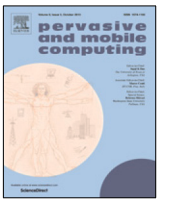
Pervasive and Mobile Computing 91 (2023) 101769

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc



Public transport congestion detection using incremental learning

Laszlo A. Makara [a], Petar Maric [a], Adrian Pekar [a,b,*]

[a] Department of Networked Systems and Services, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary
[b] ELKH-BME Information Systems Research Group, 1117, Magyar Tudosok krt. 2, Budapest, Hungary

ARTICLE INFO

Article history:
Received 23 August 2022
Received in revised form 18 February 2023
Accepted 28 February 2023
Available online 1 March 2023

Keywords:
Public transport
Trajectory data
Bus GPS data
Congestion detection
Incremental learning

ABSTRACT

In the past decade, intelligent transportation systems have emerged as an efficient way of improving transportation services, while machine learning has been the key driver that created scopes for numerous innovations and improvements. Still, most machine learning approaches integrate paradigms that fell short of providing cost-effective and scalable solutions. This work employs long short-term memory to detect congestion by capturing the long-term temporal dependency for short-term public bus travel speed prediction to detect congestion. In contrast to existing methods, we implement our solution as incremental learning that is superior to traditional batch learning, enabling efficient and sustainable congestion detection. We examine the real-world efficacy of our prototype implementation in Pécs, the fifth largest city of Hungary, and observed that the incrementally updated model can detect congestion of up to 82.37%. Additionally, we find our solution to evolve sufficiently over time, implying diverse real-world practicability. The findings emerging from this work can serve as a basis for future improvements to develop better public transportation congestion detection.

1. Introduction

The quality of life and living standards in large cities depend substantially on the efficiency of their transportation systems, which move people, goods, and services between various locations. However, as global urbanization continues, population densities in cities inevitably increase [1]. One consequence of this trend is a continuous and rapid growth in the number of vehicles on the roads, leading to one of the most pressing problems faced by modern cities: traffic congestion [2]. Congestion has several adverse effects on economic productivity, environmental quality, and safety, including worsened safety conditions, higher fuel consumption, increased air pollution, and raised costs of goods and services [3–7]. Rapid and real-time detection and prediction of traffic congestion is, therefore, a critical task.

The past decade has led to significant advances in solving complex tasks through machine learning (ML), and traffic congestion was no exception. Intelligent transportation systems (ITS) have emerged as an efficient way of improving the performance of transportation and enhancing travel security [2]. ITS typically detects and predicts congestion based on the analysis of trajectory data [3], which are characterized by being generated from multiple sources and devices and continuously transmitted in a variety of formats. Still, most ML approaches integrate disruptive technology paradigms

* Corresponding author at: Department of Networked Systems and Services, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary.
E-mail addresses: lmakara@hit.bme.hu (L.A. Makara), petar.maric@edu.bme.hu (P. Maric), apekar@hit.bme.hu (A. Pekar).

https://doi.org/10.1016/j.pmcj.2023.101769

falling short of providing a cost-effective, scalable, and real-time ITS [8]. The structure of traditional machine learning algorithms (MLA) is often complex [9], while training a high-performance model is typically time-demanding [10]. Additionally, they also lack proper capacity scalability and sustainability [11]. Public transport traffic is dynamic, with its volume and heterogeneity continuously increasing. Consequently, storage and computing resource consumption of MLAs is challenging to keep at a sustainable level [12]. Additionally, with the perpetually changing nature of measurement data, the traditional ML-driven models often get outdated. As a result, they often come short of abstracting the real world properly. Thus, developing models suitable for real-time data flow analysis to maintain adaptability, efficiency, and scalability are in great demand.

This present work aims to overcome the limitations of traditional ML approaches under-explored in current ITS by implementing public transport traffic flow congestion detection through incremental learning (IL) [13–16]. Specifically, it investigates the feasibility of IL for real-time, adaptive, and scalable public transport congestion detection. IL is employed via long short-term memory (LSTM) to detect congestion by capturing the long-term temporal dependency for short-term public bus travel speed prediction. The key question to be addressed in our research are: *How to design a proper workflow in conformity with the IL paradigm? Quantitatively, what is the efficacy of such an IL methodology in a real-world application?*

We examine the real-world efficacy of our solution in Pécs, the fifth largest city in Hungary. Based on the evaluation, the incrementally updated model detects congestion with an accuracy of up to 82.37%. Furthermore, we also assessed the capability of the model to evolve over time and found that, depending on the frequency of iterations, the accuracy of travel speed estimation can ramp up to 221.46% by as little as over a period of six days, implying the diverse applicability of our solution. Additionally, its resource consumption yielded comparable results to that required by traditional learning, showing promise for more efficient application in constrained environments. Moreover, our proposed solution is adaptive towards high accuracy, scalable, and operates in real-time. It can reduce resource consumption while providing a more reliable basis for congestion detection to improve public transport services, while in terms of scalability, the incrementally updated model expands in tandem with streaming data dimensions for model evolution and faster convergence. The novelty of this work lies in the following factors:

- We design a framework for public transport congestion detection via IL as a viable alternative to the existing traditional ML-based solution.
- We provide a methodology for incremental model training to yield high accuracy and adaptability.
- We compare the resource utilization of our solution to traditional machine learning employed across existing works to assist researchers and professionals in identifying critical factors that may impact their selection of the used methodology.

To the best of our knowledge, this present work is the first to approach the problem domain from this perspective, as no work has been carried out to date on incremental learning-based congestion detection using public bus trajectory data. Beyond the scope of this present work, the findings emerging from our research can fuel future improvements in travel-time and reliability estimation, arrival time prediction, and bus bunching and on-time performance estimation.

The rest of this paper is organized as follows. Section 2 provides a brief background including congestion control, incremental learning and related work. Section 3 presents our IL-based approach for traffic congestion detection. Section 4 discusses the results achieved through rigorous examination of the efficacy of our approach. Finally, in Section 6 we conclude this paper and provide the intended directions of future research.

2. Background and related work

During the last decade, the methodological landscape of traffic congestion has gradually integrated a wide variety of methods into congestion detection. In what follows, we briefly overview them in the context relevant to this paper.

2.1. Trajectory data

A vehicle trajectory is a path generated by a moving vehicle in space [17], and vehicle trajectory data record the movement of individual vehicles [18]. The relationship between traffic speed, flow rate, and density obtained from trajectories can help to examine traffic conditions better within arbitrarily chosen regions [19]. Thus, such data continue to show steady gain in popularity for a wide range of applications, including travel time estimation [20], driver behavior profiling [21], and traffic flow forecasting.

Trajectory data are typically gathered via image and video processing and mobile sensing. The former is slowly felling out of favor due to its substantial data-processing efforts and proneness to low video and image quality errors [22]. On the other hand, the popularity of mobile sensing is steadily increasing [23], which stems from the proliferation of smartphones having at least two location providers — cellular/WiFi and Global Positioning System (GPS). Mobile phone location data are deemed valid for traffic sensing purposes, capable of detecting various events on the roads rapidly [24]. Additionally, technological advancements have extended the capabilities of in-vehicle sensors and on-board units to also periodically report trajectory data along with other readings such as traffic volume, speed, and road segment occupancy [25,26].

2.2. Traffic congestion

Traffic congestion is essentially comprised of three components [27]: intensity (amount), extent (area or network coverage), and duration (how long it lasts). Broadly speaking, congestion occurs when vehicle space occupancy (for example, on roadways, sidewalks, and transit lines) reaches an unacceptable level of delay. Assume congestion is expressed as the number of vehicles per unit length of roadway; then, as space occupancy increases, the speed of movement decreases, entailing congestion [27]. However, 'the difference in travel times experienced during busy and lightly traffic periods', 'the ratio of actual and uncongested travel times', and 'actual versus uncongested travel times rates' [27] are also all valid and well-grounded definitions of congestion.

In this work, we limit our discussion to *speed* and *congestion indices*, however, it is noteworthy that other indices such as travel rate, delay, and level of services are also valid and frequently used measures for quantifying the level of congestion [28–30].

2.2.1. Speed

Speed reduction index (SRI) and *speed performance index* (SPI) are two measures for determining congestion via traffic speed [31,32]. SRI is given as the ratio of the relative speed change between congested and free-flow conditions [28], expressed as

$$SRI = (1 - \frac{v_{act}}{v_{ffs}}) \times 10, \tag{1}$$

where v_{act} denotes the actual travel speed and v_{ffs} indicates the free-flow speed threshold whose typical values include the average speed of the off-peak period and the 85th percentile of the off-peak speed. The ratio ranges between 0 and 10, while congestion is considered when SRI exceeds 4 or 5 [33].

SPI is determined by the ratio between the vehicle speed and the maximum permissible speed [28], formally

$$SPI = (\frac{v_{avg}}{v_{max}}) \times 100, \tag{2}$$

where v_{avg} denotes the average travel speed and v_{max} indicates the maximum permissible road speed. SPI ranges from 0 to 100 while the traffic state-levels are typically determined using three thresholds [28]: heavy congestion for anything below 25, mild congestion between 25 and 50, smooth traffic from 50 to 75, and very smooth over 75.

2.2.2. Congestion indices

Relative congestion index (RCI) is estimated as

$$RCI = \frac{T_{act} - T_{fft}}{T_{fft}}, \tag{3}$$

where T_{act} is the actual travel time (estimated as the ratio of spatial length and spatial mean speed) and T_{fft} is the free-flow travel time (determined as the ratio of spatial length and free-flow speed) [28,34,35]. RCI of 0 denotes low to no congestion while values greater than 2 indicates a high congestion level.

Road segment congestion index (RSCI) measures the normal road segment state relative to the duration of the non-congested state during the observation period [28], expressed as

$$RSCI = \frac{SPI_{avg}}{100} \times \frac{t_{NC}}{t_t}, \tag{4}$$

where t_{NC} denotes the duration of the non-congested state and t_t is the length of the observation period [34,35]. RSCI ranges from 0 to 1 while the smaller value indicates a more congested road segment [28].

2.3. Incremental learning

Modern application requirements of traditional ML approaches include the ability of the approach to be trained using data streams, reliable performance, short runtime in the face of constrained resources, and lifelong learning [36]. However, although progress has been made towards meeting such capabilities [37], existing ML approaches lack their simultaneous fulfillment. Incremental learning appears a suitable alternative to overcome these limitations [38]. It is built on the premise of continuous learning and adaptation, enabling the autonomous incremental development of complex skills and knowledge.

In ML context, IL aims to smoothly update the prediction model to account for different tasks and data distributions while still being able to re-use and retain knowledge over time. The main difference between both settings is the fact that IL provides a continuous iterative process. In contrast, the traditional batch setting is divided into two phases, training and testing, that must be repeated every time a model update is necessary. IL can build its prediction capabilities on top of what was previously learned, amending previous errors and shortcomings while efficiently adapting to new environmental conditions as new data becomes available. This way, the systematic re-training can be avoided, saving time and computational resources while maintaining the model's performance.

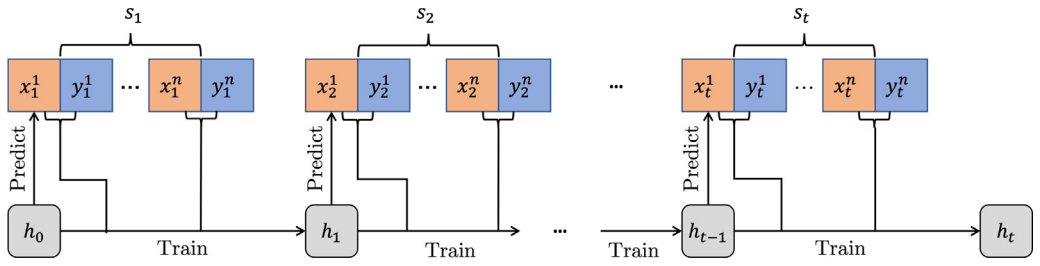


Fig. 1. Incremental learning: a sequence of models $h_1, h_2, \ldots, h_t$ are developed using batches s_t of new data with size n, where x_t^i, y_t^i indicate data i for batch t.

Fig. 1 illustrates IL. The workflow closely follows the generic online IL model [37]. The sequence of models $h_1, h_2, \ldots, h_t$ are developed using batches $s_1, s_2, \ldots, s_t$ of data stream. Data batches of size n (s_n) are used to update the model. To this end, they are analogous to data blocks used in traditional (offline) machine learning. However, each new data batch is used only once as opposed to traditional learning, where the model is trained with new data through multiple epochs. Accordingly, $s_t = \{(x_t^1, y_t^1), \ldots, (x_t^n, y_t^n)\} \in R^p \times \{1, \ldots, M\}$ is the set of new data stream blocks, where p indicates data dimension and M is the number of classes. The model $h_t : R^p \rightarrow \{1, \ldots, M\}$ depends solely on model h_{t-1} and the most recent block of new data s_t composed of n classes.

Note that we differentiate between online learning and incremental learning. Online learning updates the model using each incoming data point without storing [39]. Therefore, it can typically manage massive data volumes arriving at a high rate [40]. IL essentially learns from data batches at explicit intervals typically achieved via micro-batching [41]. Hence, it can operatively update the model with new data points while maintaining its existing knowledge intact. Besides its capability of stabilizing the historical knowledge of the model over novel learning, IL can also identify non-recurrent concept drifts and utilize them for real-time propagation and prediction [40].

2.4. Related work

Vehicular network traffic has been a subject of several research studies over the last decade. Many researchers have developed methods for various road traffic problems. The efficacy of machine learning for vehicle classification using roadside sensors has been rigorously examined in [42,43]. The mathematical landscape of methods used for traffic flow forecasting include Hidden Markov models [44], gradient boosting regression tree [45], artificial neural networks [46], decision trees [47], support vector machines [48], Long short-term memory (LSTM) [49,50], and Bayes networks [51]. Recent advances in traffic flow forecasting also include deep learning-based techniques [52–56] and ensemble approaches [57–60].

Machine learning is undoubtedly primed to overcome the complexity and multi-dimensionality of vehicular road network data, supported by the increasing volumes of labeled traffic datasets available across many urban areas. Yet, many techniques to solve various traffic problems fall short of appropriately handling the emerging need for processing traffic flow data provided by sensor networks and IoT in a streaming and resource-saving manner. There is only a handful of works aimed at congestion detection based on incremental learning. Noteworthy research is perhaps a trajectory clustering approach to segment and detect traffic behaviors automatically [8,40], which applies an incremental learning technique, specifically the Incremental Knowledge Acquiring Self-Learning [61,62] algorithm, to incrementally learn trajectory clusters and changes over time.

Existing research aimed at solving road traffic problems using public bus data focus on multi-modal contextual sensing for sub-60 s bus arrival time prediction [63], traffic congestion of road segments between bus stops using self-organizing map [64], and LSTM-based urban traffic congestion mapping using bus mobility data [65]. Bus trajectory data, along with publicly available bus schedule data, have also been extensively studied to measure the performance of public transportation systems by utilizing several metrics, including travel-time reliability, on-time performance, bus bunching, and travel-time estimation [66–69]. However, no work has been carried out to date on IL-based congestion detection using public bus trajectory data. To the best of our knowledge, this present paper is the first to tackle closing this gap.

3. Methodology

Fig. 2 shows the high-level overview of the implemented system. Model training runs in a loop. The continuous flow of trajectory data observed from the buses is stored in a database. During training, available data are first loaded for pre-processing and then parsed into sequences. Depending on the number of routes being determined, either model training is called, or the associated LSTM is updated according to the number of inputs and outputs required to handle the trajectories appropriately. Otherwise, the LSTM remains intact. Once a new model is trained, its efficacy is compared to the latest model stored in the model database. The comparison is performed using test data. If the model outperforms the latest model stored in the database, it is stored as the newest model. Otherwise, the newly trained model is discarded. In what follows, we describe the modules listed above in more detail.

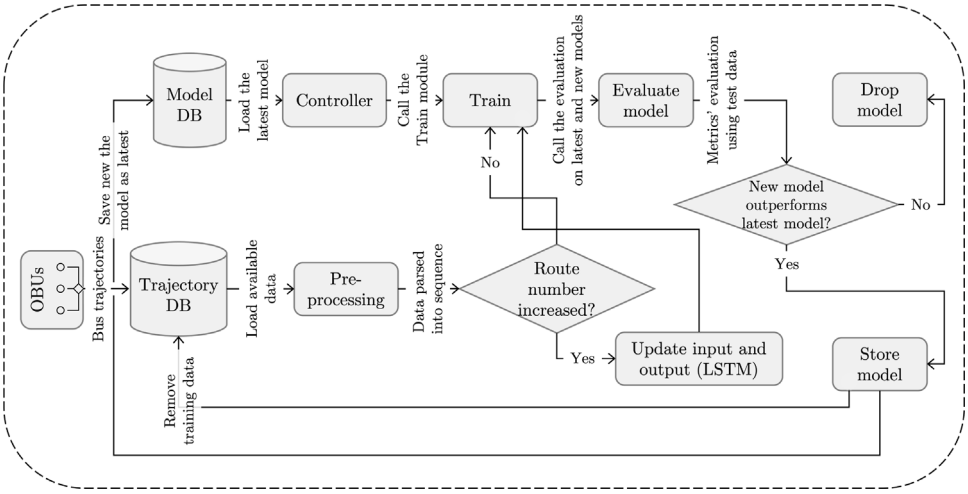


Fig. 2. High-level overview of the system.

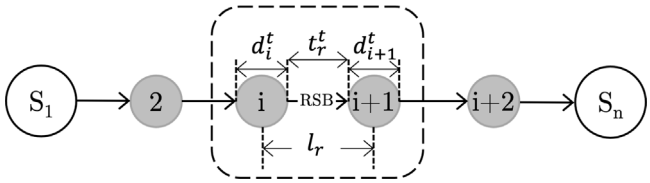


Fig. 3. General bus route considered in the present work, where S_1, S_n are the terminal stops of the bus line; *RSB* denotes one road segment between adjacent bus stops of the bus line; i indicates the index of the bus stop; l^r is the length of the *RSB* r; d_i^t is the bus dwell time; and t_r^t is the travel time.

3.1. Average travel speeds of bus routes

Closely following the reference model described by Gu et al. [64], the general bus line considered in this work is of length l and consists of n bus stops, as shown in Fig. 3. The terminal stops of the bus line are denoted by S_1 and S_n, while *RSB* indicates one road segment between adjacent bus stops of the bus line. The bus dwell time at bus stop i represents the time required for passenger alighting and boarding, as depicted in Fig. 3. The bus dwell time and travel time between adjacent bus stops i and $i+1$ were estimated by the method developed by Weng et al. [70].

In light of Section 2.2, the average travel speed (ATS) of *RSB* r in time interval t is determined as

$$ATS = \frac{s_1 + s_2 + \cdots + s_n}{n}, \tag{5}$$

where s_n is the ATS of each bus in *RSB* r in time interval t and n is the number of buses. The status of the traffic flow is estimated based on the average speed of each bus in an *RSB*.

3.2. Trajectory data

Bus trajectory data is an efficient and economical method to monitor the operational condition of the public transit network, reflect the possible traffic conditions, and evaluate the potential consequences of road infrastructure changes, such as congestion [64]. The primary source for trajectory datasets is typically global positioning system (GPS) data usually collected by in-vehicle on-board units. To this end, the input for our congestion detection algorithm is vehicle GPS trajectories that comprise a sequence of sampled data points with location and timestamp information.

3.3. Pre-processing

Based on the trajectory data obtained from the sensors, continuous curves composed of time series of velocities are formed for each bus. The granularity of the velocities are determined by the sampling frequency of the sensors. The velocity values are not normalized in our approach since normalization might bring a bias into the velocities whose minimum and maximum are unknown considering the continuous flow of trajectory data at any given time (it is oblivious to assume that the typical maximum speed of 50 km/h is not exceeded by buses in practice, especially during night services). Fig. 4 depicts

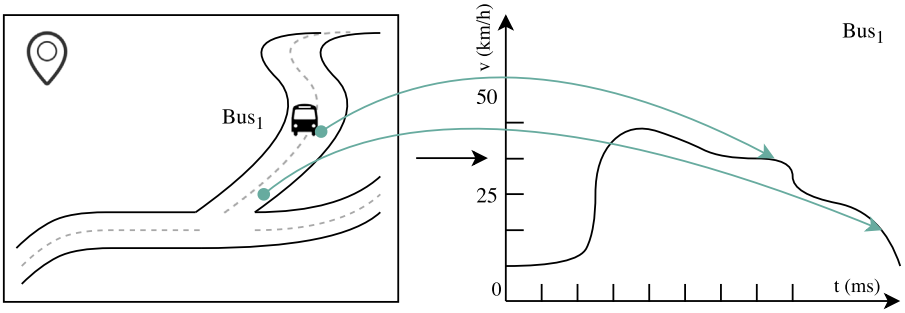


Fig. 4. Generating time series from bus trajectory data.

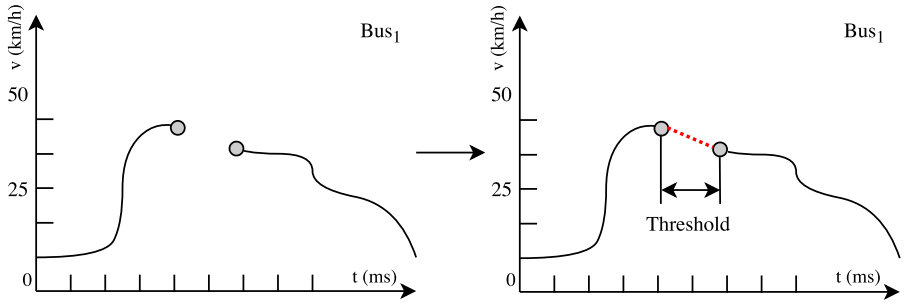


Fig. 5. Imputing missing data points.

the schematic diagram of the mapping from the trajectory data into continuous curves of time series in the coordinate system.

We would ordinarily assume that all data points are available between two observations. However, this is seldom the case, while the missing data points might often represent critical information reflecting the operational condition of traffic flows. Such a scenario is depicted in Fig. 5, where data points $[t_{i+2}, t_{i+5}]$ represent missing values in the series of t_i, t_{i+1}, t_{i+6}. Since such data points might not be recoverable in every case (due to errors in the underlying measurement instrumentation), we specify a threshold T, which determines the maximum affordable error between two samples. If the observed sample $t_j - t_i \leq T; j, i \in Z^+, i \leq j$ is within this limit, a liner movement is assumed between the two points, as depicted in Fig. 5. Otherwise, if the data point exceeds this threshold, assuming a uniform deceleration of the bus, the velocity of the bus is mapped to zero.

Suppose that all missing data points were appropriately addressed (or no missing points were present in the samples), the time series is formed by the combination of those bus trajectory data points that travel the same route, expressed as

$$\overline{R_i} = \frac{\sum_{j=0}^{n_i} B_{ij}}{n_i}, \quad R_i \subset \{B_{i0}, B_{i1}, B_{i2}, \dots\}, \tag{6}$$

where R_i denotes the ith route, B_{ij} indicates the jth bus traveling route i, whose numbering starts from zero, and n_i denotes the number of buses traveling route R_i. Fig. 6 demonstrates how such routes are formed. As a result, the bus routes combined in this way are mapped to a common interpretation domain, *i.e.* $\overline{R_i}$, which denotes hereinafter the ATS calculated for the ith route.

The individual time series are aligned according to the sampling moments. This is necessary for a coherent operation to avoid bus time series that are skewed in time relative to the routes. Additionally, data with the same windows must be used for training, that is, all the routes relevant for model training should be of the same length. This is necessary to ensure that the input is of sufficient size, required for transformation in later steps. If the lengths are not matching, the shorter ones are padded with zeros.

3.4. Data storage

Our methodology employs two databases: one for storing generated models and the other for temporarily storing trajectory data during model development. Their brief description is as follows.

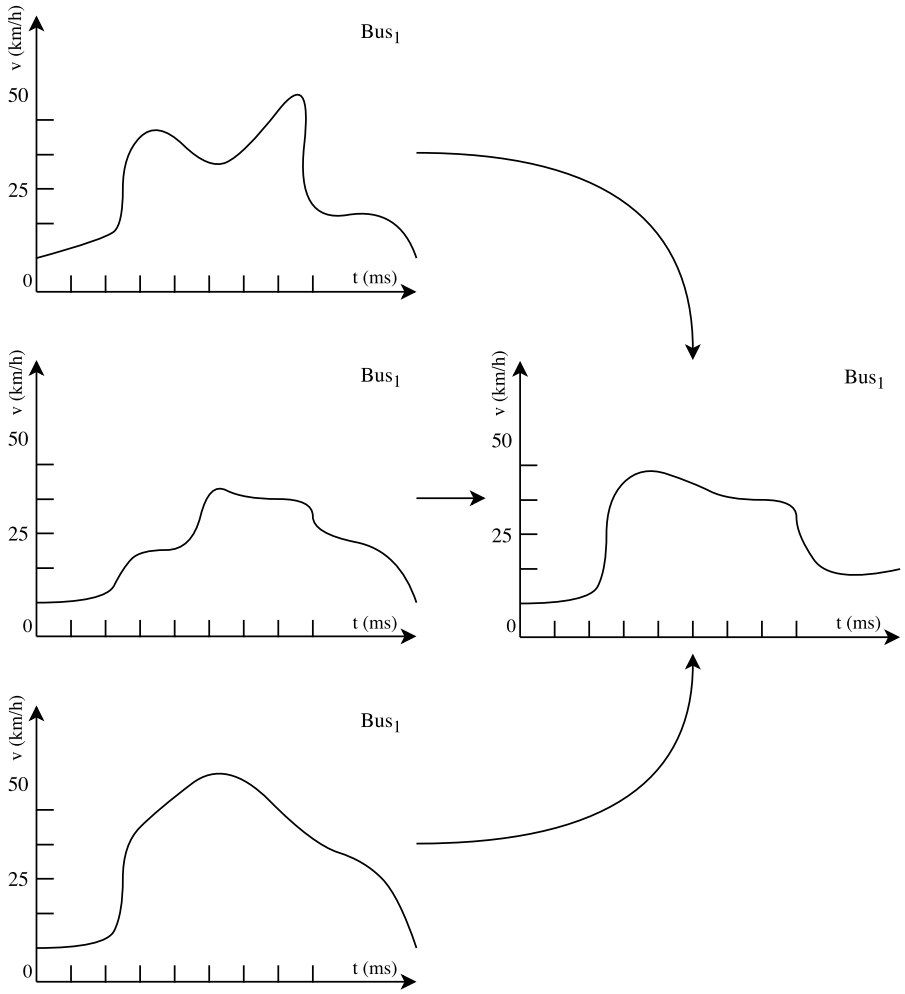


Fig. 6. Route forming.

3.4.1. Model database

The generated models are primarily stored to revert the system to an earlier knowledge (model) state. The models are stored in the database as BSOD objects, a Binary JSON object used by MongoDB. Their structure is given by the serialization function of the Keras built-in model. Additionally, the model database is also used by the prediction engine. Each prediction is preceded by checking the database for the most recent model. Given a newer version is located, it is used for the prediction. Otherwise, the last known model is used by the prediction engine.

3.4.2. Temporary database

The continuous flow of trajectory data is stored in a temporary database. The format of the trajectory data gives the structure of the stored data, which is loaded into the database post serialization. During training, samples previously used are automatically discarded by the workflow, as they have been previously built in the model. Such records are permanently erased once the efficacy of the newly generated model is assessed. If its accuracy is lower than that of the latest one, the new model is discarded while the data used for training is preserved in the temporary database. Otherwise, the new model is kept while the data used for its generation is erased.

3.5. Controller

The Controller module performs management tasks that are prerequisites before the Train phase. This component is responsible for loading the latest model into the system to deserialize and load it into memory properly. After loading the model, it stores the metrics locally in memory, used for later comparison. Furthermore, it is also responsible for initializing the system if no previous model has been generated yet.

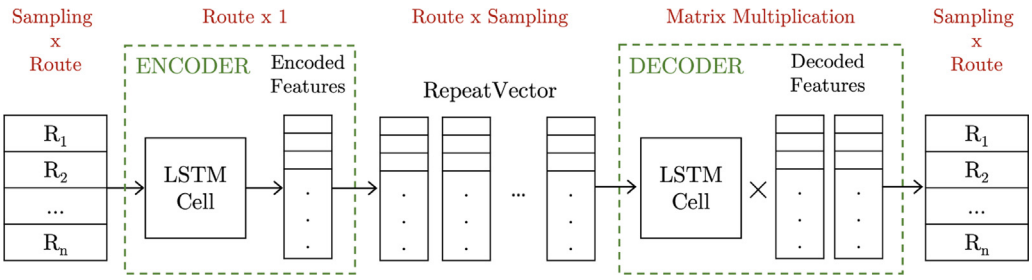


Fig. 7. The architecture of the LSTM autoencoder used in the methodology, which consists of an encoder and a decoder. The encoder generates the encoded features learned by an input data function. A bridge then prepares the encoded outputs to be received by the decoder branch LSTM block. The decoded features will be formed implemented by a TimeDistributed layer that multiplies the output by a matrix so that the dimension of the output O equals the input dimension I ($\dim(I) = \dim(O)$), where the dimension of I is given by its width denoted with $|R_{jm}|$, $0 < j < (n+1) - \Gamma$, and height expressing the number of unique bus routes. Then, a vector of the size of 'Route $\times$ 1' appears on the output of the encoder branch, which contains the encoded features. The bridge between the encoder and decoder is implemented as a RepeatVector that multiplies the encoded output features to produce a matrix of a size of 'Route $\times$ Sampling.' The output of the decoder is brought to the same size as the input matrix by additional matrix multiplication implemented using a TimeDistributed function. The condition of return sequences must be met for all the output LSTM cells.

3.6. Input/output update

Our methodology employs Long-Short Term Memory (LSTM), a type of recurrent neural network (RNN) capable of learning order dependence in sequence prediction problems [71]. The LSTM network has a fixed number of input and hidden layers by default. As the input dimension increases, the individual elements are shifted, giving space to new values. In principle, no missing values occur on the input. However, in reality, this is seldom the case. Thus, we load empty inputs as NaN rather than zeros during preprocessing, so no weights are optimized based on them. The structure of the input is of the form of $x = [R_0, R_1, \ldots, R_i, NaN_0, \ldots, NaN_K]$, where K denotes the maximum dimension of the input. This higher bound was set based on pilot measurements at the time of proof of concept implementation. However, future study is required to investigate in detail how different sizes might affect system operation.

Assume that at time T the input is sorted as $x_T = [R_0, R_1, R_5, R_{10}, \ldots, R_i, NaN_0, \ldots, NaN_K]$, where data $[R_2, R_4]$ and $[R_6, R_9]$ are missing. Then, inserting an out-of-order data, *e.g.*, R_3, into the dataset would yield a problem as it shifts the preceding data points, eventually leading to erroneous prediction. That is, for R_3 the prediction would be based on the previous data of R_5, while for R_{10}, it would be based on NaN. We overcome this problem via mapping where the route number is not bound to the input location. Nonetheless, the mapped serial numbers assigned incrementally at new data point arrivals must be stored globally so that the correct position of the items is maintained across model updates.

Furthermore, before model training takes place, the input and output are assessed for a change compared to their states in the previous step. This can occur when a new bus route path (service) appears in the system that have not been observed before. That is, formally, input $x_{T+1} = [R_0, R_1, \ldots, R_i, R_{i+1}, \ldots, R_{i+k}]$ at time $T+1$ has increased in contrast to $x_T = [R_0, R_1, \ldots, R_i]$ at time T, where $i, k \in Z^+$ denote the number of unique bus routes.

3.7. Train

The Train component is primarily responsible for model development. Besides, it also handles loads the latest model into the memory for performance evaluation and, based on the results, model increment. Fig. 7 shows the generic workflow used in our methodology. As discussed in Section 3.6, parameter K, whose value was estimated via pilot measurements, determines the size of the LSTM input. The encoder–decoder LSTM network consists of an encoder and a decoder. The first LSTM layer implements the encoder responsible for reading the input sequences. The output of the LSTM is a fixed-length vector that is the sequence interpreted by the model. The decoder receives the output of the encoder as input that is first reformatted to produce the same input for the two LSTM layers. We use a RepeatVector to achieve this. It produces the output of the encoder into a sequence at each period. The network then passes the data to the decoder LSTM layer. The output of the decoder layer is then passed to a TimeDistributed layer which produces the corresponding output. This is the next estimated vehicle speed on a given bus route.

Prior to submitting the data to the neural network in Fig. 7, data transformation is performed, which produces time series. This transformation is implemented through the following matrix operations. In Eqs. (7)–(9), v identifies a given path in R_{vw}, while w determines the given sampling moment. A matrix of $n \times m$ is generated for data received during teaching. Parameter n is the generated data point, and m equals the number of different bus paths. Then, the data point is expressed as

$$D^{n\times m} = \begin{bmatrix} R_{11} & \ldots & R_{1m} \\ \vdots & \ddots & \vdots \\ R_{n1} & \ldots & R_{nm} \end{bmatrix}. \tag{7}$$

To generate the input and output vectors, two additional parameters were required that determine the size of the input and output sequence. We introduce Γ for the input that specifies the number of samples included in the data prepared for teaching. Similarly, θ determines the output size, which is the matrix generated during prediction.

Let matrix I_j identify the input matrix j formulated as

$$I_j^{\Gamma \times m} = \begin{bmatrix} R_{j1} & \dots & R_{jm} \\ \vdots & \ddots & \vdots \\ R_{\Gamma 1} & \dots & R_{\Gamma m} \end{bmatrix}, \tag{8}$$

where $0 < j < (n+1) - \Gamma$ and $j \in Z^+$ is consistently met, and let matrix O_j denote the output j (which is the prediction for input I_j), formally

$$O_j^{\theta \times m} = \begin{bmatrix} R_{(j+1)1} & \dots & R_{(j+1)m} \\ \vdots & \ddots & \vdots \\ R_{\theta 1} & \dots & R_{\theta m} \end{bmatrix}. \tag{9}$$

Then, the total input vector I and the output vector O are produced as

$$I = \begin{bmatrix} I_0 \\ \vdots \\ I_{(n+1)-\Gamma} \end{bmatrix}, \quad O = \begin{bmatrix} O_0 \\ \vdots \\ O_{(n+1)-\Gamma} \end{bmatrix}. \tag{10}$$

3.8. Evaluate model

The Evaluate Model component invokes the metric evaluation for the previous and new models stored in the Model Database. The output of evaluation is their determined accuracy. The new model is stored if it yields better performance than the previous model. There could be an unlikely scenario where new models underperform, preventing model updates. We overcome such an obstacle via a threshold, which sets a higher bound to unsuccessful attempts to update the model. Nonetheless, depending on the configuration of this threshold, the new model is either stored via the Save Model component replacing the previous one or dropped via the Drop Model component.

3.9. Save model

Save Model is responsible for storing the trained models. It first serializes the new model and uploads it to the Model Database. Specifically, it replaces the last stored model with an incremental version. A mapping table is also stored in the Model Database to assign each bus route correctly. Then, the data used during training is deleted from the Temporary Database. This component also frees up the memory and erases the environment variables. Finally, once the component accomplishes all the tasks associated with model saving, the system proceeds to the next iteration.

3.10. Drop model

Suppose the Evaluate Model determines the new model is insufficient. In that case, the Drop Model component deletes the generated model and clears the memory by erasing all data loaded for training, including pertaining cached data in other components. Then the system proceeds to the next iteration.

4. Evaluation

This Section presents the evaluation results achieved by varying model training and configuration parameters. To assess the efficacy of our solution in real-world conditions, we performed our evaluation using real data obtained from the public transport service of the city Pécs. Our evaluation assessed two main aspects of the solution: the impact of epochs on the training time vs. model accuracy and congestion detection accuracy relative to actual traffic flow conditions. In what follows, we briefly introduce the dataset, preliminaries that emerged during the analysis of the dataset, and the overview of the metrics used. Then, we present the results achieved in a real-world application and challenges related to incremental learning and data collection we had to overcome.

4.1. Dataset

We examined the efficacy of our solution in the city of Pécs [72], the fifth largest city of Hungary. The estimated population of Pécs is around 148 000 with a density of 909 people/km^2, while its area covers 16277 hectares (162.8

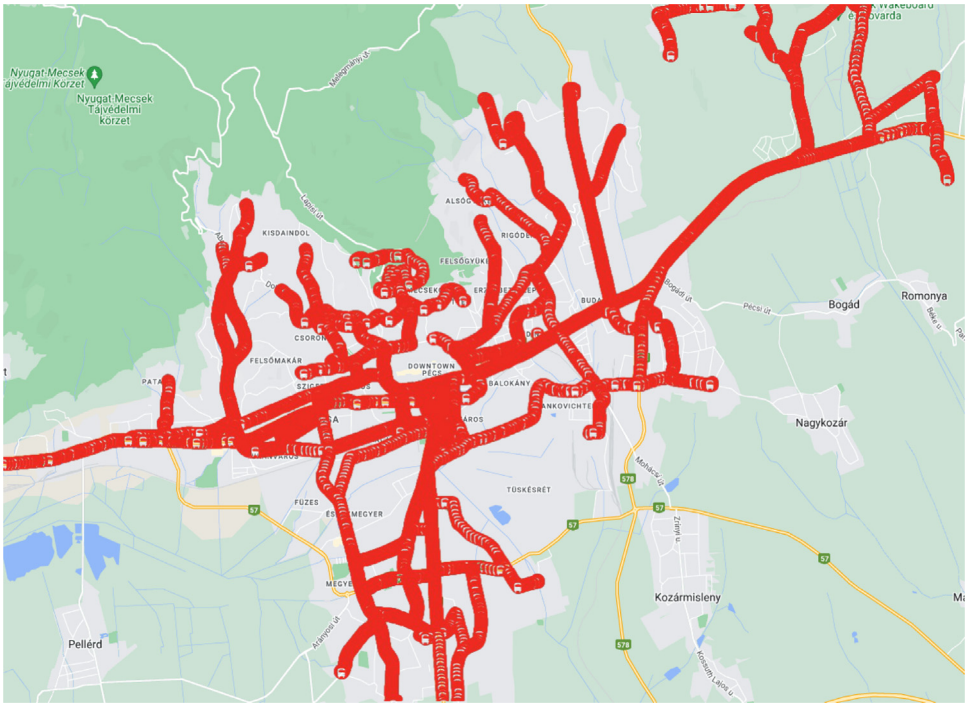


Fig. 8. Spatial coverage of the dataset relative to the urban area of Pécs.

Table 1
Sample of the bus GPS dataset.

vehicle_ID	run_ID	timestamp	GPS_H	GPS_V	Velocity [km/h]
228	11273	2020–05–05 00:00:01	65478406	165834613	21,599998
322	11242	2020–05–05 00:02:11	65643159	165840092	0
251	10800	2020–05–05 00:03:25	65634686	165834956	39,599998
322	192	2020–05–05 04:25:49	65608827	165841521	0
239	253095	2020–05–05 04:52:30	65624345	165729034	21,599998

Table 2
The structure of bus data used for evaluation.

ID	Data type	Description
vehicle_ID	integer	Unique identifier of a specific vehicle
run_ID	integer	Unique identifier of a specific route
timestamp	string	Time of observation
GPS_H	integer	Long. coordinate in 1000 angular seconds
GPS_V	integer	Lat. coordinate in 1000 angular seconds
velocity	float	Vehicle speed in km/h

km^2). Within the administrative borders of Pécs, TükeBusz Zrt. [73] provides scheduled public transportation services. At the time of evaluation, the fleet consisted of 202 buses that transported passengers on the 300 km long network. The company operates 59 daytime lines and a frequent night bus service to help citizens get around. TükeBusz Co. has recently introduced its GPS-based bus tracking system that allows real-time vehicle monitoring for location and various operational conditions, including temperature, door status, heating/cooling, and engine.

The dataset we used in our evaluation is a spatiotemporal representation of the city traffic, which sufficiently reflects the travel and mobility patterns in light, moderate, and heavy traffic conditions, covering weekdays, a public holiday, and weekends. The dataset consists of complete GPS traces of all of the buses throughout the city day-wise over a period of eight days, from August 15, 2019 to August 22, 2019. Fig. 8 shows the spatial coverage of the dataset relative to the urban area of the city. Table 1 shows a sample of the trajectories describing the bus movements, while Table 2 provides complementary information, including vehicle_ID, run_ID, timestamp, GPS_H, GPS_V, and velocity (note that additional data provided by the sensors were excluded as they were not used). These features serve as input for our congestion detection system based on incremental learning.

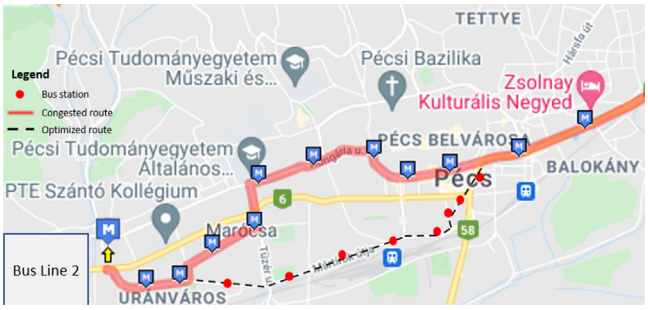


Fig. 9. Interpretative example of bus stop labels for a selected route.

4.2. Preliminaries and methodology

Our preliminary assessment has revealed that not all bus routes in the dataset had enough data for training. Such routes were typically associated with services to rural areas of the city. By excluding these routes, we concluded to use 50 different bus routes in the evaluation. We also observed significantly lower data for August 20, which is State Foundation Day, a national holiday in Hungary. As the data have not yielded sufficient configuration in our preliminary measurements and thus, no congestion detection could be performed, we transferred them to be included for the consecutive day.

It is important to note that bus stops might be interpreted by the model as locations of congestion. To appropriately handled such data points, it was necessary to identify and label the location of the stops so that the likely occurrence of false positives can be avoided. To this end, we employed the trip progress mapping - trajectory splitting technique introduced by [66]. Specifically, we first separated the GPS records by route, identifier, and date, which were mapped to bus stop sequences. Then, we split them into different trajectories where each trajectory represented a bus run from start to end stations. Finally, we also removed the idle points from these trajectories, to handle outliers better. Fig. 9 shows an interpretative example of the labeled bus stops of a selected route (2/2 A) in the dataset.

The methodology of the evaluation was as follows. We processed the data day by day. To this end, we processed the trajectories of a day followed by a prediction for the next day. Then, once we examined the accuracy of the model, we fed the system with data of the consecutive day, followed by another prediction for the next day, repeating this process iteratively. Performance evaluation took place post accomplishing all the measurements/detections.

4.3. Metrics

In what follows, we briefly overview the metrics used to assess the performance of our solution.

4.3.1. Root-mean square error

We evaluated the system using the Root-Mean Square Error (RMSE) by considering the median values for all the examined routes. The RMSE for the ith route is determined via

$$RMSE_i = \sqrt{E((\hat{\theta}_i - \theta_i)^2)}, \tag{11}$$

where θ represents the true value and $\hat{\theta}$ denotes the estimated value. Once we determined each RMSE value for the routes, we also specified the RMSE of the whole system for the given day, expressed for n routes as

$$RMSE = \frac{\sum_{i=1}^{n} RMSE_i}{n}. \tag{12}$$

Accordingly, RMSE measures the standard deviation of residuals.

4.3.2. Mean absolute error

Similar to RMSE, the Mean Absolute Error (MAE) determines the error rate of diverse bus paths, expressed for the whole system as

$$MAE = \frac{\sum_{i=1}^{n}(\frac{1}{n} \times \sum_{i=1}^{n} |\theta_i - \hat{\theta}_i|)}{n}, \tag{13}$$

where θ denotes the true value, $\hat{\theta}$ denotes the estimated value, and n indicates the number of different bus paths. In other words, MEA measures the average of the residuals in the dataset.

4.3.3. Accuracy

Accuracy is the ratio of correctly predicted observation to the total observations, formally expressed as

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}. \tag{14}$$

4.3.4. Relative congestion state index

In the absence of the parameters in Eqs. (1)–(4), we define a new congestion metric. In the congestion detection procedure, the value determined by the prediction is compared with the actual measured value at the power of e, making a difference for the bus running on the ith route as

$$C_i = 1/(1 + \exp(-\frac{\theta - \hat{\theta}_i}{\hat{\theta}_i})). \tag{15}$$

By distorting the sigmoid function, we get a state on a closed interval [0; 1] whose values can be classified into the following categories

$$RCSI_i = \begin{cases} congestion, & \text{if } C_i < \frac{1}{2} - \epsilon \\ speeding, & \text{if } C_i > \frac{1}{2} + \epsilon \\ average, & \text{else,} \end{cases} \tag{16}$$

where S_i indicates the state of the ith path, similar to that in SPI (*cf.* Eq. (2)). The *epsilon* variable in the formula is the error parameter used to shift the boundaries of each category relative to each other. By selecting this parameter, the categorization can be adapted to the predictive capabilities of the model. This creates a maximum margin distance relative to which a positive or negative displacement determines the state of the vehicle.

4.4. Results

We examined the impact of training relative to the resource capacity of the testbed (2 CPUs and 6 GB of memory). To this end, we started with a low epoch whose number we systematically increased. With 15 epochs, we observed that the resource requirements of the system and the time required for training (incremental learning) were relatively low while achieving resonable results. However, 150 epochs resulted in a relatively high duration between iterations, bringing a considerable delay into the operation of the system. Epochs over 150 did not significantly improve the performance indices. Thus, we determined epochs 15 and 150 as grounds for further evaluation. These resulted in 4 981.8 s (for 15 epochs) and 32 383.2 s (for 150 epochs), respectively, in terms of cost, giving tight bounds to the use case, especially practical applicability.

4.4.1. RMSE and MAE

As discussed in Section 4.3, the output of the network is measured alongside the RMSE and MSE metrics. Using RMSE, we determine the standard deviation of residuals, while using MAE, we specify the average of the residuals in the dataset. Fig. 10 shows the measurement results broken into six days, where the left box is formed of the values corresponding to 15 epochs, while the right box to 150 epochs. Furthermore, complementary information is provided in Table 3, specifically the minimum, maximum, mean, standard deviation, variance, and median for each epoch configuration, providing additional insights.

From the RMSE results, we find that with 15 epochs, the line of best fit can be drawn at approximately the same distance in each incremental step, suggesting insufficient learning based on future data points, coming short of prediction improvement capability. This is also evident from Table 3, where the mean metric remains in around the same range of magnitude, from 30.13 on day one up to 29.65 units on day seven. It is noteworthy that the intermediate variability was negligible for each day, ranging from [0.02; 0.57] in terms of the mean metric. Nevertheless, when examining the evolution of the model through the lens of standard deviation, with a nearly identical average RMSE value, it decreased from the initial 3.81 to 3.49, while the variance improved by 19.59%.

In light of the above mentioned, we can conclude that the model has improved and become more accurate, as the actual standard speed of the vehicle can be predicted better due to the smaller standard deviation. However, this significant improvement has manifested in a substantial drop in the number of epochs. The initial average (which was already lower than 15 epochs) decreased from 27.26 to 8.48. This improvement can be observed along a well-described curve in the arc of Fig. 10. Thus, despite the improvement in the mean, variance has decreased (from 13.51 to 11.20), while the standard deviation for 'Day 7' has not exceeded 3.35. As such, the standard deviation is still high enough (quantified for the 150 epoch, the improvement over the average is 221.46%) to lead to errors when predicting in residential areas (where the maximum permitted speed is typically 20 km/h in Pécs). The observed standard deviation for the 'Day 7' iteration is due to the very high maximum, which is 15.49, and the minimum is 1.49. There is a significant difference between the two, also comparable to the average. That is, the difference is 14.00 in this case, while the average is 8.48.

It is worth looking at the MAE values to accurately determine the actual km/h difference between the individual cases. We found that despite the improvement in RMSE, the km/h difference between the prediction and the actual value

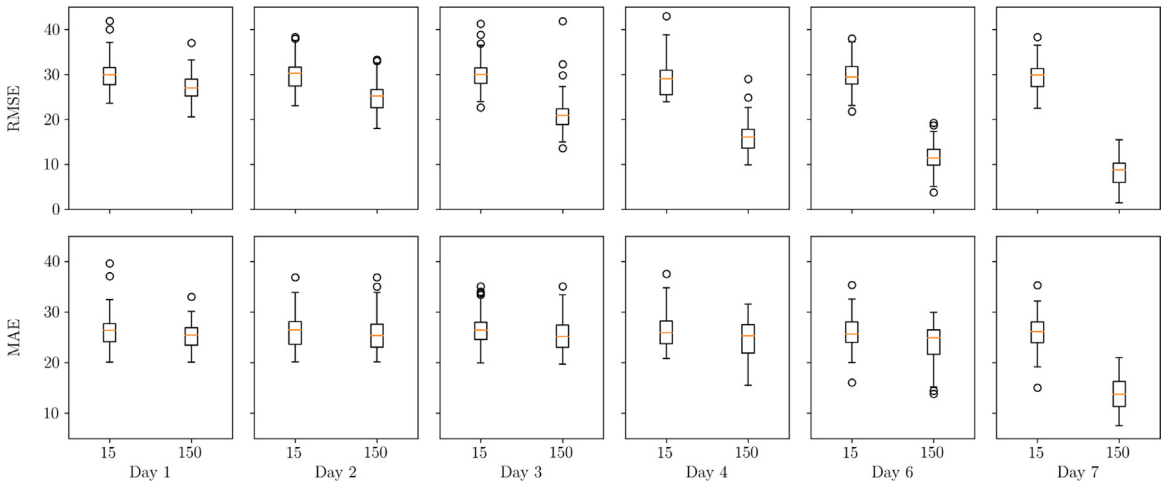


Fig. 10. Measurement results alongside the RMSE and MSE metrics broken into six days, where the left box is formed of the values corresponding to 15 epochs, while the right box to 150 epochs.

Table 3
Quantitative expression of the minimum, maximum, mean, standard deviation, variance, and median for each epoch configuration, complementing Fig. 10.

Epoch		Day 1		Day 2		Day 3		Day 4		Day 6		Day 7	
		15	150	15	150	15	150	15	150	15	150	15	150
Minimum	RMSE	23.60	20.60	23.03	18.03	22.63	13.63	30.09	9.91	21.81	3.81	22.49	1.49
	MAE	20.07	20.01	20.13	20.14	19.91	19.71	20.83	15.53	16.04	13.84	15.05	7.48
Maximum	RMSE	41.89	39.61	38.32	33.30	41.29	41.86	30.09	29.00	38.03	19.25	38.34	15.49
	MAE	39.61	33.03	36.86	36.82	35.10	35.09	37.54	31.51	35.38	29.94	35.33	21.00
Average	RMSE	30.13	27.26	30.19	25.13	30.22	21.29	30.09	16.13	29.71	11.34	29.65	8.48
	MAE	26.43	25.35	26.45	26.08	26.47	25.56	26.54	24.85	25.78	23.66	25.86	14.24
StandardDeviation	RMSE	3.81	3.67	3.85	3.82	3.87	4.79	0.00	3.98	3.57	3.30	3.49	3.35
	MAE	3.67	2.68	3.67	4.03	3.52	3.50	3.88	4.06	3.49	4.17	3.7	3.64
Variance	RMSE	14.53	13.51	14.85	14.61	14.94	22.91	0.00	15.84	12.73	10.91	12.15	11.21
	MAE	13.45	7.16	13.46	16.24	12.46	12.42	15.02	16.47	12.19	17.40	13.67	13.26
Median	RMSE	29.95	27.00	30.28	25.25	30.00	20.90	30.09	16.01	29.48	11.41	29.87	8.82
	MAE	26.38	25.45	26.47	25.35	26.41	25.20	25.91	25.28	25.63	24.89	26.14	13.76

averages around 26.43 km/h for the 15 epochs for each day with a standard deviation of 3.50. However, for the 150 epoch cycles, the MAE values decrease according to the curve observed in the RMSE (the curve descends more slowly than the RMSE). Ultimately, in the 'Day 7' iteration, the prediction reaches a state where we could only predict with a difference of 7.48 km/h from the actual value in the minimum case. Furthermore, on average, the difference was 14.24 km/h, implying that such a magnitude will not be perceived during prediction, as the condition of the vehicle falls into the appropriate category correctly due to the categorization and margin of error. We also find that the median of the 15 and 150 epochs moves together starting from the 'Day 1' iteration (where the median of 15 epochs was 26.38, while the median of 150 epochs was 25.45) up to 'Day 6' iteration. Then, a significant improvement of 89.97% can be observed for 'Day 7' in the case of 150 epochs (the median value for 15 epochs is 26.14 and 13.76 for 150 epochs). We can conclude that for all days, RMSE $>$ MAE (in terms of their median) for both short- and long-term training, implying variation in the errors. Individual median values are typically 5 to 10 units lower among the MAE results. However, this measured value is not significant enough to indicate the presence of substantial errors.

Based on the measurement results, we conclude that the model could sufficiently learn for each learning iteration with both 15 and 150 epoch parameter configurations. In terms of the mean, this learning constitutes a 1.62% improvement for 15 epochs, while 221.46% for 150 epochs. Undoubtedly, the configuration with 15 epochs yielded a lower improvement compared to that with 150 epochs. This suggests that an epoch number converging as much to 150 epochs as possible should be the ultimate goal to produce the best results. To this end, the resource capacity and hardware configuration of the runtime environment play a vital role as they seem to substantiate the online operation of incremental learning.

4.4.2. Accuracy

The congestion detection diagram using the resulting model is shown in Fig. 11. The blue line indicates the speed of each bus approaching the specified route. These bus velocities expressed in km/h were first used to train a model. The

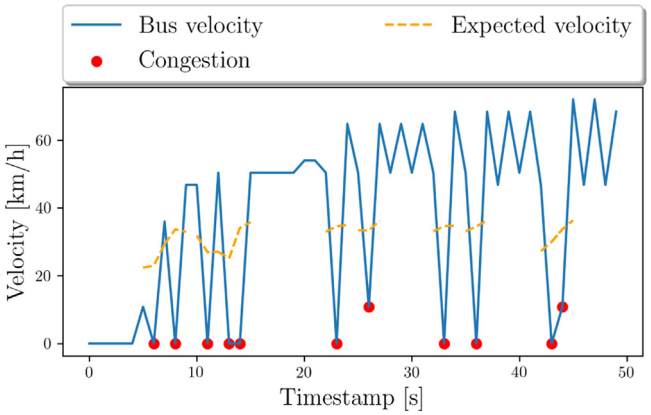


Fig. 11. The evolution of prediction accuracy of bus congestion over time relative to velocity.

model was then used to predict the speed at which the vehicle was expected to travel for the given period (the period was determined by the volatility of the speed, as discussed in Section 3.7). Next, knowing the prediction and $RSCI_i$, the margin of ϵ — that is, the traffic congestion — was determined using Eq. (16). In Fig. 11, the red dots indicate these congestion detection locations derived from the percentage difference between the predicted speed and the actual speed, obtained via Eq. (15). Lastly, the expected speed and direction of acceleration/deceleration are denoted in yellow dashed lines. Notably, the yellow line indicates the curvature to which if we were to fit the error with a maximum margin of ϵ, that would be the range within which we predict the 'average' *RSCI* state. Exceeding this radius towards the positive y-axis would yield 'speeding,' otherwise, it would indicate congestion. It is noteworthy that time synchronization is also crucial. To this end, the sampling moment must match the training time of a given day, which we achieved by GPS trajectory mapping.

We examined the efficacy of the better-performing model in terms of accuracy (*i.e.*, the one obtained with 150 epochs) using the condition test formula introduced in Section 4.3.4. In doing so, we set the error rate to 0.15 by configuring the ϵ parameter. The congestion indicator in our evaluation was satisfied via $RSCI_i$ = congestion if $RSCI_i \in [0; \frac{1}{2} - \epsilon(:= 0.15)]$.

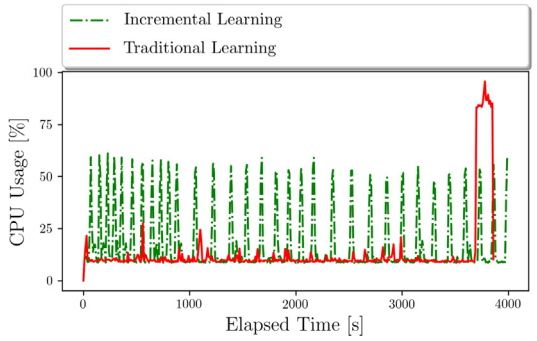
From Fig. 11, we find that the model expected a linear acceleration/deceleration. The accuracy of the system for predicting congestion points was 82.37% (we obtained 68.11% for the 15 epoch configuration, which is significantly lower). This represents a considerable precision with room for improvement and optimization. At this stage, further optimization can be achieved in several ways, such as fine-tuning the ϵ parameter and systematically adjusting the number of epochs and hyperparameters of the network layers.

4.4.3. Utilization comparison

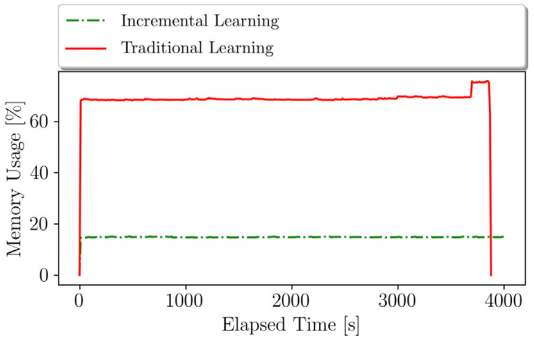
We also examined the CPU, memory, and energy consumption of our incremental learning approach over a period of time compared to a hypothetical traditional learning solution. To this end, we adjusted our incremental learning prototype to process the entire dataset at once, commonly employed across existing traditional learning solutions [65,67–69]. To achieve more realistic results for the comparison, we kept the architecture of the LSTM autoencoder with its encoder and decoder intact, as introduced in Fig. 7. Furthermore, we performed the comparison using the hardware configuration discussed earlier in Section 4.4. In what follows, we present the achieved results.

Fig. 12 shows the CPU and memory utilization relative to time of our incremental solution (denoted with a green dash-dotted line) versus a traditional learning approach (denoted with a red line). In both Figs. 12(a) and 12(b), the x-axis shows the sampling expressed in seconds, while the y-axis shows the percentage of the used resources. Our incremental learning approach loaded the data in 1000 batches to train the models periodically. In contrast, for the traditional learning approach, the dataset for the entire week under study was loaded at once for model training. Consequently, the incremental learning approach took longer to finish than traditional learning. Specifically, the total run time for incremental learning was 3 h and 16 min, as opposed to the traditional learning approach, which finished in 1 h and 4 min. Note that Fig. 12 shows only the period that provides comparable results, while the rest of the incremental learning approach being omitted for better visualization and comprehension. Nonetheless, the omitted part of incremental learning showed a similar pattern to that recorded in the range of [0; 4000].

From 12(a), we find that the traditional learning approach was more CPU-intensive when training using the entire dataset between time steps 3764 and 3849, with the utilization peaking at 95.7%. In contrast, the maximum CPU utilization of incremental learning was lower, specifically 63.3%, though approaching this value repeatedly throughout the evaluation. The pattern for incremental learning results from a learning data batch size of 1000 iteratively repeated, causing the spikes in the graph. The slight variations in the volatilities of the curve are inherently caused by factors such as the CPU temperature and the additional load on the processor generated by background processes and schedulers of the



(a) CPU consumption of the processes.



(b) Memory consumption of the processes.

Fig. 12. CPU and memory utilization of incremental learning vs. traditional learning.

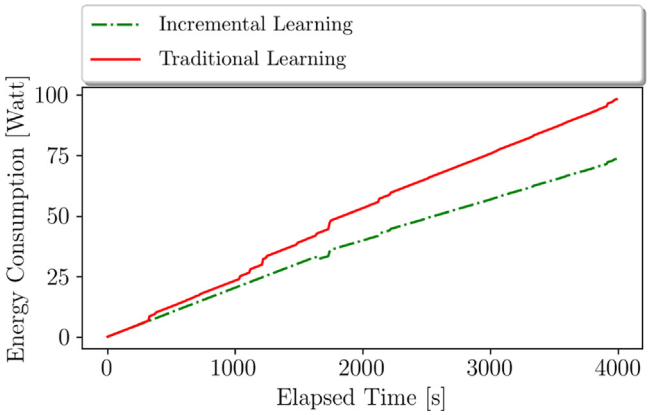


Fig. 13. Energy consumption of the processes.

operating system. Increasing the trajectory data batch size by additional 1000 samples would certainly increase CPU load as learning on a more extensive dataset causes more resources to be consumed. In contrast, decreasing the batch size would undoubtedly lead to lower CPU utilization. Considering the observed behavior, additional CPU load optimization can be achieved by distributed learning [74], alleviating the CPU from being under a constant high load. It is also worth mentioning that, in contrast to traditional learning, our solution is ready to make predictions sooner. This is due to the batch size of 1000 samples vs. the entire dataset required to be processed before prediction can start. This is the cause of the lower CPU utilization of the traditional learning approach ranging between 12.5% and 27.5% denoted with a red line observable in Fig. 12(a) since data reading is a memory-intensive process, which loads the CPU less.

Fig. 12(b) shows the memory occupancy by the two approaches with the green dash-dotted line denoting our incremental approach and the red line denoting the traditional learning approach. From Fig. 12(b), we find that the maximum memory usage during incremental learning was 15.03%. In contrast, the traditional learning approach consumed up to 71.23% of the allocated memory. Upon extracting the learned model, an additional jump in the consumption can be observed, with a memory peak of 75.71%, caused by writing the model to the underlying persistent storage. In the case of incremental learning, such additional memory overhead is not observable upon extracting the model as it is loaded to the memory at the first iteration, with the space reserved allocated for this model throughout the entire incremental learning process. In summary, the traditional learning approach with such high data volatility occupied 503.72% more memory than the incremental approach, where the previous unit of information was deleted immediately after processing. Thus, incremental learning showed superior performance in terms of memory utilization.

At last, Fig. 13 shows the energy consumption of the two approaches expressed in Watts measured using the CodeCarbon [75] Python package. From Fig. 13, we find a broadening tendency of the two lines due to the higher CPU and memory load of traditional learning (denoted with a red line). Consequently, our incremental learning approach (denoted with a green dash-dotted line) required less energy, with its energy efficacy becoming only more distinctive over time. At time step 4000, we find that incremental learning consumed 24.99% fewer Watts than traditional learning. However, it is important to note that incremental learning took longer to accomplish (running for a total of 11 785 s omitted in Fig. 13). Thus, its overall energy consumption eventually surpassed the requirements of traditional learning by 122.43%.

Nevertheless, further studies are needed to assess the long-term environmental impact of using incremental learning over traditional learning, which was out of the scope of this present research. For example, the energy consumption assessment of our evaluation did not consider additional critical factors impacting the results, such as data collection and delivery resource demands.

Considering the observed results, our incremental learning approach outperformed traditional learning in terms of CPU, memory, and energy utilization for the period under study. There is certainly a trade-off between higher energy consumption for a lower time period of traditional learning vs. lower energy consumption for a longer time period of incremental learning. In fact, the same applies to the CPU and memory utilization of the two approaches. On the one hand, one could train with traditional learning for a longer period but with higher CPU and memory demands. On the other hand, one could train with incremental learning for a lower period but under a constant and lower CPU and memory requirement. The latter is particularly beneficial if there is no dedicated hardware for the machine responsible for the learning process, with other processes competing for the same resources. Furthermore, it is also advantageous for constrained IoT devices, including intelligent transportation system components. With training being moved from the cloud to the edge commonly seen across recent advances in IoT solutions [76,77], low resource consumption capabilities becomes crucial.

5. Challenges and considerations for scalable traffic management

In today's rapidly evolving data landscape, machine learning systems must be scalable, adaptable, and able to handle vast amounts of data. This section will explore some of the challenges that arise in developing such systems, and the considerations that must be taken into account to ensure their success.

5.1. Scalability

The scalability of our solution is intended to address the diversity and volume of data. To this end, the architectural design discussed in Section 3 has identified two different scalability issues. From Fig. 7, we find that the bus paths are represented as a heterogeneous two-dimensional matrix along vertical and horizontal directions. We cannot impose homogeneity in either direction since that would force the introduction of constraints compromising the scalability, fast implementation, and applicability of the solution.

Vertical scalability of our solution refers to the system's capability to adapt to the occurrence of new paths in the matrix structure. In other words, when a new bus path is observed, an additive element is created alongside the existing ones, which will affect the weight values of neurons not yet trained by the model. This was demonstrated earlier in Section 3.6, where the matrix object is populated with "NaN" values as a function of the maximum dimension number. The consequence of this kind of scalability is that model weights become updateable without directly changing the model itself (no new model creation required), and thus, losing the hyperparameters of the previous model can be avoided.

By horizontal scalability, we refer to our quantitative problem-oriented approach to data. If we were to explicitly specify that no learning can begin unless a constant numerical value of trajectory data is observed for each path, model training would be delayed indefinitely, eventually losing the scalability of the data input from a horizontal viewpoint. In contrast, our solution operates with no such constraint as it adapts to the variable length of data points per bus paths. Consequently, when the system has available resources (that is no active learning cycle running and the previous model has been saved), learning can start immediately without entering an idle state. Due to the matrix operations, a minimum constraint must certainly be definitive on the horizontal data information. However, that can be derived from the number of bus paths.

In conclusion, scalability in our work does not refer to the traditional sense of the word, *i.e.*, scaling up/down by adding/removing nodes/resources typical for systems such as cloud computing. We instead wish to express a state or ability of machine learning models to expand in tandem with streaming data for model evolution and faster convergence, simply expressed as 'its ability to scale'. Traditional ML-based approaches discussed in Section 2.4, especially deep learning solutions, typically fix the structure of the model network in advance and cannot be changed during the training process. Determining certain parameters, such as depth, can be challenging when working with streaming data since the full training set is not available at the start of the learning task. This poses a significant challenge for traditional learning methods. However, in our incremental setting, different depths can be applied to different numbers of instances, making it a more scalable approach. This allows for greater flexibility in adapting to the available data and can lead to improved performance.

5.2. Challenges in incremental learning

Despite IL has been shown practicable, catastrophic forgetting [78] and concept drifts [79] remains long-standing challenges. Suppose a model h_{base} is trained using n classes. If this model is updated with m new classes to form h_{new}, the new model is expected to perform reliably on the set of $n + m$ classes. However, in practice, this is seldom the case since the performance of predicting old classes n typically drops radically because of the lack of old classes when training with new data [37], yielding the phenomenon of catastrophic forgetting. In this work, we use NaN filling to address this

challenge (*cf.* Section 3.6). Consequently, the weights for the bus routes that have not observed trajectory data are kept intact, preventing catastrophic forgetting.

In the real world, data distribution of learned classes (h_{base}) is likely to change [80], causing concept drifts. Concept in classification is defined as the joint distribution $P(X, Y)$, where X is the input data and Y denotes the target variable [81]. Suppose a model is trained on data streams by time t with joint distribution $P(X_t, Y_t)$ and let $P(X_n, Y_n)$ representing the joint distribution of old classes in future data streams [37]. Concept drift occurs when $P(X_t, Y_t) \neq P(X_n, Y_n)$. Concept drift inevitably leads to performance degradation. In this work, we maximize the size of the intersection to mitigate this problem such that depending on the different paths, if there are new ones in X_n, but X_t has a common intersection (the two are not disjunct and $|X_n| > |X_t|$), the input size is updated by the number of new paths in X_n. If $|X_n| < |X_t|$, we append NaN records at the appropriate position in the array so that the two become equivalent in size. If $|X_n| \equiv |X_t|$, no action is required.

Processing data and detecting congestion with low duration is highly preferred. However, since Pécs is a relatively small city, iterations with lower periods did not yield sufficient data for reliable congestion detection. In this trade-off, we prioritized evaluating our solution in real-world operational conditions over an assessment that, despite providing higher granularity, also contains more synthetic data. Furthermore, there is also a lack of a publicly accessible labeled dataset (i.e., a ground truth) that could be used to evaluate our congestion detection technique. To the best of our knowledge, there is no publicly available dataset with reliable labels indicating congested periods of traffic flows.

With granularity, additional challenges might arise. Suppose the bus data points coming from the OBU are not frequent enough. In that case, the system might abstract the driving style of specific drivers, yielding a problem as ϵ error might not be enough to compensate for the aggressiveness of the learned data. This can consequently lead to an accuracy drop in predictions. Ideally, different driving styles can be blended to produce a set of training data that can adequately represent various driving styles.

5.3. Challenges in data collection

Collecting data for intelligent transport systems and urban traffic management can present significant challenges in terms of legal, technical, and technological aspects.

One of the major legal challenges is compliance with data protection regulations, such as the General Data Protection Regulation (GDPR) in the European Union. This regulation sets strict guidelines for the collection, storage, and processing of personal data. The use of cameras and other sensors to collect traffic data can be considered as personal data, and as such, must comply with GDPR. This can add complexity to the data collection process and may require obtaining explicit consent from individuals or using anonymization techniques to protect privacy.

Another legal challenge is related to ownership and access to data. In urban areas, data may be collected by different public and private entities, such as city authorities, transportation companies, and private sensor owners. Obtaining access to all the sensor data can be bureaucratic and complex. Moreover, data access agreements and data-sharing policies must be established between these entities to ensure data interoperability and data quality.

From a technical perspective, data collection can be challenging due to the heterogeneous nature of the data sources. Data may be collected from various sensors, such as cameras, GPS, and accelerometers, which may have different data formats and sampling rates. Integrating these data sources and normalizing the data to a common format can be a significant challenge. In addition, the reliability and quality of the data may be an issue, as sensors can be prone to failure or malfunctions. The calibration of sensors and maintenance of data collection infrastructure can also be a challenge, as it requires significant resources and expertise.

In addition, it is often observed that only a small fraction of the hundreds or thousands of sensors implemented, such as loop detectors and OBUs, are actually functional. This limited number of operating sensors is typically due to insufficient funding for maintenance and upgrades of the measurement equipment. As a result, researchers and professionals are often forced to rely on artificial and simulated data to conduct their studies, as obtaining real-world, high-resolution data with labels is a significant challenge.

Lastly, from a technological perspective, there are challenges related to the processing and analysis of the collected data. The sheer volume of data generated by sensors and other sources can be overwhelming, and processing and analyzing this data in real-time requires specialized computing infrastructure and algorithms.

5.4. Applicability

Our method for urban traffic bus congestion detection and prediction was evaluated using real-world data obtained from the city of Pécs. Although Pécs is a smaller city, we believe that our approach has the potential to be adapted for use in cities of different sizes, such as Budapest. However, the practical implementation of our solution may present greater challenges in larger cities that must be considered. The legal, technical, and technological complexities discussed in Section 5.3 can be certainly more significant in larger cities. From our experience, legal challenges are the most significant obstacle to the application of traffic management systems. While data collection in larger cities may require more careful attention to ethical and privacy considerations due to the potentially higher heterogeneity of the data, and processing a massive volume of measurement data may demand more significant resource capacities, accessing data from different

sensors that fall under the jurisdiction of different legal entities can create bureaucratic hurdles that can hinder the application of a traffic management system the most.

Despite these obstacles, we believe that our method can still be applied to larger cities with careful planning and collaboration with relevant stakeholders. The use of incremental learning, coupled with stream processing, makes our solution more adaptable and scalable for application in larger cities with diverse and complex traffic conditions, enabling real-time analysis of bus congestion and prediction. However, it is important to carefully consider the legal, technical, and technological aspects of data usage, including obtaining informed consent and ensuring data privacy.

Furthermore, our proposed method has diverse applicability beyond congestion detection in public transport systems, including traffic flow prediction and optimization. By leveraging the long-term temporal dependency captured by our LSTM model, our solution can provide a more reliable basis for traffic management decision-making. In conclusion, with careful planning and collaboration, our solution can be adapted to different contexts and contribute to improving public transport services.

6. Conclusion

Traditional machine learning approaches run off-line while assuming all training data are available from the beginning. However, the world is non-stationary, and the semantics and the distribution of the observed data keep changing. The inability to run on realistic timescales can yield models that perform poorly in non-stationary scenarios.

This study confirmed that the designed incremental learning methodology can overcome this problem and is sufficient for modeling congestion detection using trajectory data. Our evaluation showed that sufficient detection efficacy, 82.37%, can be achieved in a relatively short time. We also showed that the average RMSE metric improves exponentially with an increasing epoch number, thereby further increasing the accuracy of congestion detection. This improvement quantitatively accounts for 1.62% in the case of the 15 epoch and 221.46% in the case of the 150 epoch configuration, implying the capability of our solution to evolve over time. Regarding resource consumption in a real-world application, we observed a trade-off between higher energy consumption for a lower time period of traditional learning vs. lower energy consumption for a longer time period of incremental learning. However, our approach still promises better efficacy than traditional learning, especially in constrained environments, like in systems with the logic being moved to the edge.

The advantage of the presented approach is its ability to work with real-time data. As the evaluation demonstrated, our solution is ready to make predictions sooner than traditional learning-based approaches. Additionally, our solution's scalability lies in its capability to adapt to both the occurrence of new bus paths and the variable length of data points per bus path. It also reduces the computational burden, and in doing so, it avoids re-training the models from scratch when new data are available. Furthermore, although in this work we explored the efficacy of incremental learning in the domain of public transportation optimization, our solution is universal, and so it can be potentially deployed in other disciplines, such as computer network traffic engineering and anomaly detection.

We plan to extend the solution to publicly available labeled taxi trajectory data in future work. We also aim to measure a more extended time interval to obtain more insights into the operation of the system. Examining other models in the presented architectural framework, including CNN-LSTM-based model, bidirectional-LSTM, CNN-bidirectional-LSTM, also belong among our future goals.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgments

This work was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. Supported by the ÚNKP-22-5-BME-320 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund. The research reported in this paper is part of project no. BME-NVA-02, implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021 funding scheme. The research reported in this paper is part of project no. 2018-1.3.1-VKE-2018-00021, supported by the National Research, Development and Innovation Fund.

References

[1] E. Avdeeva, T. Averina, L. Kochetova, Life quality and living standards in big cities under conditions of high-rise construction development, E3S Web Conf. 33 (2018) 03013, http://dx.doi.org/10.1051/e3sconf/20183303013.

[2] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, C. Chen, Data-driven intelligent transportation systems: A survey, IEEE Trans. Intell. Transp. Syst. 12 (4) (2011) 1624–1639, http://dx.doi.org/10.1109/TITS.2011.2158001.

[3] Q. Yang, Z. Yue, R. Chen, J. Zhang, X. Hu, Y. Zhou, Real-time detection of traffic congestion based on trajectory data, J. Eng. 2019 (11) (2019) 8251–8256, http://dx.doi.org/10.1049/joe.2019.0872.

[4] M. Barth, K. Boriboonsomsin, Real-world carbon dioxide impacts of traffic congestion, Transp. Res. Rec. 2058 (1) (2008) 163–171, http://dx.doi.org/10.3141/2058-20.

[5] Y. El-Hansali, S. Farrag, A. Yasar, H. Malik, E. Shakshuki, K. Al-Abri, Assessment of the traffic enforcement strategies impact on emission reduction and air quality, Procedia Comput. Sci. (ISSN: 1877-0509) 184 (2021) 549–556, http://dx.doi.org/10.1016/j.procs.2021.03.068.

[6] L. Liang, P. Gong, Urban and air pollution: a multi-city study of long-term effects of urban landscape patterns on air quality trends, Sci. Rep. (ISSN: 2045-2322) 10 (1) (2020) 18618, http://dx.doi.org/10.1038/s41598-020-74524-9.

[7] A.C. Sutandi, ITS impact on traffic congestion and environmental quality in large cities in developing countries, in: Proceedings of the Eastern Asia Society for Transportation Studies, vol. 2009, 2009, pp. 180–180, http://dx.doi.org/10.11175/eastpro.2009.0.180.0.

[8] T. Bandaragoda, D. De Silva, D. Kleyko, E. Osipov, U. Wiklund, D. Alahakoon, Trajectory clustering of road traffic in urban environments using incremental machine learning in combination with hyperdimensional computing, in: 2019 IEEE Intelligent Transportation Systems Conference, ITSC, 2019, pp. 1664–1670, http://dx.doi.org/10.1109/ITSC.2019.8917320.

[9] X. Wang, Y. Zhu, S. Han, L. Yang, H. Gu, F.-Y. Wang, Fast and progressive misbehavior detection in internet of vehicles based on broad learning and incremental learning systems, IEEE Internet Things J. 9 (6) (2022) 4788–4798, http://dx.doi.org/10.1109/JIOT.2021.3109276.

[10] X. Wang, Y. Zhao, F. Pourpanah, Recent advances in deep learning, Int. J. Mach. Learn. Cybern. 11 (4) (2020) 747–750, http://dx.doi.org/10.1007/s13042-020-01096-5,

[11] A. Kammoun, R. Slama, H. Tabia, T. Ouni, M. Abid, Generative adversarial networks for face generation: A survey, ACM Comput. Surv. (ISSN: 0360-0300) (2022) http://dx.doi.org/10.1145/1122445.1122456.

[12] Y. Nawal, M. Oussalah, B. Fergani, A. Fleury, New incremental SVM algorithms for human activity recognition in smart homes, J. Ambient Intell. Humaniz. Comput. (2022) http://dx.doi.org/10.1007/s12652-022-03798-w,

[13] Z. Chen, B. Liu, Lifelong machine learning, Synth. Lect. Artif. Intell. Mach. Learn. 12 (3) (2018) XIX, 187, http://dx.doi.org/10.1007/978-3-031-01581-6.

[14] S.S. Sarwar, A. Ankit, K. Roy, Incremental learning in deep convolutional neural networks using partial network sharing, IEEE Access 8 (2020) 4615–4628, http://dx.doi.org/10.1109/ACCESS.2019.2963056.

[15] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Y. Fu, Large scale incremental learning, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, IEEE Computer Society, Los Alamitos, CA, USA, 2019, pp. 374–382, http://dx.doi.org/10.1109/CVPR.2019.00046.

[16] Y. Luo, L. Yin, W. Bai, K. Mao, An appraisal of incremental learning methods, Entropy (ISSN: 1099-4300) 22 (11) (2020) http://dx.doi.org/10.3390/e22111190.

[17] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, A. Mouzakitis, A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications, IEEE Internet Things J. 5 (2) (2018) 829–846, http://dx.doi.org/10.1109/JIOT.2018.2812300.

[18] Z. Xiao, F. Li, R. Wu, H. Jiang, Y. Hu, J. Ren, C. Cai, A. Iyengar, TrajData: On vehicle trajectory collection with commodity plug-and-play OBU devices, IEEE Internet Things J. 7 (9) (2020) 9066–9079, http://dx.doi.org/10.1109/JIOT.2020.3001566.

[19] L. Li, R. Jiang, Z. He, X.M. Chen, X. Zhou, Trajectory data-based traffic flow studies: A revisit, Transp. Res. C (ISSN: 0968-090X) 114 (2020) 225–240, http://dx.doi.org/10.1016/j.trc.2020.02.016.

[20] J. Fan, C. Fu, K. Stewart, L. Zhang, Using big GPS trajectory data analytics for vehicle miles traveled estimation, Transp. Res. C (ISSN: 0968-090X) 103 (2019) 298–307, http://dx.doi.org/10.1016/j.trc.2019.04.019.

[21] Q. Xue, K. Wang, J.J. Lu, Y. Liu, Rapid driving style recognition in car-following using machine learning and vehicle trajectory data, J. Adv. Transp. (ISSN: 0197-6729) 2019 (2019) 9085238, http://dx.doi.org/10.1155/2019/9085238.

[22] Y. Miao, X. Tao, X. Xu, J. Lu, Joint 3-D shape estimation and landmark localization from monocular cameras of intelligent vehicles, IEEE Internet Things J. 6 (1) (2019) 15–25, http://dx.doi.org/10.1109/JIOT.2018.2872435.

[23] S. Djahel, R. Doolan, G.-M. Muntean, J. Murphy, A communications-oriented perspective on traffic management systems for smart cities: Challenges and innovative approaches, IEEE Commun. Surv. Tutor. 17 (1) (2015) 125–151, http://dx.doi.org/10.1109/COMST.2014.2339817.

[24] N.D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, A.T. Campbell, A survey of mobile phone sensing, IEEE Commun. Mag. 48 (9) (2010) 140–150, http://dx.doi.org/10.1109/MCOM.2010.5560598.

[25] S. Abdelhamid, H.S. Hassanein, G. Takahara, Vehicle as a mobile sensor, Procedia Comput. Sci. (ISSN: 1877-0509) 34 (2014) 286–295, http://dx.doi.org/10.1016/j.procs.2014.07.025.

[26] J. Guerrero-Ibáñez, S. Zeadally, J. Contreras-Castillo, Sensor technologies for intelligent transportation systems, Sensors (ISSN: 1424-8220) 18 (4) (2018) http://dx.doi.org/10.3390/s18041212.

[27] J.C. Falcocchio, H.S. Levinson, Measuring traffic congestion, in: Road Traffic Congestion: A Concise Guide, Springer International Publishing, Cham, ISBN: 978-3-319-15165-6, 2015, pp. 93–110, http://dx.doi.org/10.1007/978-3-319-15165-6_8.

[28] T. Afrin, N. Yodo, A survey of road traffic congestion measures towards a sustainable and resilient transportation system, Sustainability (ISSN: 2071-1050) 12 (11) (2020) http://dx.doi.org/10.3390/su12114660.

[29] A.M. Nagy, V. Simon, Improving traffic prediction using congestion propagation patterns in smart cities, Adv. Eng. Inform. (ISSN: 1474-0346) 50 (2021) 101343, http://dx.doi.org/10.1016/j.aei.2021.101343.

[30] A.M. Nagy, V. Simon, A novel congestion propagation modeling algorithm for smart cities, Pervasive Mob. Comput. (ISSN: 1574-1192) 73 (2021) 101387, http://dx.doi.org/10.1016/j.pmcj.2021.101387.

[31] F. He, X. Yan, Y. Liu, L. Ma, A traffic congestion assessment method for urban road networks based on speed performance index, Procedia Eng. (ISSN: 1877-7058) 137 (2016) 425–433, http://dx.doi.org/10.1016/j.proeng.2016.01.277.

[32] L. Tišljarić, T. Carić, B. Abramović, T. Fratrović, Traffic state estimation and classification on citywide scale using speed transition matrices, Sustainability (ISSN: 2071-1050) 12 (18) (2020) http://dx.doi.org/10.3390/su12187278.

[33] A. Nagy, V. Simon, Traffic congestion propagation identification method in smart cities, Infocommun. J. 13 (2021) 45–57, http://dx.doi.org/10.36244/ICJ.2021.1.6.

[34] C. Wan, Z. Yang, D. Zhang, X. Yan, S. Fan, Resilience in transportation systems: a systematic review and future directions, Transp. Rev. 38 (4) (2018) 479–498, http://dx.doi.org/10.1080/01441647.2017.1383532.

[35] J. Tang, H.R. Heinimann, A resilience-oriented approach for quantitatively assessing recurrent spatial-temporal congestion on urban roads, PLoS One 13 (1) (2018) 1–22, http://dx.doi.org/10.1371/journal.pone.0190616.

[36] V. Losing, B. Hammer, H. Wersing, Incremental on-line learning: A review and comparison of state of the art algorithms, Neurocomputing 275 (2018) 1261–1274, http://dx.doi.org/10.1016/j.neucom.2017.06.084.
[37] J. He, R. Mao, Z. Shao, F. Zhu, Incremental learning in online scenario, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, IEEE Computer Society, Los Alamitos, CA, USA, 2020, pp. 13923–13932, http://dx.doi.org/10.1109/CVPR42600.2020.01394.
[38] G.I. Parisi, R. Kemker, J.L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: A review, Neural Netw. (ISSN: 0893-6080) 113 (2019) 54–71, http://dx.doi.org/10.1016/j.neunet.2019.01.012.
[39] M. Sayed-Mouchaweh, Learning in dynamic environments, in: Learning from Data Streams in Dynamic Environments, Springer International Publishing, Cham, ISBN: 978-3-319-25667-2, 2016, pp. 11–32, http://dx.doi.org/10.1007/978-3-319-25667-2_2.
[40] D. Nallaperuma, R. Nawaratne, T. Bandaragoda, A. Adikari, S. Nguyen, T. Kempitiya, D. De Silva, D. Alahakoon, D. Pothuhera, Online incremental machine learning platform for big data-driven smart traffic management, IEEE Trans. Intell. Transp. Syst. (ISSN: 1558-0016) 20 (12) (2019) 4679–4690, http://dx.doi.org/10.1109/TITS.2019.2924883.
[41] M.A. Maloof, R.S. Michalski, Incremental learning with partial instance memory, Artificial Intelligence (ISSN: 0004-3702) 154 (1) (2004) 95–126, http://dx.doi.org/10.1016/j.artint.2003.04.001.
[42] D. Kleyko, R. Hostettler, W. Birk, E. Osipov, Comparison of machine learning techniques for vehicle classification using road side sensors, in: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, 2015, pp. 572–577, http://dx.doi.org/10.1109/ITSC.2015.100.
[43] D. Kleyko, R. Hostettler, N. Lyamin, W. Birk, U. Wiklund, E. Osipov, Vehicle classification using road side sensors and feature-free data smashing approach, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems, ITSC, 2016, pp. 1988–1993, http://dx.doi.org/10.1109/ITSC.2016.7795877.
[44] Y. Qi, S. Ishak, A hidden Markov model for short term prediction of traffic conditions on freeways, Transp. Res. C 43 (2014) http://dx.doi.org/10.1016/j.trc.2014.02.007.
[45] Y. Zhang, A. Haghani, A gradient boosting method to improve travel time prediction, Transp. Res. C (ISSN: 0968-090X) 58 (2015) 308–324, http://dx.doi.org/10.1016/j.trc.2015.02.019.
[46] R. More, A. Mugal, S. Rajgure, R.B. Adhao, V.K. Pachghare, Road traffic prediction and congestion control using artificial neural networks, in: 2016 International Conference on Computing, Analytics and Security Trends, CAST, 2016, pp. 52–57, http://dx.doi.org/10.1109/CAST.2016.7914939.
[47] T.S. Tamir, G. Xiong, Z. Li, H. Tao, Z. Shen, B. Hu, H.M. Menkir, Traffic congestion prediction using decision tree, logistic regression and neural networks, IFAC-PapersOnLine (ISSN: 2405-8963) 53 (5) (2020) 512–517, http://dx.doi.org/10.1016/j.ifacol.2021.04.138.
[48] Z.-q. Sun, J.-q. Feng, W. Liu, X.-m. Zhu, Traffic congestion identification based on parallel SVM, in: 2012 8th International Conference on Natural Computation, 2012, pp. 286–289, http://dx.doi.org/10.1109/ICNC.2012.6234663.
[49] Y. yuan Chen, Y. Lv, Z. Li, F.-Y. Wang, Long short-term memory model for traffic congestion prediction with online open data, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems, ITSC, 2016, pp. 132–137, http://dx.doi.org/10.1109/ITSC.2016.7795543.
[50] A. Crivellari, E. Beinat, Forecasting spatially-distributed urban traffic volumes via multi-target LSTM-based neural network regressor, Mathematics (ISSN: 2227-7390) 8 (12) (2020) http://dx.doi.org/10.3390/math8122233, URL https://www.mdpi.com/2227-7390/8/12/2233.
[51] J. Kim, G. Wang, Diagnosis and prediction of traffic congestion on urban road networks using Bayesian networks, Transp. Res. Rec. 2595 (1) (2016) 108–118, http://dx.doi.org/10.3141/2595-12.
[52] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang, Traffic flow prediction with big data: A deep learning approach, IEEE Trans. Intell. Transp. Syst. 16 (2) (2015) 865–873, http://dx.doi.org/10.1109/TITS.2014.2345663.
[53] X. Ma, Z. Tao, Y. Wang, H. Yu, Y. Wang, Long short-term memory neural network for traffic speed prediction using remote microwave sensor data, Transp. Res. C (ISSN: 0968-090X) 54 (2015) 187–197, http://dx.doi.org/10.1016/j.trc.2015.03.014.
[54] W. Xiangxue, X. Lunhui, C. Kaixun, Data-driven short-term forecasting for urban road network traffic based on data processing and LSTM-RNN, Arab. J. Sci. Eng. 44 (2019) 3043–3060, http://dx.doi.org/10.1007/s13369-018-3390-0.
[55] J. An, L. Fu, M. Hu, W. Chen, J. Zhan, A novel fuzzy-based convolutional neural network method to traffic flow prediction with uncertain traffic accident information, IEEE Access 7 (2019) 20708–20722, http://dx.doi.org/10.1109/ACCESS.2019.2896913.
[56] Y. Tian, K. Zhang, J. Li, X. Lin, B. Yang, LSTM-based traffic flow prediction with missing data, Neurocomputing (ISSN: 0925-2312) 318 (2018) 297–305, http://dx.doi.org/10.1016/j.neucom.2018.08.067.
[57] Y. Cong, J. Wang, X. Li, Traffic flow forecasting by a least squares support vector machine with a fruit fly optimization algorithm, Procedia Eng. (ISSN: 1877-7058) 137 (2016) 59–68, http://dx.doi.org/10.1016/j.proeng.2016.01.234.
[58] J. Tang, F. Liu, Y. Zou, W. Zhang, Y. Wang, An improved fuzzy neural network for traffic speed prediction considering periodic characteristic, IEEE Trans. Intell. Transp. Syst. 18 (9) (2017) 2340–2350, http://dx.doi.org/10.1109/TITS.2016.2643005.
[59] F. Moretti, S. Pizzuti, S. Panzieri, M. Annunziato, Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling, Neurocomput. (ISSN: 0925-2312) 167 (C) (2015) 3–7, http://dx.doi.org/10.1016/j.neucom.2014.08.100.
[60] D. Chen, Research on traffic flow prediction in the big data environment based on the improved RBF neural network, IEEE Trans. Ind. Inform. 13 (4) (2017) 2000–2008, http://dx.doi.org/10.1109/TII.2017.2682855.
[61] T.R. Bandaragoda, D. De Silva, D. Alahakoon, Automatic event detection in microblogs using incremental machine learning, J. Assoc. Inform. Sci. Technol. 68 (10) (2017) 2394–2411, http://dx.doi.org/10.1002/asi.23896.
[62] D. De Silva, X. Yu, D. Alahakoon, G. Holmes, A data mining framework for electricity consumption analysis from meter data, IEEE Trans. Ind. Inform. 7 (3) (2011) 399–407, http://dx.doi.org/10.1109/TII.2011.2158844.
[63] R. Mandal, P. Karmakar, S. Chatterjee, D.D. Spandan, S. Pradhan, S. Saha, S. Chakraborty, S. Nandi, Exploiting multi-modal contextual sensing for city-bus's stay location characterization: Towards sub-60 seconds accurate arrival time prediction, 2021, arXiv:2105.13131.
[64] Y. Gu, Y. Wang, S. Dong, Public traffic congestion estimation using an artificial neural network, ISPRS Int. J. Geo-Inf. (ISSN: 2220-9964) 9 (3) (2020) http://dx.doi.org/10.3390/ijgi9030152.
[65] S. Iyer, K. Boxer, L. Subramanian, Urban traffic congestion mapping using bus mobility data, in: CEUR Workshop Proceedings, vol. 2227, CEUR-WS, 2018, pp. 7–13.
[66] K. Nguyen, J. Yang, Y. Lin, J. Lin, Y.-Y. Chiang, C. Shahabi, Los angeles metro bus data analysis using GPS trajectory and schedule data (Demo paper), in: Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL '18, ISBN: 9781450358897, 2018, pp. 560–563, http://dx.doi.org/10.1145/3274895.3274911.
[67] N.C. Petersen, F. Rodrigues, F.C. Pereira, Multi-output bus travel time prediction with convolutional LSTM neural network, Expert Syst. Appl. (ISSN: 0957-4174) 120 (2019) 426–435, http://dx.doi.org/10.1016/j.eswa.2018.11.028.
[68] S. Nadeeshan, A.S. Perera, Multi-step bidirectional LSTM for low frequent bus travel time prediction, in: 2021 Moratuwa Engineering Research Conference (MERCon), 2021, pp. 462–467, http://dx.doi.org/10.1109/MERCon52712.2021.9525709.
[69] T. Zhou, W. Wu, L. Peng, M. Zhang, Z. Li, Y. Xiong, Y. Bai, Evaluation of urban bus service reliability on variable time horizons using a hybrid deep learning method, Reliab. Eng. Syst. Saf. (ISSN: 0951-8320) 217 (2022) 108090, http://dx.doi.org/10.1016/j.ress.2021.108090.
[70] J. Weng, C. Wang, H. Huang, Y. Wang, L. Zhang, Real-time bus travel speed estimation model based on bus GPS data, Adv. Mech. Eng. 8 (11) (2016) 1687814016678162, http://dx.doi.org/10.1177/1687814016678162.
[71] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. (ISSN: 0899-7667) 9 (8) (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[72] Pécs Town Hall, Pécs city web site, 2021, https://pecs.hu, (Accessed 12 December 2021).
[73] Tüke Busz Zrt., Tüke busz zrt. Web site, 2021, http://www.tukebusz.hu, (Accessed 12 December 2021).
[74] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. (ISSN: 2157-6904) 10 (2) (2019) http://dx.doi.org/10.1145/3298981.
[75] Mila, BCG GAMMA, Haverford College, Comet, CodeCarbon, 2022, https://codecarbon.io, (Accessed 12 August 2022).
[76] J. Mocnej, A. Pekar, W.K. Seah, P. Papcun, E. Kajati, D. Cupkova, J. Koziorek, I. Zolotova, Quality-enabled decentralized IoT architecture with efficient resources utilization, Robot. Comput.-Integr. Manuf. (ISSN: 0736-5845) 67 (2021) 102001, http://dx.doi.org/10.1016/j.rcim.2020.102001.
[77] A. Pekar, J. Mocnej, W.K.G. Seah, I. Zolotova, Application domain-based overview of IoT network traffic characteristics, ACM Comput. Surv. (ISSN: 0360-0300) 53 (4) (2020) http://dx.doi.org/10.1145/3399669.
[78] M. McCloskey, N. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, Psychol. Learn. Motiv. Adv. Res. Theory (ISSN: 0079-7421) 24 (C) (1989) 109–165, http://dx.doi.org/10.1016/S0079-7421(08)60536-8.
[79] J. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. (ISSN: 0360-0300) 46 (4) (2014) http://dx.doi.org/10.1145/2523813.
[80] A. Royer, C.H. Lampert, Classifier adaptation at prediction time, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015, pp. 1401–1409, http://dx.doi.org/10.1109/CVPR.2015.7298746.
[81] G.I. Webb, R. Hyde, H. Cao, H.L. Nguyen, F. Petitjean, Characterizing concept drift, Data Min. Knowl. Discov. (ISSN: 1573-756X) 30 (4) (2016) 964–994, http://dx.doi.org/10.1007/s10618-015-0448-4.