Artificial Neural Networks for Real-Time Data Quality Assurance

Inês Sousa, António Casimiro, José Cecílio

Faculty of Sciences, University of Lisbon, Lisbon, Portugal; email: fc51588@alunos.fc.ul.pt, casim@ciencias.ulisboa.pt, jmcecilio@ciencias.ulisboa.pt

Abstract

Wireless Sensor Networks used in aquatic environments for continuous monitoring are typically subject to physical or environmental factors that create anomalies in collected data. A possible approach to identify and correct these anomalies, hence to improve the quality of data, is to use artificial neural networks, as done by the previously proposed ANNODE (Artificial Neural Network-based Outlier Detection) framework [1].

In this paper we propose ANNODE+, which extends the ANNODE framework by detecting missing data in addition to outliers. We also describe the design and implementation of ANNODE+, implemented in Python to exploit readily available machine learning (ML) tools and libraries, also allowing online processing of incoming measurements. To evaluate the ANNODE+ capabilities, we used a dataset from a sensor deployment in Seixal's bay, Portugal. This dataset includes measurements of water level, temperature and salinity. We observed that our implementation of ANNODE+ performed as intended, being able to detect injected anomalies and successfully correcting them.

Keywords: Neural networks, Environmental monitoring, Sensor networks, Forecasting, Data quality

1 Introduction

Nowadays, maintaining good water quality is important for the aquatic fauna and flora and our life quality. It has become a scarce resource, so it is crucial to monitor it. The Internet of Things (IoT) and Wireless Sensor Networks (WSNs) play an important role to monitor and inspect its quality. WSNs are networks with dedicated sensors that detect specific phenomenons or events. WSNs have been used to remotely monitor many different aquatic environments such as rivers, coasts, lakes and bays [1, 2].

Given that WSNs and their sensors are exposed to physical or environmental factors that often create anomalies in collected data, existing solutions can benefit from platforms for detecting erroneous data or data omissions, to provide the required reliability.

In this work, we propose the ANNODE+ framework, an artificial neural network-based framework for online data quality assurance. Taking inspiration from ANNODE, an outlier detection framework based on Artificial Neural Networks (ANNs) previously proposed by Jesus et al. [1], we report on the on-going design and implementation of a new, more generic and extended framework, usable in multiple settings. With the support of ANNs, the framework considers incoming measurements as time series (e.g., temperature values over time), predicting future values in the series. Each received measurement goes through a set of blocks to determine if it is an outlier, to estimate its quality, and, if considered an outlier, to replace it by a corrected measurement. Our framework was designed for online processing of incoming sensor measurements, and implemented with real-time concerns in mind, to reduce the time taken to process each incoming measurement and avoid arbitrarily large processing times. It offers capabilities to deal with a single sensor (data source) or multiple sensors providing correlated measurements. In fact, the ability to detect outliers can be significantly improved when correlated data sources are available. Measurements can be correlated if different variables have an impact within one another, e.g, salinity levels can change temperature levels. If more than one data source is available, some events can be explained as incidents. For instance, if it is detected a change in water levels, this change will also be detected in other sensors. However, if an event is detected by only one data source, it is most likely that that event is an anomaly.

2 Related Work

There have been many investigations and many different projects using WSNs. However, there is little work on the detection and correction of anomalies in sensor data, which can be frequent when considering deployments affected by harsh environmental conditions. We reviewed previous work done by different authors and the current state-of-the-art in the context of this work for each topic.

Ensuring that sensor data is of good quality, is of great importance to applications that rely on these types of data [3]. As discussed in the survey from Hui Yie Teh et al. [4], artificial intelligence (AI) and machine learning (ML) solutions are now commonly used to ensure data quality despite sensor and network failures. In [4], Hui Yie Teh et al. made a literature search for keywords related to these topics. Table 1 shows the most frequent types of sensor data errors mentioned in papers and the number of times they were mentioned.

From these values, it is clear that outlier and missing data errors are those that gather comparatively more attention in

Table 1: Most common types of errors in sensor data [4] and number of papers mentioning them.

Type of error	Total	Type of error	Total
Outliers	32	Noise	8
Missing Data	16	Constant value	7
Bias	12	Uncertainty	6
Drift	12	Stuck-at-zero	6

the literature, perhaps because they are prominent in real deployments. These are the two types of errors that we address with ANNODE+.

As previously stated, ML methods are commonly used to detect anomalies in sensor data. Several different methods, including supervised and unsupervised ones, can be used to detect anomalies based on past data. In [4], an analysis of the most common ML methods for error detection was also done, being the three most prominent ones the Principal Component Analysis (PCA), Artificial Neural Networks (ANNs) and Ensemble Classifiers. Our framework uses ANNs for this process, more specifically MultiLayer Perceptron Neural Networks (MLP).

In addition to detecting outliers, ANNODE+ also detects missing data. This is done only for periodic data, assuming that the period is known.

In [5], a similar experience was made in Aveiro, Portugal. The authors used and adapted a custom-deployment based forecasting platform to the Portuguese coast. This allowed to create numerical models to provide forecasts of sea level variations, currents, temperatures, etc. However, they also recognize that further research and solutions to deal with data errors are needed when considering sensor deployments in harsh environments. Furthermore, they refer to the possibility of exploiting existing temporal and spatial correlations in sensor data.

To understand the relevance of correlations between data from different sensors, several ML methods were considered in [6] to forecast long-term and short-term water demand when considering variables such as rain, hour of the day and air temperature. It was possible to conclude that using multiple correlated variables helps improving the accuracy of forecasts, with some variables having more impact on the achieved results.

In summary, the ANNODE+ framework is designed to detect and correct the most common sensor data errors. Additionally, the framework includes mechanisms, based on timers, to deal with missing data and exploit data correlations that may help with the predictive model and the detection of anomalies.

3 ANNODE+ Architecture

The framework's architecture is illustrated in figures 1 and 2. It is composed of two blocks: the training and execution blocks. The training block corresponds to an offline execution for models' training. This training step is supported by a dataset containing sufficient information to represent all the main characteristics and dynamics of the variable being modelled (e.g., represent the seasonality present in the real phenomenon). The user must prepare a configuration file with all the training requirements, such as the number of sensors, their characteristics (e.g., sampling period), and data characteristics (e.g., representative period).

The characteristics of the MultiLayer Perceptron (MLP) neural networks trained in the framework are the same as those proposed in the ANNODE framework, which are described in [1], and consist of two hidden layers with 20 neurons in the first layer and 15 in the second, using a hyperbolic tangent sigmoid (tansig) as the activation function.

Models are trained using datasets that must be provided by the user, which must have been previously collected and must include only data considered correct. Annotating or cleaning training data is a typical requirement when using ML methods. When data from multiple correlated sensors is available, several models are created, corresponding to different combinations of sensors. In fact, to predict the next measurement of a sensor, it is possible to use a model that was trained using only data from that sensor (exploiting temporal correlation), or using a model combining data from multiple neighbor sensors (exploiting spatial correlation).

After training the ANN models, the framework is ready to run (online execution). The Execution block follows a multi-service implementation approach. Currently, there are three primary services: Communication, Omission detection and Processing. The Communication service is responsible for receiving sensor data, for identifying its source sensor, and for inserting these data in that sensor measurements queue, which is shared with the Omission detection service.

The Omission detection service is a multi-thread service with a timer thread for each sensor from which periodic data is to be received. Considering that each sensor s sends a new measurement with period P_s, a corresponding timer is set to expire after $P_s + \delta$, where δ corresponds to the jitter assumed for the measurement reception instant. If no measurement from that sensor is received before $P_s + \delta$, an omission is detected and a special value (like a NaN) is inserted in the sensor measurements queue. By considering the channel jitter, the probability of wrongly detecting an omission is reduced to the probability of considering a wrong (too small) jitter.

Lastly, there is the Processing service. This service is triggered by new incoming measurements, stored in each sensor's measurements queue. However, actual processing only starts after a certain number of measurements have been received, covering an entire representative period of the physical process being monitored (e.g., 12 hours for sea water level). When this condition is fulfilled, actual processing is executed for every new incoming measurement. Firstly, measurements are temporally aligned, to match the alignments used during model training. Then, input vectors are built from the stored and aligned measurements, to be fed to the relevant ANN prediction models. If correlated sensors are available, then the following models (which have been previously trained) are used to generate forecasts:

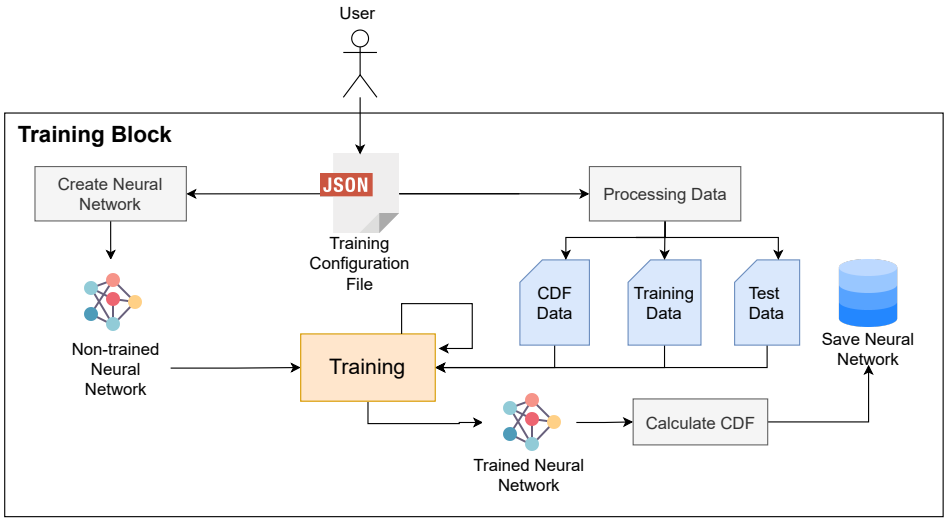

Figure 1: ANNODE+ Architecture - Training Block

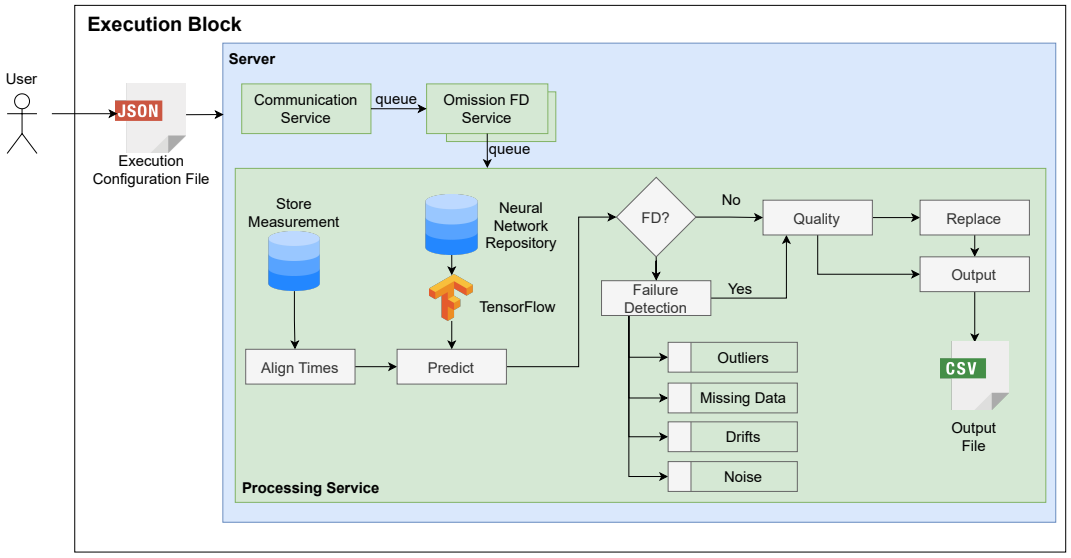

Figure 2: ANNODE+ Architecture - Execution Block

1. A model that only uses data from the target sensor whose measurement is being processed (temporal correlation);
2. A model that uses data from the target sensor and its neighbours (temporal and spatial correlation);
3. A model that uses data only from the target sensor neighbours (spatial correlation only).

The benefit of using these multiple forecasts is to be able to distinguish real environmental events (even if they look like outliers) from real outliers (only affecting the target sensor, but not the neighbor ones). It is then possible to check the correctness of the received measurement, comparing it with the generated forecasts. In fact, given that sensor data can be affected by different kinds of errors, it is at this point of the processing chain that it is possible to determine if some failure may have happened, leading to those different errors. While ANNODE+ is only detecting outliers, it may be extended in future work with new failure detection blocks to detect data drifts and noise. When some failure is detected (either missing data or outlier), then the received measurement (or the special NaN value) is replaced by an average of the calculated forecasts.

In addition to detecting failures, it is also possible to calculate the quality of the received measurement. When the difference between the received measurement and the forecasts provided by the models is small, then the quality is high.

4 Results

To evaluate our ANNODE+ implementation, we used a dataset with temperature measurements collected from a sensor in the Seixal Bay, in Portugal. We divided this dataset in two parts, containing data from different temporal periods. One part was used to train a single model (in this initial evaluation we only considered one model, exploiting temporal correlations) and the other was used to test the framework. Training a single model took about one full day on our hardware with 16GB of RAM and an AMD Ryzen 5 3500X CPU. Given our objective of checking the ability of the framework

to detect outliers and missing values, we randomly injected these anomalies in the second part of the dataset, by changing some measurements and by removing some of them from the temporal series.

To emulate a real online usage of the framework, we built a framework client that plays the role of a sensor and sends a new measurement (taken from the dataset) to the framework with a period of 500ms (except when injecting an omission).

We observed very positive results, with the framework replacing the injected outliers with their respective predictions. As stated before, when training is done with a sufficiently large and representative dataset, the obtained predictions are very accurate. We were also able to detect omissions, as expected, and correctly replacing the missing measurements with the values predicted by the ANN model.

Given the need to execute the full processing chain before the arrival of a new measurement, we also measured the response time of the framework, from the moment a new measurement arrives until it is fully processed, to assess the achievable performance and potential capacity of the framework in handling incoming data. The ability to ensure a bounded execution time necessarily depends on the underlying execution environment (namely the operating system being used), but having an idea of the time needed to process a single measurement is already important to make sure that the framework is not overloaded. In our experiments we also measured the CPU load, and we considered the execution of the framework with and without background load. The background load was created by running the framework for model training.

Table 2: Run-time cost of the algorithm for one measurement.

Without load	Total	With load	Total
Max time (ms)	165	Max time (ms)	153
Avg time (ms)	52	Avg time (ms)	54
Best time (ms)	46	Best time (ms)	49
Avg CPU load	11%	Avg CPU load	42%

Table 2 provides the collected performance measurements. The results show that in this case it would be possible to fully process each new measurement in about 50ms (in average), even considering a loaded CPU. For most applications performing environmental monitoring, this kind of performance is sufficiently good to allow the framework to be used for online processing of incoming measurements, even if measurements from several sensors have to be processed.

5 Conclusion and Future Work

The preliminary results show very positive outcomes. However, we aim to complete and test the implementation of ANN-ODE+ to handle more than one data source, and improve the calculation of the measurement quality using correlations between sensors. We also plan on improving the training block by automating some parts of the training process.

Moving away from applications in the environmental monitoring area, we plan to use the framework in a use case of arc detection in DC distribution cabinets, in the context of the VEDLIoT EU project (https://vedliot.eu). In this case there is a stream of sensor data being continually produced and sent in batches to the framework, while timing requirements for the detection of arcs (which will create data that will look like outlier in relation to normal data) are very stringent, in the order of a few milliseconds. A different execution platform will be required, with more resources and making it possible to satisfy timeliness requirements.

Acknowledgments

This work was supported by FCT through funding of the AQUAMON project (ref. PTDC/CCI-COM/30142/2017) and the LASIGE Research Unit (ref. UIDB/00408/2020 and ref. UIDP/00408/2020), and by the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197 (VEDLIoT project).

References

[1] G. Jesus, A. Casimiro, and A. Oliveira, "Using machine learning for dependable outlier detection in environmental monitoring systems," *ACM Trans. Cyber-Phys. Syst.*, vol. 5, jul 2021.

[2] B. O'Flyrm, R. Martinez, J. Cleary, C. Slater, F. Regan, D. Diamond, and H. Murphy, "Smartcoast: A wireless sensor network for water quality monitoring," pp. 815 – 816, 11 2007.

[3] G. Jesus, A. Casimiro, and A. Oliveira, "A survey on data quality for dependable monitoring in wireless sensor networks," *Sensors*, vol. 17, no. 9, p. 2010, 2017.

[4] H. Teh, A. Kempa-Liehr, and K. Wang, "Sensor data quality: a systematic review," *Journal of Big Data*, vol. 7, 02 2020.

[5] J. Gomes, M. Rodrigues, A. Azevedo, G. Jesus, J. Rogeiro, and A. Oliveira, "Managing a coastal sensors network in a nowcast-forecast information system," in *2013 Eighth International Conference on Broadband and Wireless Computing, Communication and Applications*, pp. 518–523, IEEE, 2013.

[6] B. Brentan, G. Meirelles, M. Herrera, E. Luvizotto Jr, and J. Izquierdo, "Correlation analysis of water demand and predictive variables for short-term forecasting models," *Mathematical Problems in Engineering*, vol. 2017, 12 2017.