Anomaly detection in time series data using Long Short Term Memory algorithm

Rania Souidi

NI: 111245375

Research Director

Peter A. Vanrolleghem

April 2020

Contents

List of Figures

List of Tables

1 Introduction

Strict discharge limit force water resource recovery facilities to maintain good effluent quality [Eiteneuer and Niggemann, August 2018]. New strategies are considered to optimize the WRRF performance in order to prevent water body pollution to reduce greenhouse gas emissions and to minimize treatment costs. Online monitoring of WRRF is used to successfully follow the dynamics within the plant and control the system efficiently. Sensors are installed in different compartments of the plant and data are collected with SCADA systems [Malhotra et al., 2015].
Sensors have the ability to measure at high frequency. In WRRF, sensors are immersed in the wastewater and are in permanent contact with high concentrations of particles and suspended solids. To guarantee correct measurements, sensor maintenance, calibration and validation should be performed on a regular basis. However, this does not sometimes suffice. Sensor values can deviate and give wrong values, called outliers. The uncertainty in sensor data will thus increase and decisions for process optimization and water monitoring in WRRF will be wrong. This makes the task of anomaly detection in time series data important [Aijala and Lumley, 2006].
This project will evaluate anomaly detection in time series data with a neural network method. Advanced deep learning methods are capable of capturing the patterns in time series data and predicting future trends in data. In this work, the Long Short Term Memory algorithm will be used to detect outliers within time series data [Strobl and Robillard, 2008].

2 Case Study

2.1 pilEAUte

The pilEAUte is a pilot scale WRRF located at the département de génie civil et de génie des eaux of Université Laval. It treats domestic wastewater fed from a university residence. The WRRF consists of a pumping station, a storage tank, a primary clarifier tank and two parallel biological reactors called pilot and co-pilot. pilEAUte is monitored with sensors installed at different locations in the treatment process. This allows research students to understand the treatment chain process with the purpose of modelling and optimization. All sensor data are collected by two acquisition systems. A database is accessible through an interface which gives access to all the sensor data from different locations and periods of time. However, monitoring each sensor data is a difficult task given the frequently occurring sensor failure and deviation. Using data of sensors installed

in wastewater is challenging and sensors therefore need regularly cleaning and validation/ calibration procedures. Yet, this may not be enough. Detecting when the sensor is giving wrong values is important for modelling purposes. In this project, data are taken from sensors installed at both outlets of the plant (pilot and co-pilot). They measure the $NH_4 - N$ concentration in the effluent of both trains. This variable is an important effluent quality indicator.

2.2 Operational conditions

The pilEAUte WRRF described above offers great operational flexibility to run under different conditions. Its set-up allows parallel experimentation on the efficacy of control systems for nutrient removal and energy consumption optimization. With this in mind, a new research project was started in pilEAUte. It aims to control the Dissolved Oxygen (DO) in order to have simultaneous nitrification denitrification in the biological tanks. Two control strategies were compared. In the pilot train, an alternating aeration was considered while for the co-pilot train continuous aeration was studied. The data were collected under these conditions in the summer period of 2019. In particular, the control was as follows with respect to the DO concentration:

- Aeration conditions in pilot: 30 minutes 2mg/L, 30 minutes 0mg/L (anoxic conditions).
- Aeration conditions in co-pilot: maintained at a low DO 0.25 mg/L.

3 Deep learning methods for sensor anomaly detection

3.1 Recurrent Neural Network RNN

RNNs have been found to be an answer to a lot of sequential data and natural language processing (NLP) problems. To understand how RNNs work its basic conceptual architecture should be understood.

RNN architecture:
The main difference between feed-forward neural networks and RNNs is how the input is taken to the model. A feed-forward neural network takes the inputs all at once. On the other hand, the RNN takes one input at a time. So, at each time step, the RNN model takes one input and uses the previous input as a hidden state to produce the next output. This loop continues until all inputs are used. Figure 1 describes the basic concept of a RNN.

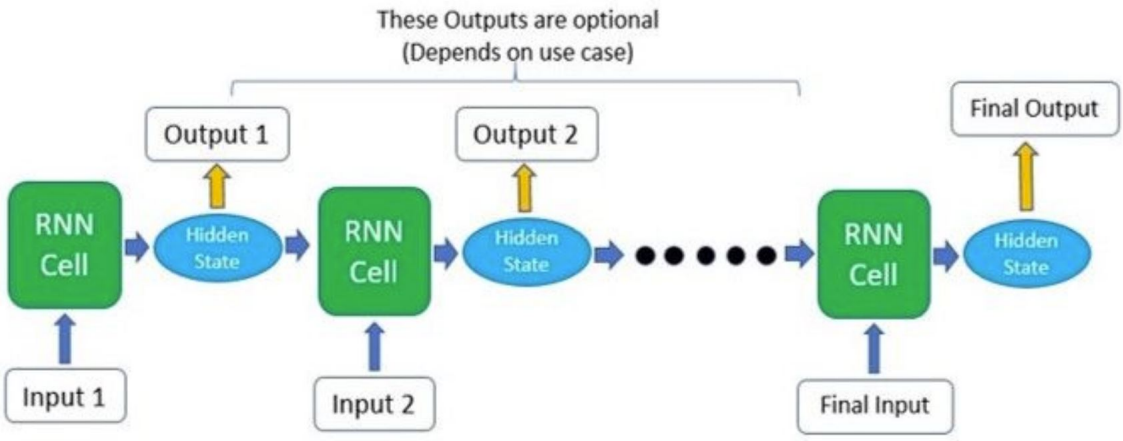

Figure 1: RNN structure
[Loyel, 2019]

3.2 Auto-Encoder Neural Network

An Auto-Encoder Neural Network for time series (AENNTS) was developed in view of detecting outliers in time series data [Jordan, 2018]. It is based on a neural network where a bottleneck is imposed to allow the NN to be compressed. Two scenarios occur:
- The inputs are independent and therefore, the reconstruction phase will be difficult.
- The inputs are correlated and their correlation structure can be learned.

Outliers are identified by applying on exponential weighted moving average (EWMA) and a mean absolute deviation (MAD) on the residuals between the original time series and a time series predicted by the AENNTS replication.

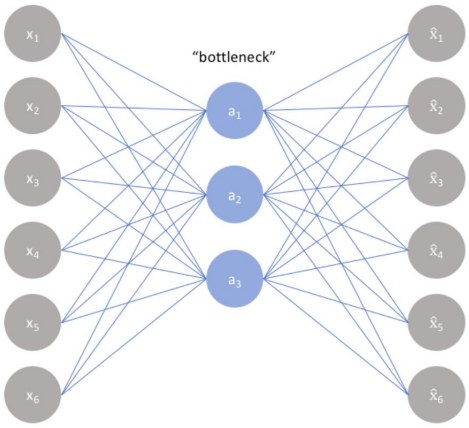

Figure 2: AENN structure [Jordan, 2018]

3.3 Long Short Term Memory LSTM

LSTM models are similar to the RNN models but with a large ability to construct a long term memory of the time series. This proved to be efficient for time series and natural language processing (NLP). It offers a large memory capacity and remembers information which a RNN models cannot do. In the section below, the LSTM model will be detailed [Loyel, 2019].

3.3.1 How does LSTM work?

The success of the LSTM is found in the gates it is built around. The LSTM models are a bit more complex than a normal RNN model. In fact, within each LSTM cell, there are three gates that take three types of information:
- The current input
- The short term memory which is referred to as the hidden state.
- The long term memory which is referred to as the cell state.

The LSTM cell uses these three gates to classify and regulate the information: which are the data that need to be kept, to be forgotten or to be discarded. In this way, the cell will work as a filter, letting only the good information to pass through and remove all irrelevant data. This task can be done by the the different gates discussed below [Loyel, 2019].

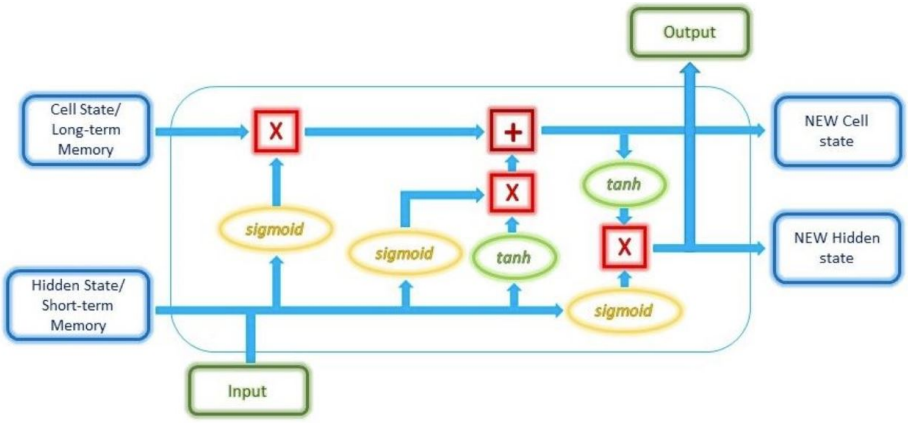

Figure 3: Overall LSTM structure
[Loyel, 2019]

The input gate

The input gate has a filter function. It specifies which information will be accepted and which one is discarded. It has two inputs: the current input data and the previous inputs referred to as the hidden state which is also known as the short term memory.

To better understand how it works, its mathematical description is necessary. Basically, it has two filters. The input data and the short term memory pass through a sigmoid function as a first filter. This nonlinear function transforms the data into an interval between 0 and 1. 0 refers to data that need to be discarded and 1 refers to accepted data. The layer will be trained by back propagation so it will learn to pass only the useful data [Loyel, 2019].

$$i_1 = \sigma(W_{i1}.(H_{t-1}, x1) + bias_{i1})$$

The second filter usually is a tanh function. It is used to regulate the network. It also takes the current input and the short term memory:

$$i_2 = tanh(W_{i2}.(H_{t-1}, x1) + bias_{i2})$$

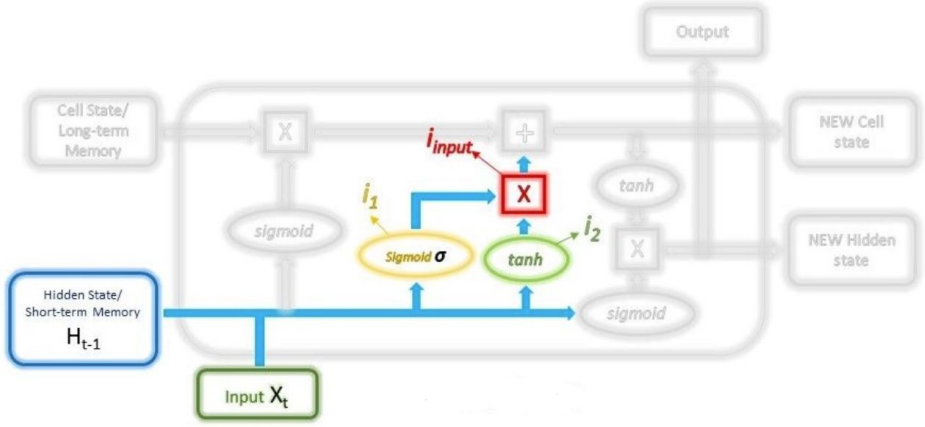

Figure 4: Input Gate [Loyel, 2019]

The outputs of these two functions are multiplied and brought to the long term memory to be used in the output gate.

$$i_{input} = i1 * i2$$

Forget gate (Figure 5)

The forget gate selects which information from the long term memory will be kept and which will be forgotten. It uses the long term memory, the current input and the short term memory as inputs.

The current input and the short term memory are passed through a sigmoid function. Again the output of the sigmoid function will be between 0 and 1 and it will be multiplied with the long term memory to select information that will be kept or discarded.

$$f = \sigma(W_{forget}.(H_{t-1}, x1) + bias_{forget})$$

The outcomes of the input gate and the forget gate are added to give the new long term memory that will be used as input to the output gate.

$$C_t = C_{T-1} * f + i_{input}$$

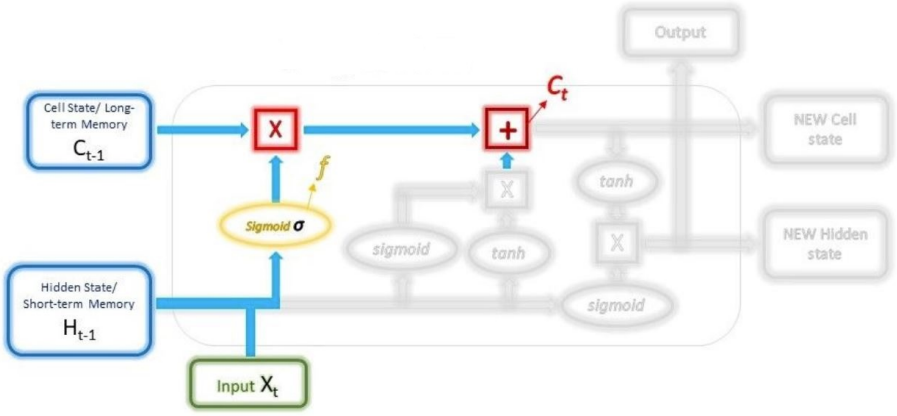

Figure 5: Forget Gate [Loyel, 2019]

Output gate (Figure 6)

The output gate takes as input, the current input, the short term memory and the long term memory. First, the previous hidden state and the current input pass through a sigmoid function creating the third and final filter (each sigmoid function in the three gates has its own weight W). On the other hand, the long term memory will pass through a tanh function. Finally, the outputs of the sigmoid and the tanh function are multiplied to get the new hidden state known as the new short term memory.

$$O_1 = \sigma(W_{output1}.(H_{t-1}, x1) + bias_{output1})$$

$$O_2 = tanh(W_{output_2}.(H_{t-1}, x1) + bias_{output_2})$$

$$H_t, O_t = O_1 * O_2$$

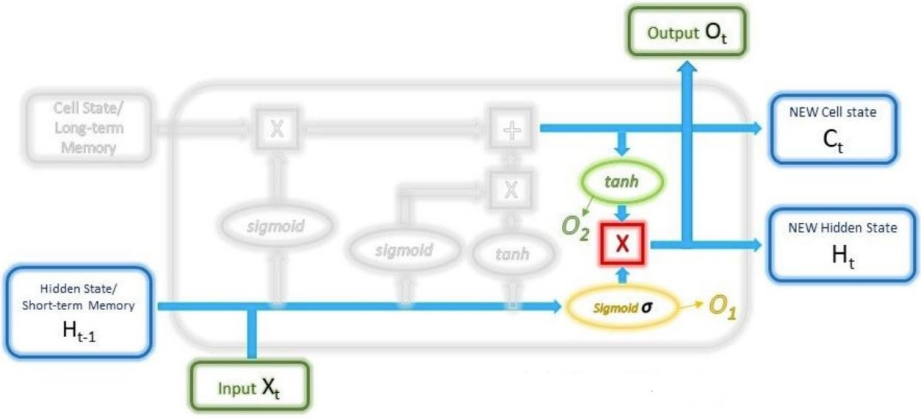

Figure 6: Output Gate [Loyel, 2019]

4 LSTM for anomaly detection in time series data

Anomaly detection is applied in different fields such as health and environmental monitoring, fraud detection, trading markets and much more [ReNom, 2016]. It is based on detecting and identifying rare events or data points that are considered outliers.
LSTM methods are largely applied in NLP and time series. They have the ability to memorise a large amount of information. To this end they use a powerful model when it comes to predicting and identifying anomalies in time series data.
This method will be applied to outlier detection in time series data [ReNom, 2016].

4.1 Dataset

As explained in section 2, the data that is used are the concentration of $NH_4 - N$ in the effluent of both pilot and co-pilot trains with the purpose of testing the model under different operational scenarios.
The concentration of NH_4 provided by one of the sensors placed in the effluent of each of the two WRRFs was chosen because of their relatively good performance during the whole period of July 2019. This will have a good effect on the LSTM model training.

Co-pilot data
The co-pilot train was under a low continuous aeration controlled at 0.25 mg/l of DO. The data fed to the model are the concentrations of $NH_4 - N$ in the effluent of the WRRF. The process of training is discussed below.

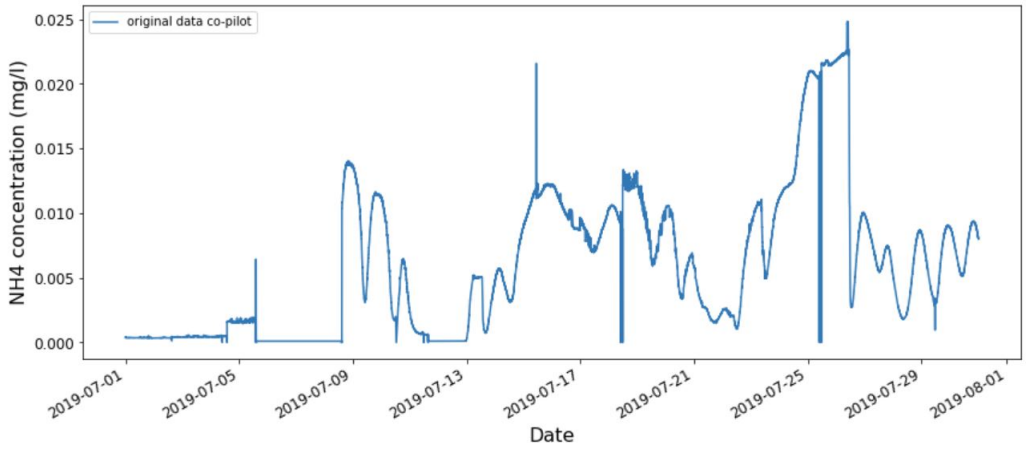

Figure 7: Input: Co-pilot $NH_4 - N$ concentrations of July 2019

Pilot data

The biological reactor in the pilot WRRF is subject to alternating aeration between 0 and 2 mg/l of DO with a time interval of 30 minutes ON and 30 minutes OFF aeration.

Again the data fed to the model are the concentrations of NH4-N in the effluent of the pilot train.

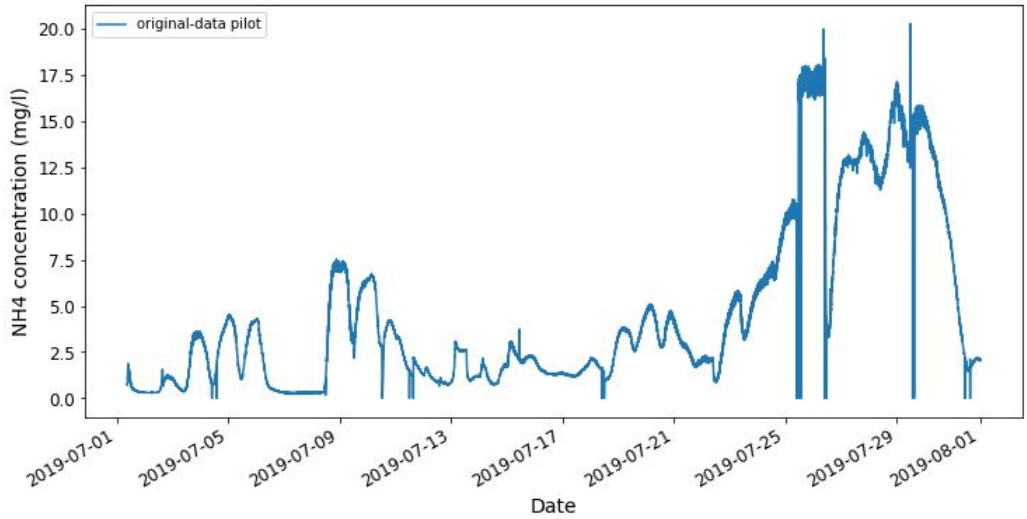

Figure 8: Input: Pilot $NH_4 - N$ concentrations of July 2019

4.2 Experimental setup

After choosing the data that will be fed to the model in both of the two scenarios (pilot and co-pilot), in this section the experimental setup that was adopted is explained. The algorithm consists of three steps:

Step 1

The LSTM model uses the previous n data to predict the next p data. The sequence of n data that is used as input X_t to the LSTM cell is determined by the user. In this work, a sequence of 10 inputs of NH4 values is used to predict the 11th value. In order to train the model, the following parameters were used:
- Input dimensions: x1, x2, x3, x4...,x10, list of time series in a m-dimensional vector is used as the input for the model to predict the 11th value. In this case, a many to one algorithm is used [ReNom, 2016].
- Hidden dimensions: this parameter represents the size of the hidden state and the cell state at each time step. At the beginning of the training, these matrices are initialized randomly.
- Number of layers: this parameter describes the number of LSTM cells stacked on top of each other.
- Batch size: Feed the the data by batch to the training model.
- Epochs: this parameter describes how many times the LSTM model will see all the data and do all the prediction and optimization processes.
- Loss function or learning rate; this controls how much the weights of the networks are adjusted with respect to the loss gradient.

Step 2

In this step, the error vector is computed by calculating: $X = X_{(pred)} - X_{(orig)}$ where X_{pred} and X_{orig} are the predicted data and the original data respectively.
In a next step, a gaussian distribution is fit to the error vectors by a maximum likelihood estimation computed based on the test data.

Gaussian distribution
The parameters of the gaussian distribution are:
- The variance σ
- The mean μ

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The optimization of the LSTM model is performed by back-propagation of

the errors.

Step 3

Now, the model can predict the outputs and by plotting both the original and the predicted data anomalies can be detected by evaluating when the error vector will be located at the extremes of the gaussian distribution.
To better visualize the anomalies, the Mahalanobis distance was calculated. It measures the distance between a point and a distribution. It is considered an excellent approach for outlier detection [Loyel, 2019].

Mahalanobis distance
This method visualizes the outliers in time series data. It shows the distribution of the data around the mean devided by the covariance matrix [Prabhakaran, 2015]. If the datset is strongly correlated, the covariance will be high. Dividing the errors by the covariance will normalize the distance. Similarly, when working with a non-correlated dataset, the covariance will be low and the distance will not be reduced significantly. This solves the problem of scale and correlation at the same time.

$$D^2 = (x-\mu)^T \cdot C^{-1} \cdot (x-m)$$

where:

- D^2: the square of the Mahalanobis distance
- : x: the vector of observations
- : μ: the vector of mean values
- : C^{-1}: the inverse covariance matrix

The pytorch library was used for model implementation.

4.3 Results

4.3.1 Pilot results

First Prediction results

As a first step, the parameters were initialized as described in Table 1. However, the results were not good and optimization of the parameters was necessary (Figure 9).

Table 1: Initialization of the LSTM model parameters

PARAMETERS	Values
Batch size	50
Epochs	5
Input sequence	10
Loss function parameter	0.01
Hidden layer size	50

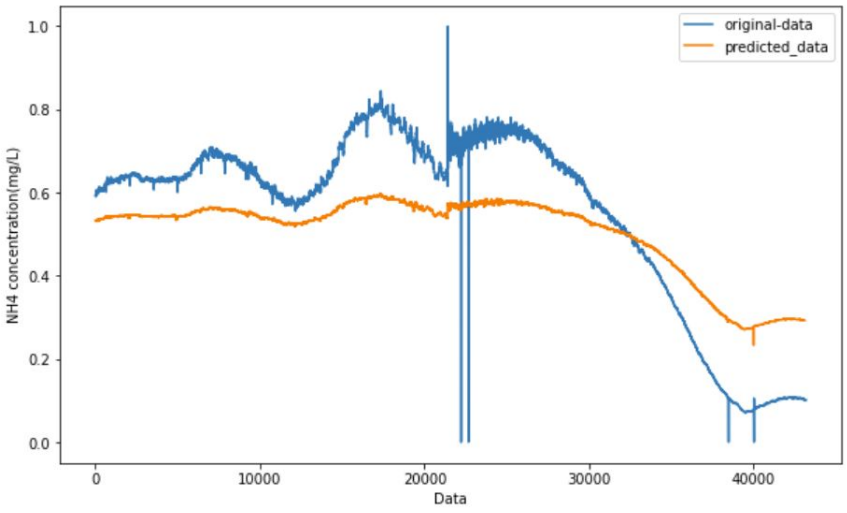

Figure 9: First prediction results on 20% of all data

Second Prediction results

In a second try, the complexity of the LSTM model was increased (Table 2).
As shown in Figure 10 the original data represents the concentration of NH_4 at the effluent of the WRRF, with 80% in the training and the remaining 20% in the model test. The figure shows the original data used in the test. It represents obvious outliers that the LSTM model will try to detect and eliminate.

Table 2: Initialization of the LSTM model parameters

PARAMETERS	Values
Batch size	128
Epochs	10
Input sequence	10
Loss function parameter	0.001
Hidden layer size	100

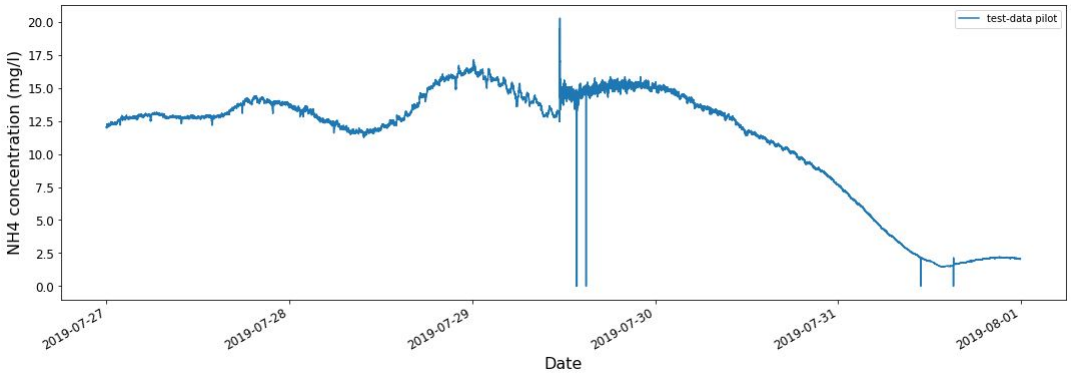

Figure 10: Original test data

Loss function results

In order to minimize the error and optimize the LSTM, 10 epochs were used which means that the model will read the whole dataset 10 times and reduce the error by back-propagation. This process is shown in Figure 11.

```
10%|          | 1/10 [00:18<02:48, 18.73s/it]epoch:   0 loss: 0.0124822184
20%|          | 2/10 [00:37<02:30, 18.87s/it]epoch:   1 loss: 0.0318179131
30%|          | 3/10 [00:56<02:11, 18.84s/it]epoch:   2 loss: 0.0236063432
40%|          | 4/10 [01:15<01:52, 18.68s/it]epoch:   3 loss: 0.0011239738
50%|          | 5/10 [01:33<01:32, 18.51s/it]epoch:   4 loss: 0.0014084985
60%|          | 6/10 [01:50<01:13, 18.30s/it]epoch:   5 loss: 0.0019456027
70%|          | 7/10 [02:09<00:55, 18.42s/it]epoch:   6 loss: 0.0001695003
80%|          | 8/10 [02:27<00:36, 18.23s/it]epoch:   7 loss: 0.0004888157
90%|          | 9/10 [02:44<00:17, 17.99s/it]epoch:   8 loss: 0.0006192051
100%|          | 10/10 [03:01<00:00, 18.18s/it]epoch:   9 loss: 0.0005028247
```

Figure 11: The decrease in error during the 10 epochs

The model was able to detect the outliers and at the same time smoothen the data at all points.

As seen in Figure 12, the data predicted by the model shows an improvement in data quality by detecting and partially eliminating the anomalies.

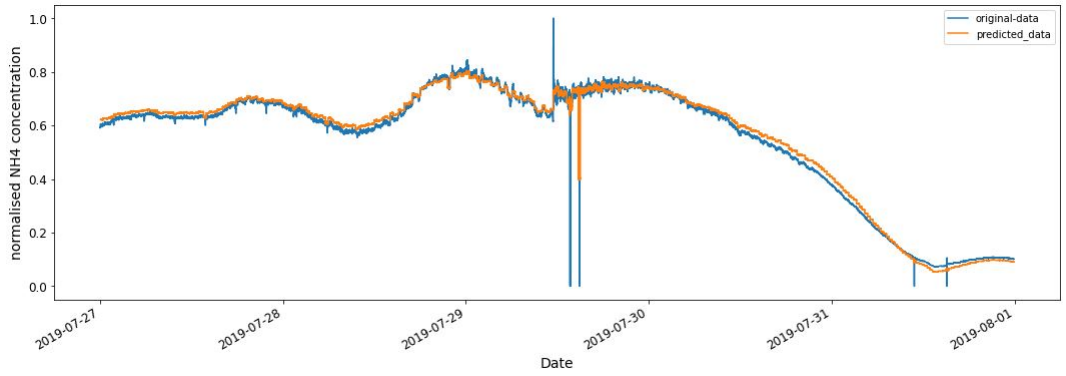

Figure 12: Original Vs predicted data for pilot data after optimization of the LSTM method

Error calculation results

The error calculated on the data set is the difference between predicted and original data. A gaussian distribution (Figure 14) was fit to these errors (Figure 13). The figure shows that most of the errors are around zero with some extremes at some data points. This confirms that the model is predicting very well and identifies the points with big errors as outliers represented at the end of the gaussian distribution.

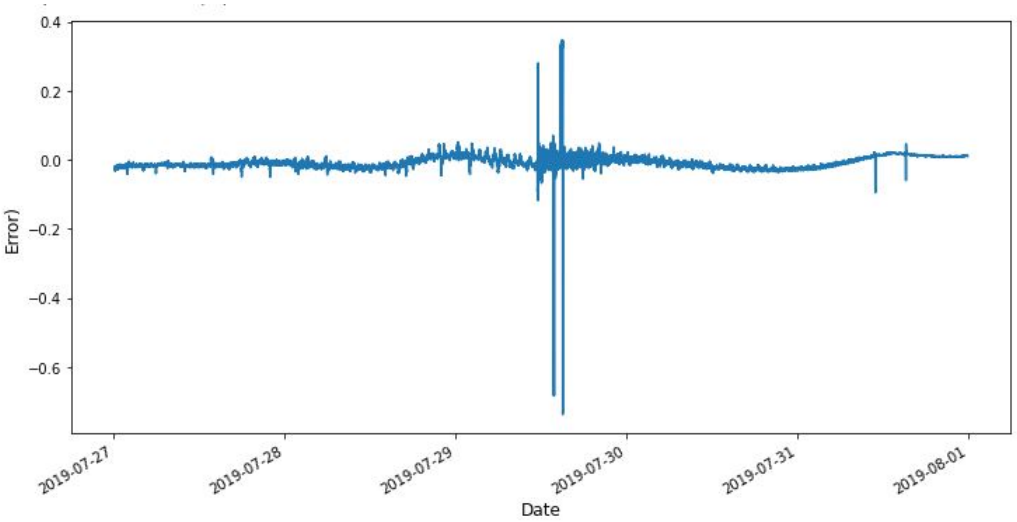

Figure 13: Outlier identification based on error vector: Pilot error vector during model validation

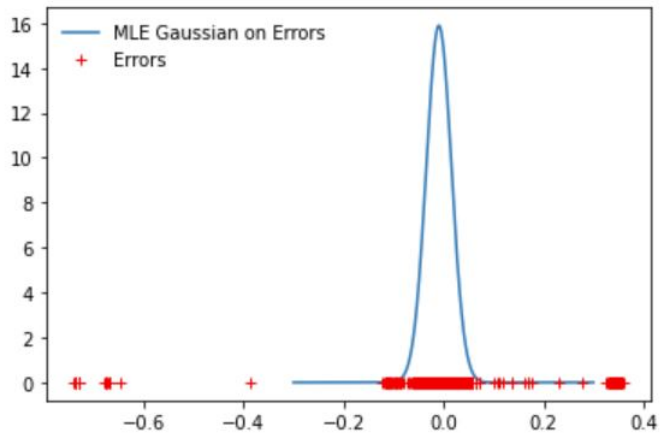

Figure 14: Gaussian distribution fitted over the pilot error vector

Mahalanobis distance

Given the original data, the model reconstruct it and outputs the predicted data (Figure 12). The error vector was then calculated (Figure 13). It shows a big variation in some points of the dataset. At the same time, the Mahalanobis distance was calculated, where it identifies where these outliers are located in the dataset. The green lines in Figure 15 shows where the outliers are which corresponds to the big difference between the original data plotted on top and the reconstructed ones plotted in the middle of the same Figure.

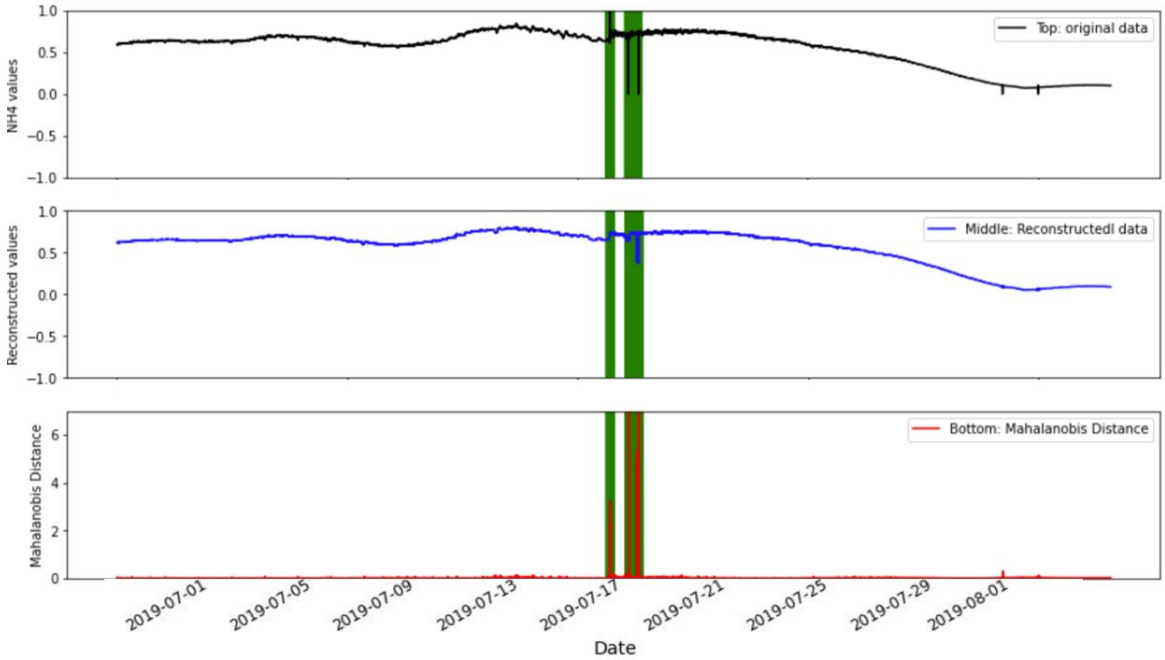

Figure 15: Mahalanobis distance in the bottom with the original data on top and the reconstructed data in the middle where the green lines shows where the outliers are located in the dataset

Testing the model on a different dataset

In order to validate the model performance, the LSTM was tested on data of the same sensor ($NH_4 - N$ concentration at the effluent of the pilot train) obtained under the same operational conditions. The model showed good prediction ability and it was able to detect the outliers in the dataset. The predictions follow the same pattern and the dynamics in the original data. The results of the prediction are compared with the original data in Figure 16. The error vectors and the Mahalanobis distance were calculated as well and are shown in Figures 17 and 18.

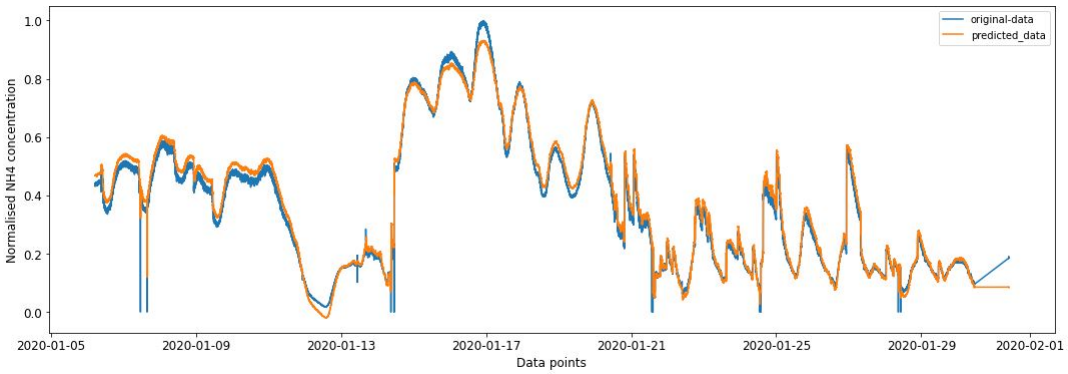

Figure 16: Original VS predicted data of the pilot NH_4 testing dataset

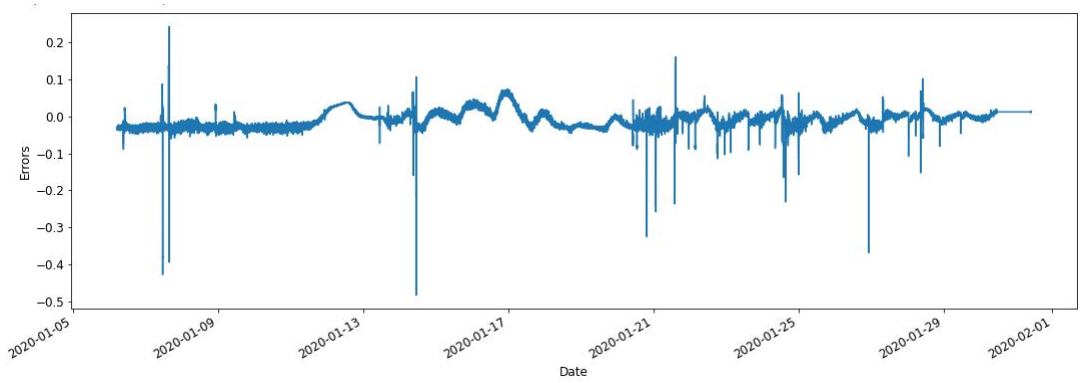

Figure 17: Error vector of the testing data during January 2020

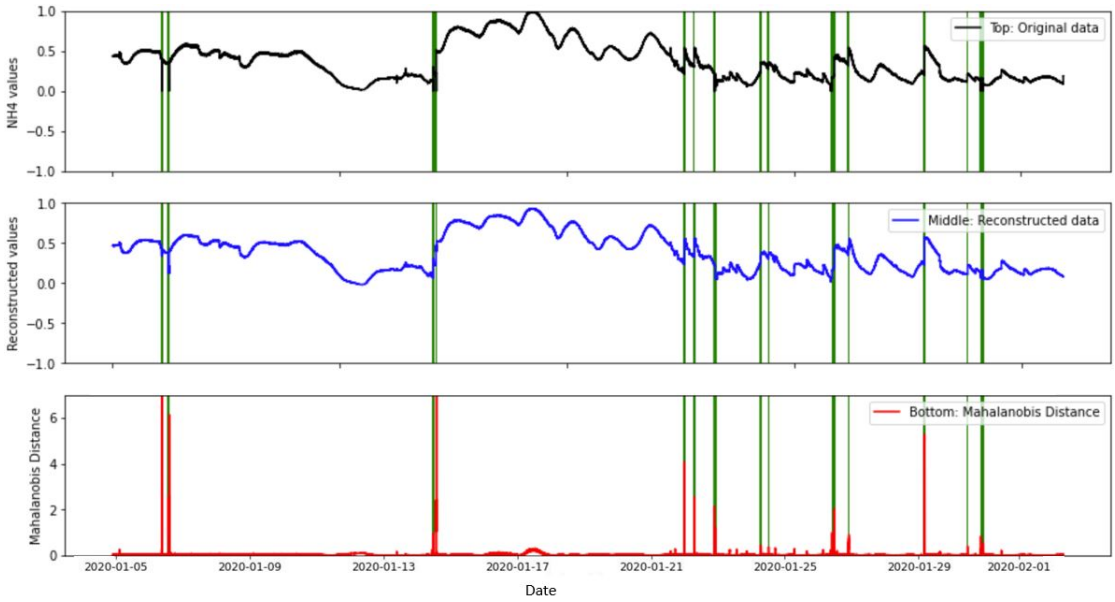

Figure 18: Mahalanobis distance on the testing data during January 2020 with the original data on top and the reconstructed data in the middle where the green lines shows where the outliers are located in the dataset

The error vector shows the big variation in some points of the dataset. These points are outliers. At the same time, the Mahalanobis identified where these outliers are located in the dataset (shown with the green lines in Figure 18).

4.3.2 Co-pilot Results

The same steps cited in the experimental setup were followed here: training and prediction. However, validation of the model on January data was not performed, because of data unavailability.
The same parameters used in the previous section were used because they gave optimal results for prediction and outlier detection.
The model was trained in 10 epochs. The results show a remarkable reduction in the loss function (Figure 19).

```
10%|▉         | 1/10 [00:20<03:00, 20.04s/it]epoch:   0 loss: 0.0131916124
20%|█▊        | 2/10 [00:39<02:39, 19.96s/it]epoch:   1 loss: 0.0111890351
30%|██▋       | 3/10 [00:59<02:19, 19.89s/it]epoch:   2 loss: 0.0004552336
40%|███▌      | 4/10 [01:19<01:58, 19.80s/it]epoch:   3 loss: 0.0001321191
50%|████▌     | 5/10 [01:38<01:38, 19.69s/it]epoch:   4 loss: 0.0003503792
60%|█████▍    | 6/10 [01:58<01:18, 19.70s/it]epoch:   5 loss: 0.0002091282
70%|██████▎   | 7/10 [02:17<00:58, 19.65s/it]epoch:   6 loss: 0.0000685923
80%|███████▏  | 8/10 [02:38<00:39, 19.87s/it]epoch:   7 loss: 0.0000579997
90%|████████  | 9/10 [02:58<00:20, 20.14s/it]epoch:   8 loss: 0.0000395011
100%|█████████| 10/10 [03:19<00:00, 19.90s/it]epoch:   9 loss: 0.0000289818
```

Figure 19: Function loss results when the LSTM model was trained over 10 epochs on co-pilot $NH-4$ data

Figure 20, shows the predicted and original data of the co-pilot NH_4 dataset. Based on these results, it can be stated that the model was able to detect outliers well and the data were smoother. However, for some points, the model was unable to completely eliminate the outliers. This can be explained by the fact that these anomalies persisted for a long time.
The error vector was calculated as in the methodology section and results were around zero except for the data points where the outliers occurred.

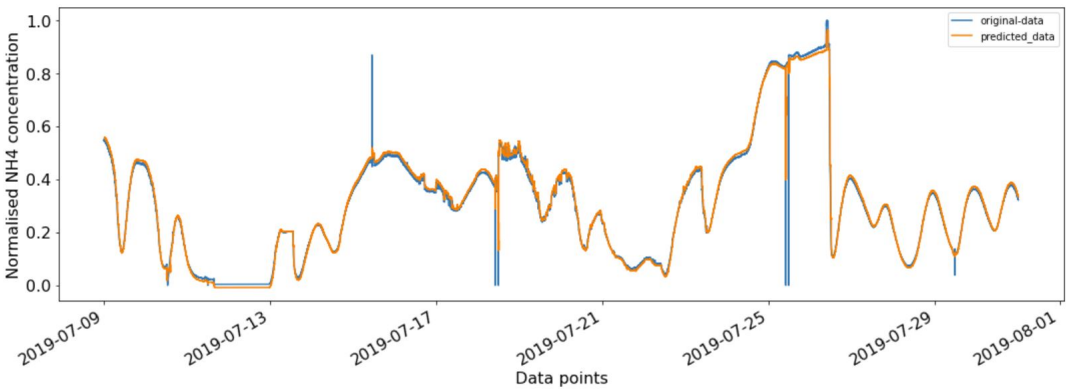

Figure 20: Predicted VS original data of the co-pilot NH_4

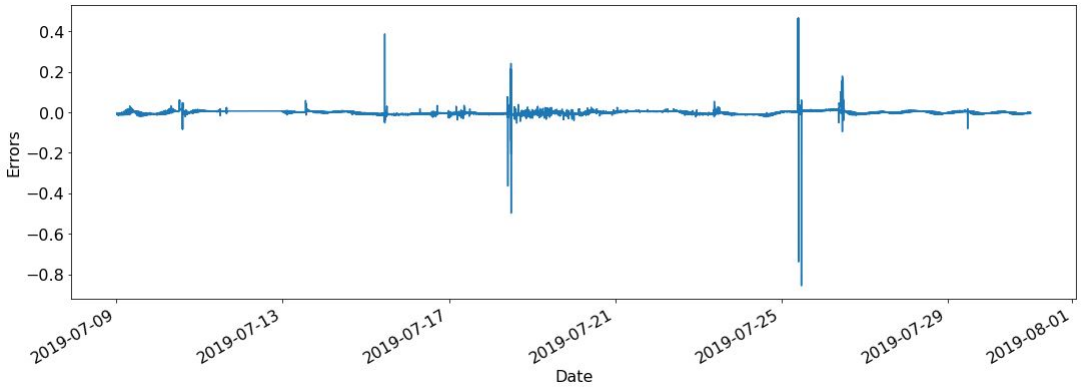

Figure 21: Co-pilot error vector during model validation

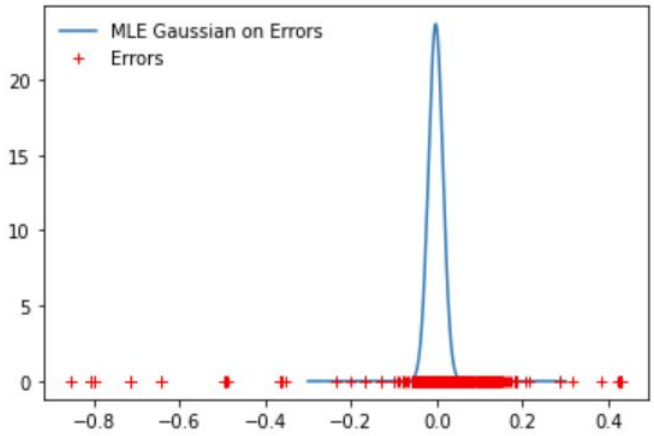

Figure 22: Gaussian distribution fitted over the co-pilot error vector

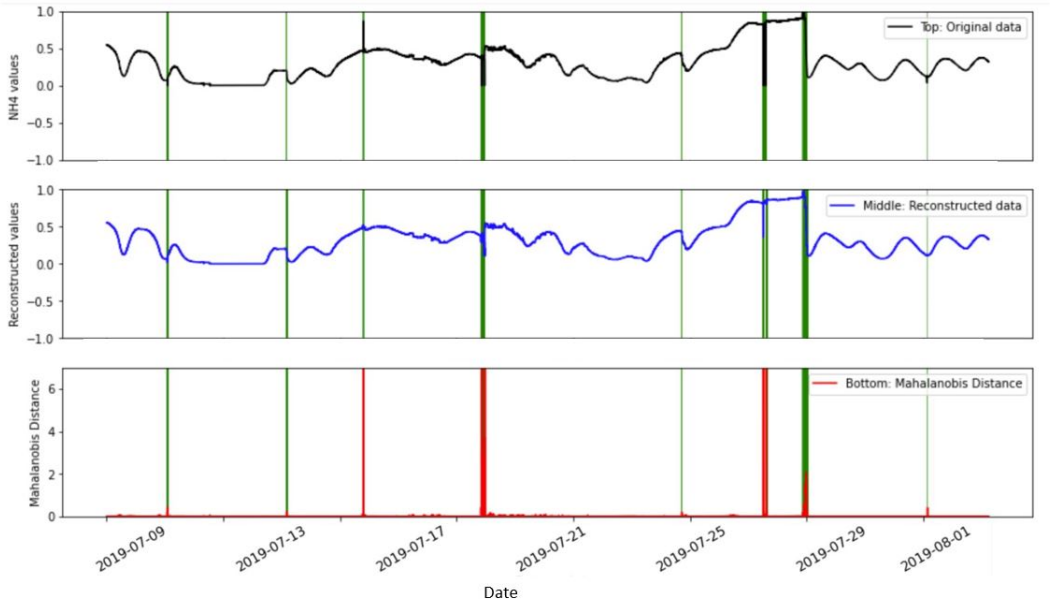

Figure 23: Mahalanobis distance on co-pilot dataset with the original data on top and the reconstructed data by the model in the middle where the green lines shows where the outliers are located in the dataset

The results of the Mahalanobis distance calculation confirmed that it is a good approach to identify outliers. This step is important to decide whether a data point is an outlier or a rare event that would need to be considered in decision making in WRRF.

5 Conclusion

Anomaly detection is the task of detecting abnormal values in a time series given a specific context. WRRF use online sensors for better control and process management, but the sensors used suffer from the difficult conditions caused by immersion in wastewater. These conditions make the tasks of maintaining data quality by sensor cleaning, validation and calibration even more difficult. The measurements lack reliability and precision which explains the need for data treatment and outlier detection. In order to approach this problem, mathematical algorithms are applied for data filtering and anomaly detection, which make the data easy to interpret and use for different objectives.

In this work, the Long Short Term Memory method was used to predict and detect anomalies in WRRF effluent time series data. This demonstrated that the method can successfully detect outliers thanks to its ability to maintain both along and short term memory [Charef et al., 2000]. The resulting errors can be modeled as a Gaussian distribution estimated through maximum likelihood. The LSTM model showed good performance in detecting outliers and smoothimg two different datasets collected under different operational conditions.

However, the model showed some weakness in elimination of outliers when the error persisted for a long period. It was able to reduce the number of outliers but not totally eliminate them.

References

G Aijala and D Lumley. Integrated soft sensor model for flow control. *Water Science and Technology*, 53(4-5):473–482, 2006.

A Charef, A Ghauch, P Baussand, and M Bouyer. Water quality monitoring using a smart sensing system. *Measurement*, 28(3):219–224, 2000.

B Eiteneuer and O Niggemann. LSTM for model-based anomaly detection in cyber-physical systems. In *Proceedings Safe process*, volume 29-31, August 2018.

J Jordan. Introduction to autoencoders, 2018. URL https://www.jeremyjordan.me/autoencoders/.

G Loyel. Long short-term memory: From zero to hero with pytorch, 2019. URL https://blog.floydhub.com/long-short-term-memory-from-zero-to-hero-with-pytorch/?fbclid=IwAR2R8ELk3j3K-8mJRD$_i$em2WsFTPVVguS$_C$e1Rw9ySg$-$7fzVY$_{pgpfL248}$.

P Malhotra, L Vig, G Shroff, and P Agarwal. Long Short Term Memory networks for anomaly detection in time series. In *: Proceedings*, volume 89, pages 89-94. Presses universitaires de Louvain, 2015.

S Prabhakaran. Mahalonobis distance - understanding the math with examples (python), 2015. URL https://www.machinelearningplus.com/statistics/mahalanobis-distance/.

ReNom. "LSTM for Anomaly Detection in Time Series Data", 2016. URL https://www.renom.jp/notebooks/tutorial/time$_s$eries/lstm $-$ anomalydetection/notebook.html?fbclid=IwAR3yhwD34VGXSRPxnTjVjMQGCZv$-$P8MZFwa1xVyZ_ImG$_s$EPECQ54Qfk4g.

R O Strobl and P D Robillard. Network design for water quality monitoring of surface freshwaters: A review. *Journal of Environmental Management*, 87(4):639-648, 2008.