ANNODE+: Outlier and Omission detection framework based on Machine Learning

Inês Sousa, António Casimiro, José Cecílio

Faculty of Sciences of the University of Lisbon, Portugal
fc51588@alunos.fc.ul.pt, casim@fc.ul.pt, jmcecilio@fc.ul.pt

Abstract. Wireless Sensor Networks used in aquatic environments for continuous monitoring are constantly being affected by physical or environmental factors that can cause anomalies in collected data. In this work, we propose ANNODE+, an improved version of ANNODE, an outlier detection framework based on Artificial Neural Networks (ANNs). This new version comprises improved methods for detecting anomalies from one or more data sources and calculation of accurate forecasts in order to correct erroneous measurements. From the experiments, we conclude that the framework is reliable in our application context, allowing us to identify outliers and missing data. When needed, ANNODE+ also allows for replacing missing or erroneous data with the forecasted value.

Keywords: Neural Networks, Water Quality Monitoring, Sensor Networks, Real-time Forecasting, Data Quality

1 Introduction

Water can become a scarce resource. It is crucial to monitor it and ensure its quality. The Internet of Things (IoT) and Wireless Sensor Networks (WSNs) play a critical role to monitor 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 and bays [4] [5] [7] [8] [10]. WSNs have the advantage of providing real-time data, which is vital to alert emergency systems or the authorities in case of atypical situations.

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. It is based on ANNODE, an outlier detection framework based on Artificial Neural Networks (ANNs) previously proposed by Jesus et al [7].

This new version of the framework aims to be more generic, extending previous work with additions of new failure (outliers and missing data) detection methods, as well as a new implementation using more efficient libraries which allows to train and run ML models efficiently.

ANNODE+ was designed for online processing of incoming sensor measurements, and implemented with real-time concerns in mind, in order to reduce the time taken to process each in-coming measurement and to avoid 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 event is an anomaly.

2 Related Work

ANNODE+ is being developed in the context of the AQUAMON project [4]. AQUAMON has the objective to develop a dependable platform based on WSNs to monitor aquatic environments. This project offers mechanisms to deliver a timely notification to managers and/or authorities and users of the system with the help of real-time monitoring coming from WSNs.

The ANNODE framework[7] was developed to receive data, and process it by detecting outliers, consequently replacing those values with predictions according to previously received data. In [7], authors used neural networks to predict measurements based on entry vectors composed of received data. Data from the Center for Coastal Margin Observation & Prediction [2] was used. In this work, data collected from the Seixal Bay is used.

The previous version of the framework [7], [9] is able to find and correct outliers. However, it was designed to run in an offline environment. Thus, an online logic is needed, where data must be received and processed in order to detect and correct anomalies, in real time.

The ANNODE architecture is relatively simple. It is divided into two blocks:

- **Model training** - Trains neural networks (models) with previous data from the same sensor. The training needs to occur before the usage of the framework;
- **Utilization of models in real-time** - Uses previously trained models in real-time and processes data.

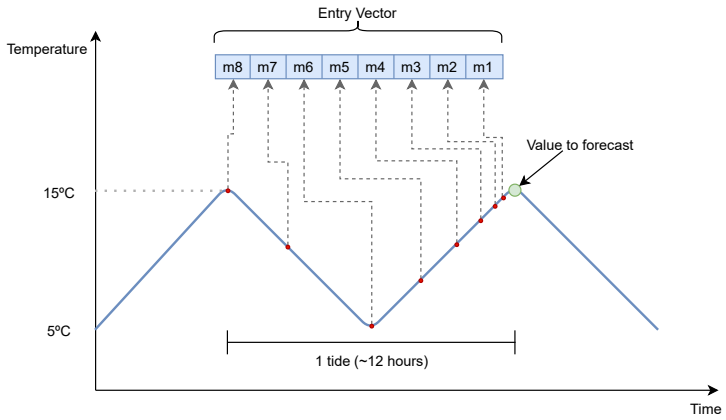
To train neural networks and compare measurements to models, the ANNODE uses entry vectors. Entry vectors are lists of values used not only to train neural networks but also to calculate forecasts. These vectors are created based on the values that are received. The received values from multiple sensors need to be aligned to create correct entry vectors. Otherwise, it could cause problems during the execution of the ANNODE framework. For instance, in an example of four sensors, the target sensor starts its measurements at 09:00 a.m, and its neighbours start their measurements at 09:10, 09:15 and 09:30. The values

can only be aligned after 09:30 because that is the only time the four sensors have the same number of measurements in common. By aligning the times, the timestamps of these measurements will be rewritten to 09:30 a.m.

After the measurements' alignment, the entry vectors can be safely deployed. Its creation uses a given number of measurements, specified by the user.

Figure 1 shows an example of these entry vectors.

Fig. 1: Example of the entry vector's construction using the target sensor [9]



In ANNODE each entry vector is made by using previous measurements that are chosen in an exponential approach. In other words, it is first calculated how many measurements should be considered on those vectors. Assuming that input data is periodic, the number of samples must be enough to represent a full tide period. That number sets a maximum index in which cannot be surpassed. It is then exponentially determined the chosen N indexes from zero to the maximum index number. Finally, the entry vector is completed with the values from each chosen index, making it a vector with data equivalent to the last twelve hours of received measurements.

There has been many investigations and many different projects using Wireless Sensor Networks. However, there is little information on how sensors detect and correct anomalies caused by harsh conditions. Next we review previous work done concerning sensor data quality and dependable monitoring systems.

2.1 Sensor Data Quality

Types of errors in sensor data Hui Yie Teh et al. in [11] searched for keywords in a database with multiple papers and articles. Table 1 shows the most frequent types of error found in papers and the number of papers that address the respective error.

Table 1: Most Common Types of Error in Sensor Data [11]

Type of error	**Total**
Outliers	32
Missing Data	16
Bias	12
Drift	12
Noise	8

The most frequent errors detected in sensor data are outliers, missing data, bias and drifts. These errors can be caused by changes in the environment, device malfunction or obstructions in the communication between nodes. This is also the case for missing data. These cases can then lead to incorrect decision making when creating predictions with the framework.

Moreover, according to [6] these are the possible errors that can occur in individual sensor measurements:

- **Random errors** - Arbitrary errors that can occur due to physical structure of the sensor;
- **Systematic errors** - These can be of three types: calibration errors, loading errors and environmental errors;
- **Non-systematic reading errors** - These errors happen when a physical event takes place.

The previous errors correspond to errors in measurements obtained by sensors. However, these can always have failures and be broken by significant events, consequently recording inaccurate data.

Similarly to the types of error, the authors of [11] also searched for the most common methods to detect the types of errors in Table 1. Table 2 shows different methods for error detection. The total represents the number of methods based on these error detection methods.

Regardless of all these different methods, there is still space for time-based methods to detect missing data. For instance, if a sensor always sends data every five minutes, new data is expected within that time frame. So if none comes, it can be considered as an error.

ANNODE uses artificial intelligence and machine learning methods to detect anomalies in sensor data. Many techniques, including supervised and unsupervised methods, can detect anomalies based on past data.

An ANN works almost the same way as a human brain. The main processing power is the neuron. It receives data and passes it to other neurons that process it until it reaches the output's last neuron. There are three components to an artificial neural network: the input layer, hidden/occult layers and the output layer. There are many different types of ANNs, but they are all mainly used for pattern recognition.

Table 2: Most Common Methods for Error Detection in Sensor Data [11]

Method	**Errors addressed**	**Total**
Principal Component Analysis	Outliers, bias, drift, stuck-at-zero	7
Artificial Neural Network	Outliers, bias, drift, constant values, noise, stuck-at-zero, uncertainty	6
Ensemble Classifiers	Outliers, drift, constant values, noise, uncertainty	4
Support Vector Machine	Outliers	2
Clustering	Outliers	2
Ontology / knowledge-based systems	Uncertainty, missing data	2
Univariate autoregressive models	Outliers	1
Statistical Generative Models	Outliers	1
Grey Prediction Model	Outliers, noise, constant values	1
Particle Filtering	Bias, scaling	1
Association Rule Mining	Outliers	1
Bayesian Network	Outliers, noise	1
Euclidean Distance	Outliers	1
Hybrid Methods	Outliers, drift, noise	1

Methods to detect and correct anomalies In the same way that there are methods that detect anomalies, the following techniques are more complete considering that they can detect and correct anomalies.

The next list presents three proposed methods by Hui Yie Teh et al. in [11] to detect and correct anomalies in sensor data:

- **Principal Component Analysis (PCA)**;
- **Artifitial Neural Network (ANN)**;
- **Bayesian Network**.

Our framework uses ANNs and the PCA model to detect and correct anomalies. Just like the methods for anomaly detection, these also build models to compare with observed values. Suppose the value is significantly different from the model. In that case, it is considered an anomaly and it is then replaced with a prediction or an estimated value from the model in that instant.

2.2 Dependable Monitoring Systems

There have been many experiments with dependable monitoring systems over the years, but few in aquatic environments.

From those few, we have the project SmartCoast [8]. It consists of a wireless sensor network for water quality monitoring in Cork, Ireland. This project investigates the water's temperature, phosphate, dissolved oxygen, conductivity, pH, turbidity and water level variables, so it is a multi-sensor project. These parameters are almost universal to other projects, being the most required variables for these experiments. The SmartCoast project uses the ZigBee protocol for low power consumption. Because this was the start of projects with water quality monitoring, data quality was lacking.

Another project is presented in [5]. It was made in Aveiro, Portugal by LNEC. 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. The authors explain that investigation about applications in harsh environments is minimal. There is a need to explore what happens to sensors and their data in this type of environment.

This motivates the work that we are proposing, as saline waters have significant changes to objects because of their salinity levels. It can also affect data. This experiment was the beginning of developing a real-time system for water monitoring. The platform adapted by LNEC was then implemented in the Tagus estuary [10] with the same sensor architecture and models.

3 ANNODE+ architecture

This section exposes the main techniques that are being used by the ANNODE framework. The first phase of the work that we did consisted of understanding the framework and its details, such as the data requirements, configurations and used techniques. Then, in the second phase, new features were implemented to enhance the framework's capabilities.

The ANNODE+ architecture has not changed the training block from the previous version (ANNODE). A user fills out one configuration file to process the input data sets and to start the training phase. The data sets need to be processed first as they need to be separated into three data sets, one for training, one for testing the neural network, and one for calculating the neural network's Empirical Cumulative Distribution Function (ECDF).

The three data sets are used during training in order to create trained neural networks that are then stored. The training process involves 2000 epochs (number of times a network passes backwards and forwards) and is repeated 10 times, resulting in 10 trained neural networks with different loss functions. In the end, the best neural network, the one with an inferior loss function is chosen to forecast measurements.

ANNODE+ uses artificial neural networks to identify and correct anomalies. The neural networks are used to train models that later help on detecting faulty measurements. For the calculation of forecasts, a deep learning API was used called Keras [1], built on top of Tensor Flow [3]. Tensor Flow allows the creation of neural networks and statistical models. To calculate the error between pre-

Fig. 2: ANNODE+ Architecture - Training Block

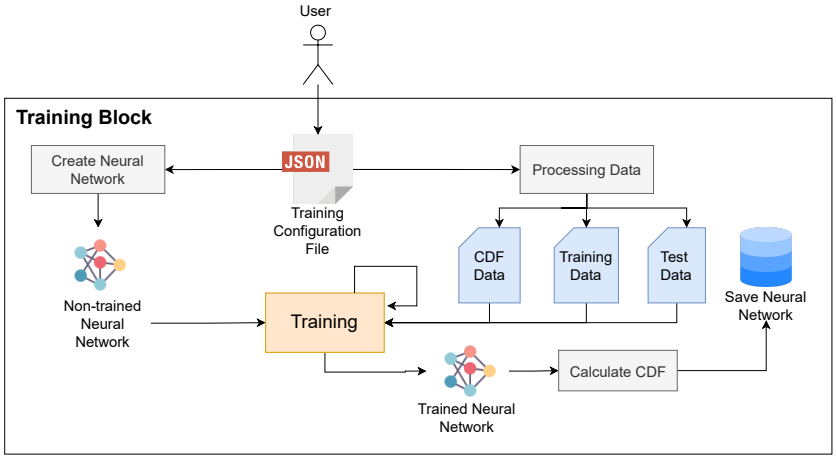
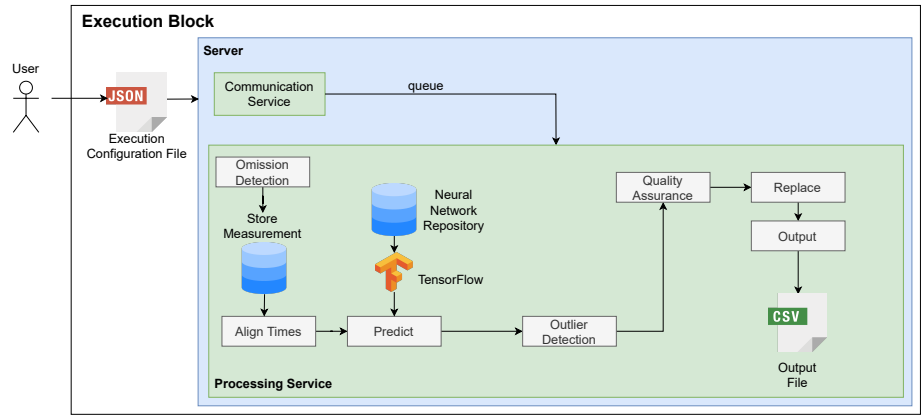


Fig. 3: ANNODE+ Architecture - Execution Block



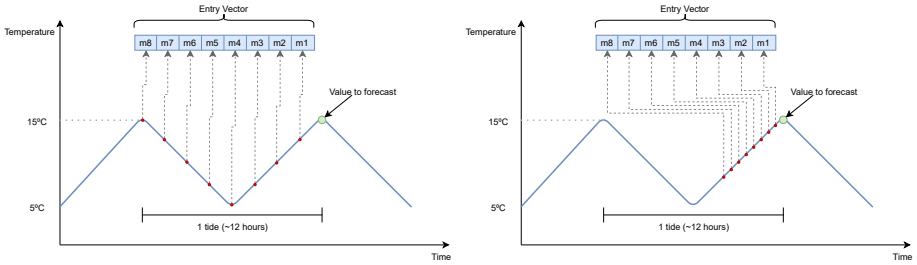
dictions and measurements, the framework applies the squared prediction error formula, which is useful for the determination of the ECDF.

Forecasts are calculated depending on entry vectors and its values. These values are selected from the received data in an exponential approach. It was also implemented two new approaches for entry vectors:

- **Linear approach** - with this approach, the algorithm would get ten values in a linear way;
- **Last ten approach** - with this approach, the algorithm would get the last ten received values.

In the linear approach, the entry vector is filled with values spaced between themselves linearly. The entry vector should consist of values corresponding to twelve hours of measurements. In Figure 4a we have a value to predict in green, where the entry vector is complete with values equally spaced.

In the last ten approach, the entry vector is completed with the previous ten measurements. These values are the ones immediately before the value to forecast. In Figure 4b there is an example of how this algorithm works.



(a) Linear Approach (b) Last Ten Approach

Fig. 4: Selection of values for the entry vectors

Another method that was implemented is related to detecting omission failures (missing data). We created a new thread that only detects this kind of error, checking each measurement's timestamp. We defined a *jitter* that is added to the period in which a measurement is scheduled to be received. If no data is received within that period, an omission is detected and filled with a senseless value (NaN) to be later replaced by its prediction in that timestamp.

ANNODE+ execution is based on a configuration file, which provides more control and flexibility over the framework. This file would provide the necessary information for the framework's execution without it being hard-coded. This way, the user can also change the settings and configurations of the framework and provide information on all available sensors.

With all these changes, the ANNODE framework was extended, resulting in a new version called ANNODE+. The new architecture has two different services in the execution block: Communication Service and Processing Service. The Communication Service is responsible for receiving data from a node and inserting it into a queue to which the Processing Service has access. In the Processing Service, the received measurement is added into an entity which stores all measurements. Here, the framework checks if possible data omissions could occur while the sensors send their data. If an omission is detected, the initial queue is filled with a senseless value to be later dealt with in the prediction and quality block. Then, when enough data covers an entire tide, the times are aligned as previously explained. If all requirements are met, meaning there are enough values, a forecast is calculated according to the entry vector created

with previously received values. A filter then decides if a value is erroneous or not according to some conditions, specifically the calculated squared prediction error. If the value is faulty, it is replaced with the respective forecast. In the end, we have a set of correct values without anomalies.

4 Results

After adopting the framework, we started implementing it with our data set from the Seixal. We separated it into three different data sets containing data from different months. These data sets were used to train models and simulate a real-time occasion with the framework detecting outliers and correcting the respective values. Training took almost a full day for our data with a hardware of 16GB of RAM and an AMD Ryzen 5 3500X CPU. In order to check if the framework would be able to detect anomalies, we injected multiple anomalies into the data set, 5 outliers and 2 periods of missing data. To emulate sensors, we implemented a simulator which would send each measurement to our server in order to process them.

We observed very promising results, with the framework replacing faulty values with their respective forecasts. Since we used the previous two months worth of data to train models, the calculation of forecasts were very accurate. The framework was also able to detect missing data in data sets as well as forecast values during that time thanks to the models that were trained beforehand.

Fig. 5: Framework's outcome with data from the Seixal bay

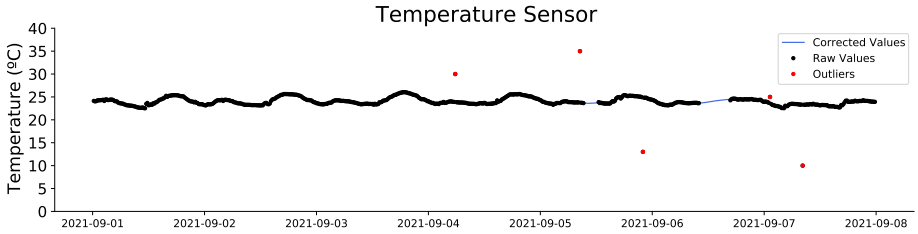


Figure 5 shows the results we were able to get with a data set of a whole week of raw measurements. As we can see, the framework correctly detected all 5 outliers and corrected those values with their respective forecast. The same happened in periods where there was no data (missing data). The two periods of data omission were detected and corrected with forecasts.

When testing the framework with multiple sensors, we were able to get similar results. Although this is still a work in progress, the framework correlates measurements from all sensors, and thus with more fidelity, determine an anomaly.

5 Conclusion

In conclusion, the results show very positive outcomes, and we are on the right track to having a reliable framework. Some tweaks can be made to improve the framework such as enhancing the framework's performance for more than one sensor, and improve the calculation of the measurement quality using correlations between other sensors. There are also a few metrics that need to be considered in the quality assurance service, mainly correlations between different variables, such as salinity. Also, in a network of three sensors, if one has a value that is significantly different from the others, that measurement is considered faulty.

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. Keras, https://keras.io/
2. Saturn observation network: Endurance stations, http://www.stccmop.org/datamart/observation$_n$*etwork*
3. Tensor flow, https://www.tensorflow.org/
4. Casimiro, A., Cecılio, J., Ferreira, P., Oliveira, A., Freire, P., Rodrigues, M., Almeida, L.: Aquamon–a dependable monitoring platform based on wireless sensor networks for water environments. In: Proceedings of the 38th International Conference on Computer Safety, Reliability and Security (2019)
5. Gomes, J., Rodrigues, M., Azevedo, A., Jesus, G., Rogeiro, J., Oliveira, A.: 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. Jesus, G., Casimiro, A., Oliveira, A.: A survey on data quality for dependable monitoring in wireless sensor networks. Sensors **17**(9), 2010 (2017)
7. Jesus, G., Casimiro, A., Oliveira, A.: Dependable outlier detection in harsh environments monitoring systems. In: Gallina, B., Skavhaug, A., Schoitsch, E., Bitsch, F. (eds.) Computer Safety, Reliability, and Security. pp. 224–233. Springer International Publishing, Cham (2018)
8. O'Flyrm, B., Martinez, R., Cleary, J., Slater, C., Regan, F., Diamond, D., Murphy, H.: Smartcoast: A wireless sensor network for water quality monitoring. pp. 815 – 816 (11 2007). https://doi.org/10.1109/LCN.2007.34
9. Penim, J.: Implementação de soluções para confiabilidade de dados em redes de sensores sem fios (2020)
10. Rodrigues, M., Costa, J., Jesus, G., Fortunato, A., Rogeiro, J., Gomes, J., Oliveira, A., David, L.: Application of an estuarine and coastal nowcast-forecast information system to the tagus estuary. Proceedings of the 6th SCACR, Lisbon (2013)
11. Teh, H., Kempa-Liehr, A., Wang, K.: Sensor data quality: a systematic review. Journal of Big Data **7** (02 2020). https://doi.org/10.1186/s40537-020-0285-1