Using deep learning to detect anomalies in traffic flow *

Manuel Méndez[1], Alfredo Ibias[2], and Manuel Núñez[1][0000−0001−9808−6401]

[1] Universidad Complutense de Madrid, Madrid, Spain
manumend@ucm.es,manuelnu@ucm.es
[2] SANO-Centre for Computational Medicine, Krakow, Poland
a.ibias@sanoscience.org

Abstract. Uncertainty is an ever present challenge in data analysis. In particular, it is important to detect, as precisely as possible, unforeseen phenomena. In this paper we study the usefulness of two deep learning based methods (CNN auto-encoder and BiLSTM auto-encoder) to detect anomalies in situations that can be defined in terms of time series. In order to evaluate our approaches, we consider traffic flow data and perform experiments in two orthogonal scenarios: a guided scenario (training only with data considered as 'normal' after a naïve labelling) and a basic scenario. Our results show that if we train the models using only the considered 'normal' data, the obtained models do not achieve good results because none of them are able to detect all type of abnormal data correctly. In contrast, both models can detect all type of time series anomalies when we consider the basic scenario.

Keywords: Deep learning; Auto-encoders; Anomalies detection; Unsupervised learning

1 Introduction

Anomaly detection [15] is a process that consists in detecting different abnormal behaviours, called *outliers*, in a dataset and provide useful information about their causes. Applications of anomaly detection include fraud detection [13], heart abnormalities [20], fault detection [10] and spam classification [16], among many others. In addition, anomaly detection can be applied to the analysis of time series. For example, it can be applied to streaming data related to the Internet of Things [24] and the stock market [14]. In addition, it can be applied to traffic flow time series, the application field that we consider in this paper. This field presents a main issue that does not appear in the previously mentioned cases: the lack of knowledge about what an anomaly is. Therefore, this technique requires the application of unsupervised models. At a first glance, we might be tempted to think that anomalies are the same as extreme values. Nevertheless, ideally, a well-performed model should be able to detect not just the extreme values but the values that imply, for example, the failure of a given pattern or a sudden increase or decrease from

* This work has been supported by the Spanish MINECO/FEDER projects FAME (RTI2018-093608-B-C31) and AwESOMe (PID2021-122215NB-C31) and the Region of Madrid project FORTE-CM (S2018/TCS-4314) co-funded by EIE Funds of the European Union.

the previous value. Summarising, the definition of either a "normal" or an "abnormal" data should be a time-dependent quantity [24].

Several deep learning methods have been proposed to identify abnormal data in time series. Recurrent neural networks (RNN), mainly long short-term memory (LSTM) networks based models, are the primary option. RNN are applied in two different scopes. On the one hand, RNN aims to predict the future values and, subsequently, compare them with the actual values or with predefined thresholds by determining whether the corresponding data is or not an abnormal value [4, 12, 21]. On the other hand, auto-encoders (AE) or variational auto-encoders (VAE) based on RNNs (specifically, on LSTMs) are developed. Their aim is to restore the original values and compare the actual and the reconstructed values. Then, a threshold is established. If the discrepancy between the actual and the reconstructed value is greater than the threshold, then the corresponding data will be considered as abnormal [8, 11, 22].

In this work, we present two models to detect anomalies in time series data by using previous data as input. In order to assess the usefulness of our models, we will apply them to traffic flow data obtained in the city of Madrid. First, we develop a CNN auto-encoder. In addition to the dependent variable, this model will use another eight independent variables related to traffic flow, that is, we will be working within a multivariate time series. The second model is a novel Bidirectional LSTM auto-encoder model. This model will only use as input the traffic flow data (dependent variable), that is, we will work within a univariate time series. These models will be applied in both a basic scenario and a guided scenario (training only with data considered as 'normal' after a naïve labelling).

Anomaly detection in traffic flow by employing auto-encoders is an unexplored research line. There are some recent approaches dealing with this task by using other techniques. A method known as *local outlier factor* was employed in the city of Odense [7]. By leveraging the availability of video surveillance, an algorithm based on fuzzy theory was developed to detect anomalies in road traffic flow [18]. An original and novel method, which does not take into account the historical data but uses expert feedback to deal with the fluid definition of anomaly, was developed in Vienna and was able to provide high accuracy [1]. To detect anomalies in Zagreb traffic flow, a tensor based method was proposed [23]. An adapted k-nearest neighbours method was used with the same purpose in Beijing [5]. If the reader is interested in further discussion on this topic, there is a survey about urban traffic flow anomalies detection algorithms [6]. To the best of our knowledge, there are no scientific contributions that develop a Bidirectional auto-encoder to detect anomalies in a time series, despite the fact of its capability to obtain good results in unsupervised scenarios.

The rest of the paper is organised as follows. In Section 2 we present the main characteristics of the anomaly detection problem in traffic flow, emphasising the application to our specific case study and used data. In Section 3 we present the main characteristics of our two models. In Section 4 we present our experiments and analyse the obtained results. Finally, in Section 5 we give our conclusions and outline some directions for future work.

2 Problem description

In this section, we will describe the data used in this work and, subsequently, we will present the two scenarios that we consider: a guided scenario and a basic one.

2.1 Data

We use hourly traffic flow data of a station located in Madrid. We will use the last available 30 months of data: from January 2018 to July 2020. We also have eight additional variables related to traffic flow data that will be used to allow the model to better understand the context. These eight variables are:

- (1, 2) *Hourly traffic flow* in two near *additional stations*.
- (3, 4, 5) *Daily average*, *maximum* and *minimum temperature*.
- (6) *Daily rainfall*.
- (7) *Type of day* (working day, weekend or public holiday).
- (8) *Time* stamp of the data.

The values of the weather variables (3–6) where measured in the nearest weather station, while the traffic flow concerning additional stations (1–2) was measured in the two nearest traffic stations to the target traffic station.

2.2 Scenarios

Before we describe the considered scenarios, it is important to know the types of abnormal values that we can expect in time series [19]. These are:

- A value is considered a *global anomaly* if it is far outside of the dataset range.
- A value is considered a *contextual anomaly* if it significantly deviates from other data in a similar context.
- A subset of the dataset is considered a *collective anomaly* if those values, as a subset, significantly deviate from the dataset, but individually the values are not considered anomalous by themselves.

The goal of this paper is to provide a method to automatically detect abnormal behaviour of traffic flow data in Madrid using two different scenarios.

In the basic scenario we use all the data for training, and the aim of the models is to analyse and detect values corresponding to abnormal data. Obviously, the efficacy of the model cannot be assessed with traditional methods. Thus, we will evaluate the models by graphically analysing whether the values considered as anomalies correspond to any type of abnormal value in time series and by checking whether any remarkable event happened in Madrid in the date in which the abnormal data is detected.

In the guided scenario the aim is the same but data is divided into two sets. The first set includes the values greater than a high percentile of the dataset, which we know for sure they are abnormal values. The second set includes the rest of the data values. Obviously, not all abnormal values of the time series are included in the first set.

However, the goal of this division is to check whether training the algorithm without some abnormal values improves or not the overall performance of the algorithm.

Formally, if X is the input data, then the models in both scenarios aim to find a function f such that:

$$f(X, \delta) = y,$$

being $y = 0$ if the value is not considered an anomaly and $y = 1$ if the value is considered an anomaly. The δ parameter is a pre-set value whose role depends on the model and will be described in the next section.

3 Auto-encoder models

In this section we will present the basics of the models that will be applied in this work: *CNN* and *BiLSTM* auto-encoder models. The interested reader can find additional information about these models in specialised work [9, 17] that introduce the principles of LSTM and CNN neural networks. First, let us describe how an auto-encoder works.

A variational auto-encoder (VAE) is an algorithm composed by an encoder and a decoder. The idea is that the encoder will translate the input, X, from its high-dimensional space to a lower-dimensional space. Then, the decoder will take that lower-dimensional representation of the input, what we usually called the *latent vector*, and will try to reconstruct the original input in the high-dimensional space, producing $\tilde{X}$. Due to its nature, the decoder usually has a structure that mimics the one of the encoder, being its goal to obtain a value $\tilde{X}$ as similar as possible to the original input, X. The dissimilarity between the original and reconstructed data is defined by a loss function. Therefore, the goal of the model is to reduce, as much as possible, this value during the training process.

This model is trained in a unsupervised way due to the output trying to be the input and, therefore, there is no need for labels. We will use two versions of a VAE to create the encoder and decoder: one using CNNs and another using BiLSTMs. We will use these models over two scenarios: a *basic* scenario and a *guided* one.

In both scenarios we train our model and then compute the dissimilarity score of all the trained data. This dissimilarity score is computed by comparing the input data with the reconstructed output data using a dissimilarity measure. We will use these dissimilarity scores to compute a threshold δ. This threshold is usually set to have the $90 - 99$th percentile of dissimilarity scores as abnormal data. Then, for new data, if the model can reconstruct input data with a dissimilarity value lower than δ, then it will be considered *normal data*; otherwise, it will be classified as *abnormal data*.

Regarding our scenarios, the difference is in training. In the basic scenario, we train the model with all the data (normal and abnormal), while in the guided scenario we train the model with only the data that we consider normal (the ones in the second set explained before). The idea is that, in this second case, the algorithm will learn to represent better the normal data, as the abnormal data has been taken away from training. Consequently, we expect the reconstruction of abnormal data to be more deficient, what will allow us to detect them more easily.

3.1 CNN auto-encoder model

Let y be the traffic flow in the target station, $x^1, \ldots, x^8$ be the additional variables defined in Section 2.1 and t be a time step. In order to decide whether the traffic flow data in time step t, that is y_t, is an abnormal data, we will use the variables $y, x^1, \ldots, x^8$ from time step $t-23$ to t. We reshape these values into a 24×9 matrix, where 24 is the number of values that we use in order to detect the anomalies. Therefore, we have

$$X = \begin{bmatrix} y_{t-23} & x^0_{t-23} & \cdots & x^8_{t-23} \\ y_{t-22} & x^0_{t-22} & \cdots & x^8_{t-22} \\ \cdots & \cdots & \cdots & \cdots \\ y_{t-1} & x^0_{t-1} & \cdots & x^8_{t-1} \\ y_t & x^0_t & \cdots & x^8_t \end{bmatrix}$$

In addition, we will use as dissimilarity loss function the *mean absolute error*:

$$Loss(X, \tilde{X}) = MAE = \frac{1}{216} \sum_{i=1}^{24} \sum_{j=1}^{9} |X_{i,j} - \tilde{X}_{i,j}|,$$

where i and j represent, respectively, the rows and columns of the matrix.

Concerning the specific structure of our CNN model, as we previously said, this model is composed by two parts: encoder and decoder. Our encoder has two convolutional layers of 32 and 64 filters, respectively, each of them with a kernel size of 3 and a stride of 2. Then, a layer vector is added in order to transform the data into a flatten vector. Finally, we add a dense layer with 8 units to generate the 8-dimensional latent vector. Our decoder is composed by a set of layers whose operation is exactly the opposite to that of the encoder. First, we add a dense layer, which increases the dimension to the dimension of the flatten layer, and then we add a reshape layer, which modifies the data shape into a matrix form. Finally, we add three transpose convolutional layers of 64, 32 and 1 filters, respectively, a kernel size of 3 and a stride of 2. Summarising, the inverse encoder process is the one used to reconstruct input data from the latent vector.

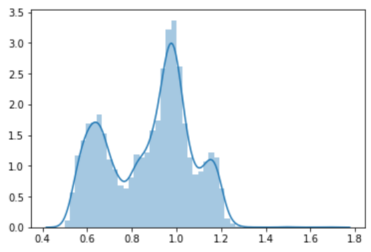
3.2 BiLSTM auto-encoder model.

This model is also composed by an encoder and a decoder. In this case, BiLSTM layers are used because they leverage the advantages of BiLSTM and encoder-decoder architectures, respectively. The general behaviour of this model is similar to the previously described CNN model.

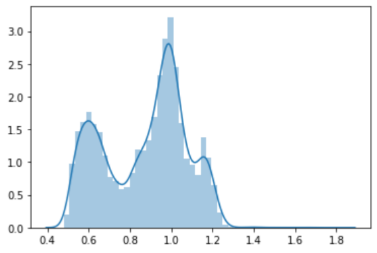
In order to determine whether traffic flow data in time step t, that is the y_t value, is an abnormal data, we will use the univariate time series of traffic flow, from time step $t-23$ to t. We use as input data a 24-sized vector, where 24 is the number of previous values that we use in order to detect anomalies. Therefore, we have the following:

$$X = [y_{t-23}, y_{t-22}, \ldots, y_{t-1}, y_t]$$

In this case, the dissimilarity is also measured between the input sequence and the output, that is, the reconstructed sequence. We use again the *mean absolute error* as dissimilarity loss function:



(a) (P. Sta María, CNN auto-encoder)



(b) (P. Sta María, BiLSTM auto-encoder)

Fig. 1: Histogram of the dissimilarity loss distribution in training data.

$$Loss(X, \tilde{X}) = MAE = \frac{1}{24} \sum_{t-23}^{t} |y_i - \tilde{y}_i|.$$

Concerning the specific structure of our model, the encoder is composed of a BiLSTM layer with 128 units followed by a dropout layer, with a rate of 0.2, which reduces the over-fitting of the model. Then, we add a repeat vector layer which repeats the input k times (one for time step). Our decoder also includes a BiLSTM layer with 128 units, which in this case returns the entire sequence, followed by a dropout layer with a rate of 0.2. Finally, a dense layer with one unit is added to the network in order to output each time step in the final output sequence. We would like to emphasise that in this model we do not use predictor variables, as in the previous case, because BiLSTM networks do not work optimally by using matrices as input. Therefore, using as input the matrix considered in previous case, even as a flatten vector, would imply a significant loss of the quality of the model.

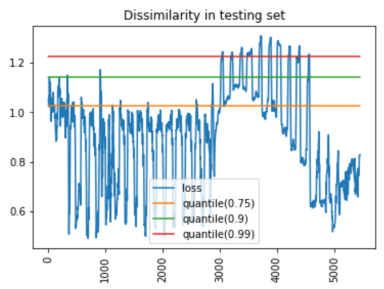
4 Experiments

In this section we report on the experiments that we performed to evaluate both models in the considered scenarios.

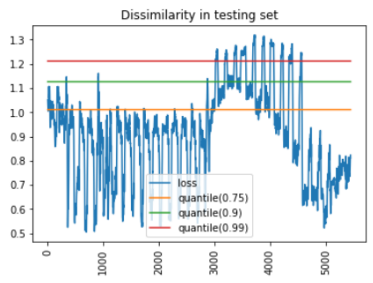
4.1 Basic scenario

We will analyse the results of applying the CNN and BiLSTM auto-encoder models to the basic scenario for Paseo de Santa María de la Cabeza traffic station (N$40.4065°$ E$3.6946°$). In both cases, we will use as training set data 16.391 observations and as testing set data 5.448 observations. We train all the models by using 15 epochs and 16 as batch size.

For both models, we will plot three different graphs. The first graphs (see Figure 1) are histograms with 50 bins that represent the distribution of the dissimilarity loss in training data. A higher concentration of dissimilarity loss with low representation in the



(a) (P. Sta María, CNN auto-encoder)



(b) (P. Sta María, BiLSTM auto-encoder)

Fig. 2: Specific data detected as anomalous by threshold.

extreme right of the graphs gives the possibility to choose higher thresholds that virtually will not return any false positive at time to detect anomalies. The second graphs (see Figure 2) represent the dissimilarity score for the observations of the testing set and which of them are considered anomalies depending on the selected threshold. The y-axis represents the dissimilarity loss and the x-axis represents each observation of the testing set. The chosen thresholds (represented by horizontal lines) are the 75th, 90th and 99th quantiles of dissimilarity loss in training data. The values above each horizontal line are the anomalies considered according to its respective threshold. The third pair of graphs (see Figures 3 and 4) show the detected anomalies (plotted in the time series). The x-axis represents the date while the y-axis represents the hourly traffic flow. The different coloured points show the values considered abnormal data depending on the selected threshold. Obviously, the data considered abnormal by using a given quantile as threshold will be also considered abnormal by higher quantiles. In these graphs, we can analyse, graphically and by date, the reasons why different observations are considered anomalies.

Comparison between both models A simple glance shows a great similarity between the results in the two models. However, there are some remarkable differences that we would like to discuss. On average, the dissimilarity measure is smaller in the BiLSTM auto-encoder model and the thresholds are higher in the CNN auto-encoder model. Consequently, the BiLSTM auto-encoder will detect more possible outliers than the CNN auto-encoder. By using the 99th quantile as threshold (see Figures 3 and 4), the anomalies are detected in a same range of dates that, interestingly enough, corresponded to the beginning of the restrictions caused by the COVID-19 pandemic. By using the 90th quantile as threshold, both models can also detect as anomalies data which corresponds to the Christmas festivities and to the last weekend of November. However, the BiLSTM auto-encoder detects more outliers in the range of dates that correspond to the restrictions (with 90th quantile as threshold, BiLSTM auto-encoder obtains 788 anomalies in total versus the 638 obtained by CNN auto-encoder). By using the 75th quantile as threshold, there are some detected anomalies which do not correspond to

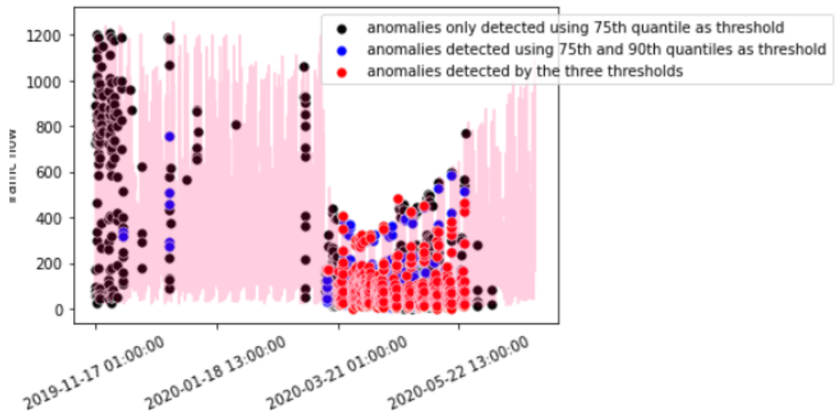


Fig. 3: Specific data detected as anomalous by threshold (P. Sta María, CNN auto-encoder).

any important date or to any graphical deviation. Thus, we may conclude that they are not actual anomalies.

As explained before, if we use smaller quantiles as threshold, then the number of detected anomalies increases. Figures 5 and 6 show this effect in both models. Interestingly enough, the percentage of anomalies remains practically constant in the CNN auto-encoder model from the 90th quantile up to the 99th quantile while in the BiLSTM auto-encoder model, this *stability* happens only from the 95th quantile up to the 99th quantile. The presence of this stability in a wider range implies a greater certainty on the anomalies detected by using as threshold the lower end of the range under consideration.

In conclusion, both models obtain similar results in the basic scenario. However, the BiLSTM auto-encoder model shows a better reconstruction of the input and, therefore, it can detect more anomalies with the same quantile set as threshold. Moreover, we should take into account that the CNN auto-encoder needs eight independent variables while the BiLSTM auto-encoder only needs the time series of the target variable.

4.2 Guided scenario

We will analyse the results by using the CNN and BiLSTM auto-encoder models in the guided scenario for Paseo de Santa María de la Cabeza station (the one that we used in the basic scenario). We propose to consider for the abnormal set every value higher than the 90% and 99% of the range of traffic flow time series data. The idea is to analyse how modifying the number of abnormal data used for training affects the performance of the algorithm. We use 75th, 90th and 99th quantiles for the dissimilarity loss threshold.

In contrast with the basic scenario, we will use as training set only the data that is considered "normal", that is, those data points that are not in the abnormal set. Thus, ideally, the models will learn the representation of normal data and will obtain a small

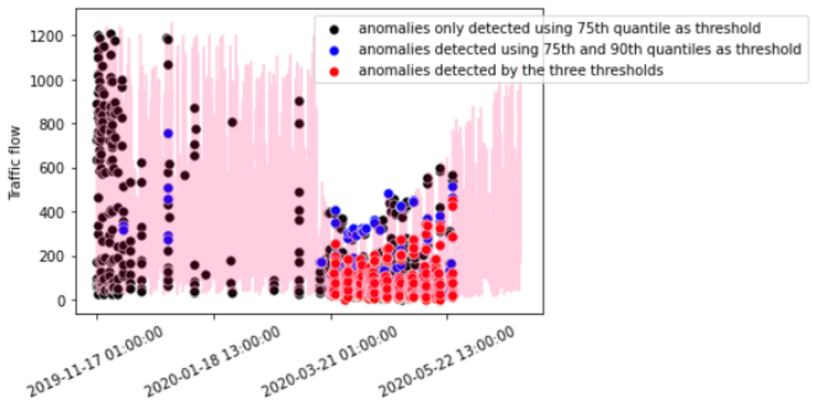
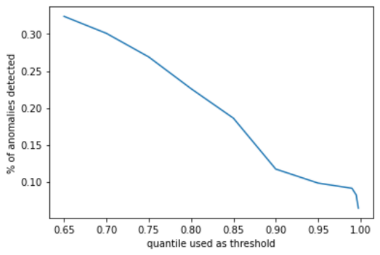


Fig. 4: Specific data detected as anomalous by threshold (P.Sta María, BiLSTM auto-encoder).



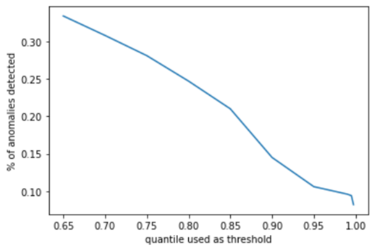
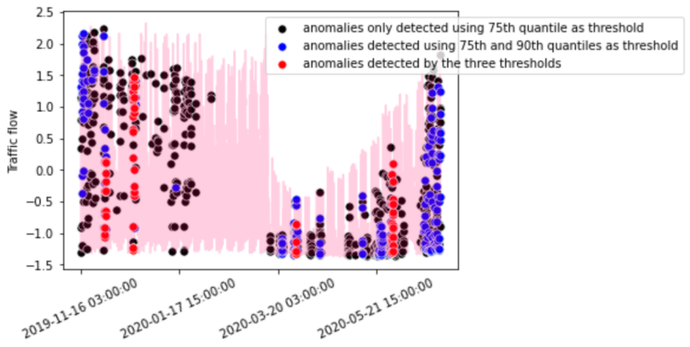


Fig. 5: % of anomaly detection by quantile used as threshold (CNN auto-encoder).

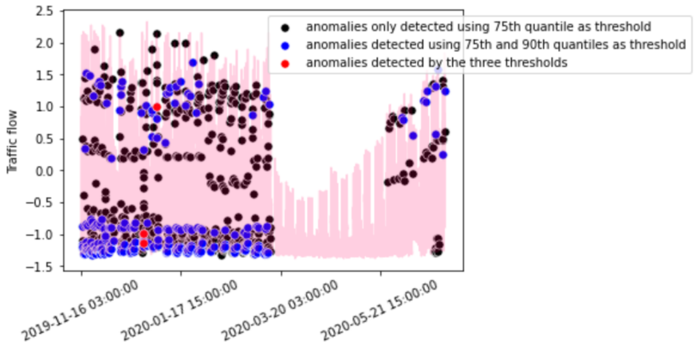
Fig. 6: % of anomaly detection by quantile used as threshold (BiLSTM auto-encoder).

dissimilarity loss. Therefore, it is expected that when we consider the testing set (including, in particular, abnormal data), the dissimilarity loss will be greater.

Comparison between models considering as lower limit of abnormal data the 90% of the data range If we compare the models by considering as lower limit of abnormal data the 90% of the data range (see Figures 7a and 7b) we appreciate a different behaviour between them. On the one hand, regarding the CNN auto-encoder model, the model with 99th quantile as threshold can detect only the beginning of specific and remarkable periods such as December public holidays, Christmas holidays or the most restrictive period caused by the COVID19 pandemic. However, by using either the 90th or 75th quantiles, the abnormal data detected are not apparently related to any remarkable date or event. On the other hand, regarding the BiLSTM auto-encoder model, the model with 99th quantile as threshold only detects three abnormal timesteps, which seems a very poor result. By using either the 90th or 75th quantiles, the model detects data of the extremes. That is, very high or very low values. But, for example, the model does not appreciate an abnormal behaviour in the COVID19 period. That is clearly a



(a) CNN auto-encoder.

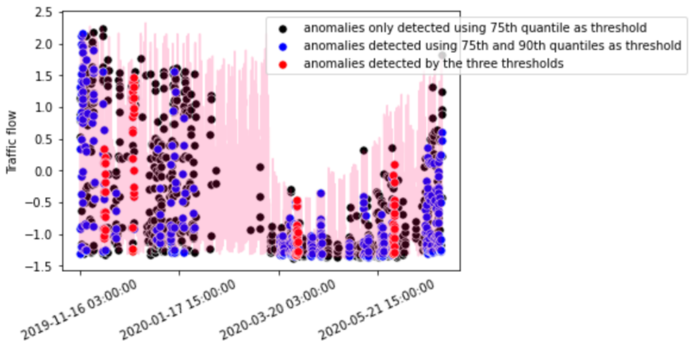


(b) BiLSTM auto-encoder.

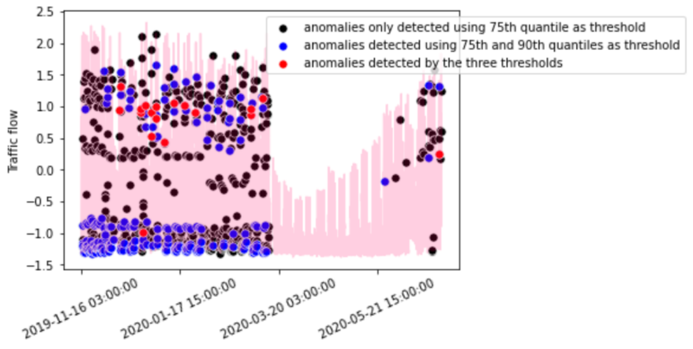
Fig. 7: Specific data detected as anomalous by threshold (considering as lower limit of abnormal data the 90% of the data range).

direct cause of the training method. In general, both models obtain worst results in this scenario than in the basic scenario. Nevertheless, the CNN auto-encoder can detect, although in a poorly way, all type of abnormal data while the BiLSTM auto-encoder can detect only the abnormal data known as *global anomaly*.

Comparison between models considering as lower limit of abnormal data the 99% of the data range If we consider as lower limit of abnormal data the 99% of the data range (see Figures 8a and 8b) we appreciate a similar behaviour to the previous case. As a first conclusion, we have that an increase of the lower limit of abnormal data implies an increase of the number of detected abnormal data. Regarding the CNN auto-encoder model, the model with 99th quantile as threshold detects exactly the same abnormal data as in the previous case. By using 90th or 75th quantiles, the detected abnormal data increases significantly but still there is no relation to any remarkable date or event. Regarding the BiLSTM auto-encoder model, the model with 99th quantile as threshold can detect more abnormal data than in previous case, but they are only *global anomalies*. By using the 90th and 75th quantiles, the model detects practically the same data as



(a) CNN auto-encoder.



(b) BiLSTM auto-encoder.

Fig. 8: Specific data detected as anomalous by threshold (considering as lower limit of abnormal data the 99% of the data range).

in the previous case. Again, we conclude that the CNN auto-encoder, especially using 99th quantile as threshold, can detect exclusively abnormal data (but not all) of the three type defined above, while the BiLSTM model can detect almost all global anomalies, but none of other type.

5 Conclusions and future work

We have developed two anomaly detection methods based on neural networks: CNN auto-encoder and BiLSTM auto-encoder. Both models have been applied in a guided and a basic scenarios with the goal of detecting abnormal data. In the guided scenario, we break up values higher than an established quantile into an abnormal set. Results indicate that the CNN auto-encoder, although can detect anomalies of the three types, cannot detect a significant number of them while BiLSTM auto-encoder can only detect anomalies of one type (global anomalies). Therefore, we conclude that the usefulness of these models in this scenario is limited. However, in the basic scenario, we appreciate that both models can virtually detect all anomalies of the three types (global anomalies, contextual anomalies and collective anomalies). These anomalies can be appreciated both graphically and by date.

Generally, we conclude that auto-encoder methods work fine to detect anomalies in time series in a basic scenario. Moreover, they work well with all types of abnormal data because they base their operation on the difference between the input and the reconstructed output. However, standard deep learning methods do not work as well because they base their operation on finding similar characteristics between data, but different types of anomalies are not similar to each other.

As future work, we would like to improve our proposal with better encoder/decoder architectures. We would like to combine our models with current work on Complex Event Processing [2, 3] so that we are able to more accurately detect potentially anomalous values. Also, we would like to further analyse the performance of our proposals, looking for different scenarios with which to generalise our conclusions. Finally, other interesting idea is to use our auto-encoders in other type of time series urban data such as pollutant concentrations.

References

1. Md Rakibul Alam, Ilias Gerostathopoulos, Sasan Amini, Christian Prehofer, and Alessandro Attanasi. Adaptable anomaly detection in traffic flow time series. In *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 1–9, 2019.
2. David Corral-Plaza, Inmaculada Medina-Bulo, Guadalupe Ortiz, and Juan Boubeta-Puig. A stream processing architecture for heterogeneous data sources in the Internet of Things. *Computer Standards & Interfaces*, 70:103426, 2020.
3. David Corral-Plaza, Guadalupe Ortiz, Inmaculada Medina-Bulo, and Juan Boubeta-Puig. MEdit4CEP-SP: A model-driven solution to improve decision-making through user-friendly management and real-time processing of heterogeneous data streams. *Knowledge Based Systems*, 213:106682, 2021.
4. Nan Ding, HaoXuan Ma, Huanbo Gao, YanHua Ma, and GuoZhen Tan. Real-time anomaly detection based on long short-term memory and gaussian mixture model. *Computers & Electrical Engineering*, 79:106458, 10 2019.
5. Youcef Djenouri, Asma Belhadi, Jerry Chun-Wei Lin, and Alberto Cano. Adapted k-nearest neighbors for detecting anomalies on spatio–temporal traffic flow. *IEEE Access*, 7:10015–10027, 2019.
6. Youcef Djenouri, Asma Belhadi, Jerry Chun-Wei Lin, Djamel Djenouri, and Alberto Cano. A survey on urban traffic anomalies detection algorithms. *IEEE Access*, 7:12192–12205, 2019.
7. Youcef Djenouri, Arthur Zimek, and Marco Chiarandini. Outlier detection in urban traffic flow distributions. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 935–940, 2018.
8. Narendhar Gugulothu, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. Sparse neural networks for anomaly detection in high-dimensional time series. 07 2018.
9. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
10. Yang Hong, Lisong Wang, Jiexiang Kang, Hui Wang, and Zhongjie Gao. A novel application approach for anomaly detection and fault determination process based on machine learning. In *2020 6th International Symposium on System and Software Reliability (ISSSR)*, pages 1–5, 2020.

11. Ruei-Jie Hsieh, Jerry Chou, and Chih-Hsiang Ho. Unsupervised online anomaly detection on multivariate sensing time series data for smart manufacturing. pages 90–97, 11 2019.
12. Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Söderström. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. *CoRR*, abs/1802.04431, 2018.
13. Jianguo Jiang, Jiuming Chen, Tianbo Gu, Kim-Kwang Raymond Choo, Chao Liu, Min Yu, Weiqing Huang, and Prasant Mohapatra. Anomaly detection with graph convolutional networks for insider threat and fraud detection. In *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, pages 109–114, 2019.
14. S Karthik, H V Supreetha, and S Sandhya. Detection of anomalies in time series data. In *2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, pages 1–5, 2021.
15. Vijay Kotu and Bala Deshpande. Anomaly detection. In *Data Science*, chapter 13, pages 447–465. Morgan Kaufmann, second edition, 2019.
16. Carlos Laorden, Xabier Ugarte-Pedrero, Igor Santos, Borja Sanz, Javier Nieves, and Pablo G. Bringas. Study on the effectiveness of anomaly detection for spam filtering. *Information Sciences*, 277:421–444, 2014.
17. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.
18. Yanshan Li, Tianyu Guo, Rongjie Xia, and Weixin Xie. Road traffic anomaly detection based on fuzzy theory. *IEEE Access*, 6:40281–40288, 2018.
19. A.M. Rajeswari, S.K. Yalini, R. Janani, N. Rajeswari, N. Rajeswari, and C. Deisy. A comparative evaluation of supervised and unsupervised methods for detecting outliers. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 1068–1073, 2018.
20. Rony Chowdhury Ripan, Iqbal H. Sarker, Md Hasan Furhad, Md Musfique Anwar, and Mohammed Moshiul Hoque. An effective heart disease prediction model based on machine learning techniques, 2020. Available at `https://www.preprints.org/manuscript/202011.0744/`.
21. Lifeng Shen, Zhuocong Li, and James Kwok. Timeseries anomaly detection using temporal hierarchical one-class network. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13016–13026. Curran Associates, Inc., 2020.
22. Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2828–2837, New York, NY, USA, 2019. Association for Computing Machinery.
23. Leo Tišljarić, Sofia Fernandes, Tonči Carić, and João Gama. Spatiotemporal road traffic anomaly detection: A tensor-based approach. *Applied Sciences*, 11(24), 2021.
24. Leandro von Werra, Lewis Tunstall, and Simon Hofer. Unsupervised anomaly detection for seasonal time series. In *2019 6th Swiss Conference on Data Science (SDS)*, pages 136–137, 2019.