washington, USA (1994)
8. Braei, M., Wagner, S.: Anomaly detection in univariate time-series: A survey on the state-of-the-art. CoRR **abs/2004.00433** (2020). URL `https://arxiv.org/abs/2004.00433`
9. Chen, Z., Yu, X., Ling, Y., Song, B., Quan, W., Hu, X., Yan, E.: Correlated anomaly detection from large streaming data. In: IEEE International Conference on Big Data, Big Data 2018, pp. 982–992 (2018). DOI 10.1109/BigData.2018.8622004. URL `https://doi.org/10.1109/BigData.2018.8622004`
10. Cleveland, R.: Stl: A seasonal-trend decomposition procedure based on loess. Journal of Official Statistics **6** (1990)
11. Giridhar, P., Amin, M.T., Abdelzaher, T., Wang, D., Kaplan, L., George, J., Ganti, R.: Clarisense+: An enhanced traffic anomaly explanation service using social network feeds. Pervasive and Mobile Computing **33**, 140 – 155 (2016). DOI https://doi.org/10.1016/j.pmcj.2016.03.005
12. Görnitz, N., Kloft, M., Rieck, K., Brefeld, U.: Toward supervised anomaly detection. Journal of Artificial Intelligence Research (JAIR) **45** (2012). DOI 10.1613/jair.3623
13. Kurian, N., Thomas, A., George, B.: Automated fault diagnosis in multiple inductive loop detectors. 11th IEEE India Conference: Emerging Trends and Innovation in Technology, INDICON 2014 (2015). DOI 10.1109/INDICON.2014.7030431
14. Lee, S., Kim, H.K.: Adsas: Comprehensive real-time anomaly detection system. In: B.B. Kang, J.S. Jang (eds.) Information Security Applications - 19th International Conference, WISA 2018, Jeju Island, Korea, August 23-25, 2018, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 11402, pp. 29–41. Springer (2018). DOI 10.1007/978-3-030-17982-3_3. URL `https://doi.org/10.1007/978-3-030-17982-3_3`
15. Liu, S., Wright, A., Hauskrecht, M.: Online conditional outlier detection in nonstationary time series. In: Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2017, Marco Island, Florida, USA, May 22-24, 2017, pp. 86–91.

AAAI Press (2017). URL https://aaai.org/ocs/index.php/FLAIRS/FLAIRS17/paper/view/15486
16. López, P.Á., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., WieBner, E.: Microscopic traffic simulation using SUMO. In: 21st International Conference on Intelligent Transportation Systems, ITSC 2018, Maui, HI, USA, November 4-7, 2018, pp. 2575–2582. IEEE (2018). DOI 10.1109/ITSC.2018.8569938. URL https://doi.org/10.1109/ITSC.2018.8569938
17. Po, L., Rollo, F., Bachechi, C., Corni, A.: From sensors data to urban traffic flow analysis. In: 2019 IEEE International Smart Cities Conference, ISC2 2019, Casablanca, Morocco, October 14-17, 2019, pp. 478–485. IEEE (2019). DOI 10.1109/ISC246665.2019.9071639. URL https://doi.org/10.1109/ISC246665.2019.9071639
18. Po, L., Rollo, F., Viqueira, J.R.R., Lado, R.T., Bigi, A., López, J.C., Paolucci, M., Nesi, P.: TRAFAIR: understanding traffic flow to improve air quality. In: 2019 IEEE International Smart Cities Conference, ISC2 2019, Casablanca, Morocco, October 14-17, 2019, pp. 36–43. IEEE (2019). DOI 10.1109/ISC246665.2019.9071661. URL https://doi.org/10.1109/ISC246665.2019.9071661
19. Priestley: Non-linear and non-stationary time series analysis. London : San Diego : Academic Press (1988)
20. Ramchandran, A., Sangaiah, A.K.: Chapter 11 - unsupervised anomaly detection for high dimensional data—an exploratory analysis. In: A.K. Sangaiah, M. Sheng, Z. Zhang (eds.) Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications, Intelligent Data-Centric Systems, pp. 233 – 251. Academic Press (2018). DOI https://doi.org/10.1016/B978-0-12-813314-9.00011-6
21. Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. Intell. Data Anal. **11**(5), 561–580 (2007). URL http://dl.acm.org/citation.cfm?id=1367985.1367993
22. Schober, P., Boer, C., Schwarte, L.: Correlation coefficients: Appropriate use and interpretation. Anesthesia and Analgesia **126**, 1 (2018). DOI 10.1213/ANE.0000000000002864
23. İsmail Sezen, Unal, A., Deniz, A.: Anomaly detection by stl decomposition and extended isolation forest on environmental univariate time series. In: EGU General Assembly 2020, Online, 4–8 May 2020 (2020). DOI https://doi.org/10.5194/egusphere-egu2020-18471
24. Souto, G., Liebig, T.: Analyzing the correlation among traffic loop sensors to detect anomalies in traffic loop data streams. In: BRACIS (ed.) Proceedings of the 3rd Symposium on Knowledge Discovery and Machine Learning (2015)
25. Wang, X., Wang, C.: Time series data cleaning: A survey. IEEE Access **8**, 1866–1881 (2020). DOI 10.1109/ACCESS.2019.2962152. URL https://doi.org/10.1109/ACCESS.2019.2962152
26. Wen, Q., Gao, J., Song, X., Sun, L., Xu, H., Zhu, S.: Robuststl: A robust seasonal-trend decomposition algorithm for long time series. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pp. 5409–5416 (2019). DOI 10.1609/aaai.v33i01.33015409. URL https://doi.org/10.1609/aaai.v33i01.33015409
27. Xu, N., Shang, P., Kamae, S.: Modeling traffic flow correlation using dfa and dcca. Nonlinear Dynamics **61**(1), 207–216 (2010)
28. Zebende, G.: Dcca cross-correlation coefficient: Quantifying level of cross-correlation. Physica A: Statistical Mechanics and its Applications **390**(4), 614 – 618 (2011). DOI https://doi.org/10.1016/j.physa.2010.10.022. URL http://www.sciencedirect.com/science/article/pii/S0378437110008800
29. Zhao, X., Shang, P., HUANG, J.: Several fundamental properties of dcca cross-correlation coefficient. Fractals **25**, 1750017 (2017). DOI 10.1142/S0218348X17500177
30. Zhu, L., Krishnan, R., Sivakumar, A., Guo, F., Polak, J.W.: Traffic monitoring and anomaly detection based on simulation of luxembourg road network. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 382–387 (2019)
31. Zygouras, N., Panagiotou, N., Zacheilas, N., Boutsis, I., Kalogeraki, V., Katakis, I., Gunopulos, D.: Towards detection of faulty traffic sensors in real-time. In: MUD@ICML (2015)