Water Contamination Detection With Artificial Neural Networks

Rikard Fridsén Skogsberg and Martin Gelin

***Abstract*—Drinking water is one of our most important resources, so the ability to reliably monitor harmful contaminations in our water distribution network is vital. In order to minimize false alarms for water monitoring, while keeping a high sensitivity, a machine learning approach was evaluated in this project. Measurement data captured with a new kind of sensor, an electronic tongue, was provided by Linköping university. The solution was an artificial neural network, in the structure of an Autoencoder, which could learn the dynamic behaviour of natural deviations and with a false alarm rate of approximately one false alarm per week. This was done by evaluating the data and assembling an input structure to account for daily cyclic phenomena, which then was used to train the neural network. The solution could detect anomalies as small as 1.5% by comparing the input with the reconstructed vector, and raise an alarm. In conclusion, an Autoencoder is a viable method for detecting anomalies in water quality.**

***Sammanfattning*—Drickvatten är en av våra mest värdefulla tillgångar, det är därför mycket viktigt att det finns sätt att pålitligt övervaka om dricksvattennätet blivit förorenat. För att kunna minimera antalet falsklarm och samtidigt ha hög känslighet mot dessa föroreningar undersöktes och implementerades en lösning med maskininlärningsalgoritmer. Mätdata tillhandahölls av Linköpings universitet och kom från en ny sensor kallad elektronisk tunga. Lösningen var ett artificiellt neuralt nätverk i form av en Autoencoder, som kunde lära sig det dynamiska beteende som ofarliga avvikelser utgjorde. Detta gav en lösning som i medel gav ett falsklarm per sju dagar. Detta gjordes genom att utvärdera rådata och konstruera en struktur på indata som tar hänsyn till dygnsbunda naturliga fenomen. Denna struktur användes sedan för att träna det neurala nätverket. Lösningen kunde upptäcka fel ner till 1.5% genom att jämföra indata med den rekonstruerade vektorn, och på så sätt ge ett alarm.**

***Index Terms*—Water distribution network, Anomaly detection, Artificial neural networks, Autoencoder, Machine Learning, Time-series**

Supervisor: Rong Du

TRITA number: TRITA-EECS-EX-2020:154

I. Introduction

The quality of our drinking water is of utmost importance and thus requires robust systems to continually monitor its state, which is done by interpreting data collected by a network of sensors. The causes of a compromised and contaminated water supply can occur because of structural failures in pipes, human error and water treatment errors. These three causes are unintentional but since the water distribution networks are considered key critical infrastructure, it may also be subject to intentional sabotage. This requires short response times and reliability for the supervising method.

There has been several cases where monitoring has failed or reacted too slow, which led to severe health effects for the users. One example of this is the Nokia water supply contamination in Nokia, Finland in 2007, where treated sewage water was mixed with drinking water in the water supply. Due to this, about 12 000 inhabitants were infected, causing an epidemic [1].

Incidents like this need to be prevented and minimized for the sake of public health, but many sensors used today react not only to contamination in the water, but also on variation in other parameters, such as temperature, water pressure and chlorine concentration. It is therefore important to retain sensitivity to harmful contamination's but there is a need to suppress the non-harmful parameters. One of these sensors, called an electronic tongue, has in previous research [2] yielded promising results in the area of anomaly detection for wireless sensors, but is still sensitive for time dependant factors and is therefore subject to false alarms.

Machine learning has, due to increasing computing power, gained more relevance in the area of information technology, according to [3]. This is partly because of the applicability it provides on problems which otherwise have complex or unknown models, such as environmental effects on readings. One such important field is within the emerging "Internet of Things" (IoT) technologies, where many small, smart utilities are linked together to optimize and ease up everyday problems. These techniques could make measurements and surveillance more reliable and make vital infrastructure less susceptible to errors and minimize inefficiencies. The purpose of this project is to apply these techniques to surveil the quality of drinking water distribution, and detect potentially harmful contamination with a low probability of raising false alarms.

The project leads to a structure called an Autoencoder being tested for this application. Inputs to the Autoencoder were constructed as vectors of raw measurements and temperatures, where each component was mapped to a specific time of day which for each sample was switched for the most recent sample. To test this structure, an anomaly of various magnitude was added to the data and the reconstructed data was compared to the input, in a mean square sense. This resulted in a solution capable of detecting anomalies of 1.5% and bigger, with a false alarm rate of one per seven days.

This report will be divided into several sections with specific content and intent. In Section II, previous research and results that are relevant to this project are briefly presented, such as tests of the electronic tongue and usage of Autoencoders. Section III presents a set measurable goal, as well as some limitations for the project. Theoretical knowledge relevant for the project, such as methods of preprocessing, artificial neural networks (ANNs) and anomaly detection, is introduced in Section IV. In Section V, the theory is put into practice and the work done in the project is described. The results of the project are presented in Section VI, and discussed in Section VII, where alternative solutions are evaluated and future work is proposed. Lastly, the report is summarized in Section VIII.

Throughout this report, the following notation is used. Matrices are denoted with capital, bold italic letters, e.g. $\boldsymbol{A}$ and vectors are denoted with lowercase, bold italic letters, e.g. $\boldsymbol{x}$. The transpose of a matrix or vector is denoted as $^\intercal$, making $\boldsymbol{A}^\intercal$ the transpose of matrix $\boldsymbol{A}$.

II. Prerequisites

The sensor data provided for this project is from a type of wireless IoT sensor called an electronic tongue. According to [2], the sensor responds fast and accurately to relevant water contaminations such as sewage leaks. The report also states that the electronic tongue is more versatile compared to other comparable sensors investigated in the report. It is however sensitive to time dependant factors such as temperature and water pressure, and is therefore prone to false alarms at an elevated rate. Tests for detecting contamination via linear estimators have been carried out and showed good results in controlled environments, where only a few well known factors influence the measurements, but bad results when used on measurements from real world sampling, where many factors are unknown or unmodelled. This calls for more studies of other detection methods, for example via machine learning.

Autoencoders are commonly used when dealing with anomaly detection due to the methods ability to handle complex non-linear parameters. Both Autoencoders as well as other sorts of ANNs have been used frequently for anomaly detection in many areas; In [4], researchers used an Autoencoder to detect faults in cross-domain data communication, the study in [5] used an Autoencoder to improve energy usage by finding operational faults and improper control strategies in buildings and the authors of [6] used it to detect geochemical anomalies to evaluate cost and risk of drilling at new sites. In [7], both Autoencoders and general ANNs are used in classification of HIV.

The previous research indicates that Autoencoders are a viable method for detecting anomalies, which could be paired with an electronic tongue sensor to detect anomalies of water quality.

III. Problem formulation

The main goal of this project is to create a program that is able to detect anomalies in measurements of water quality that may be caused by contamination, as well as minimize the risk for false alarms. This program should be able to do this using machine learning algorithms on an ANN and train on a provided set of real data series from Linköping University. The data for this machine learning structure was measured with the electronic tongue in both lab environment and real world scenarios.

Since there are countless variations of machine learning algorithms and neural network structures that may work well, the project will focus on a special kind of neural network structure, called an Autoencoder, which will be constructed, trained and evaluated based on its performance.

One way to minimize the influence of time dependant factors is to use machine learning on historic sensor data to learn the behaviour of non-contaminated water flows. A potential method is to apply a neural network that uses the historic sensor data to change its parameters based on gradient descent. Training such networks with an appropriate amount of data will optimize the parameters to give the ability to take the non-harmful time dependant factors, such as temperature, into account when identifying input data.

With this, the impact of some factors that contribute to false alarms from sensors can be reduced or, in best case, completely eliminated.

IV. Theory

To be able to understand and construct a solution to the problem at hand in this project, some theoretical background is needed. This theory is presented in this section and contains info on ANNs in Section IV-A and how they work on a basic level. It also contains methods of signal processing in Section IV-B, so the data can be both optimized before being used by the ANN as well as analyzed after being processed by it.

A. Artificial neural networks

In this project an ANN is to be constructed and its ability to detect anomalies evaluated. This section will contain descriptions of ANNs in general and how they work as well as more details on the special kind of ANN called an Autoencoder which is used in the project.

1) General artificial neural networks: The easiest way to look at a neural network is as a sequence of linear combinations. Let's denote $\boldsymbol{x}$ as the input features to the ANN, $\boldsymbol{W_1}$ as the weights between the input *layer* and the first *hidden layer* $\boldsymbol{z_1}$. Let σ be the element wise *activation function* for each neuron in each layer. The latent representation $\boldsymbol{z_1}$ of input $\boldsymbol{x}$ can then be expressed as

$$\boldsymbol{z_1} = \sigma_1(\boldsymbol{W_1}^\intercal \boldsymbol{x})\,. \tag{1}$$

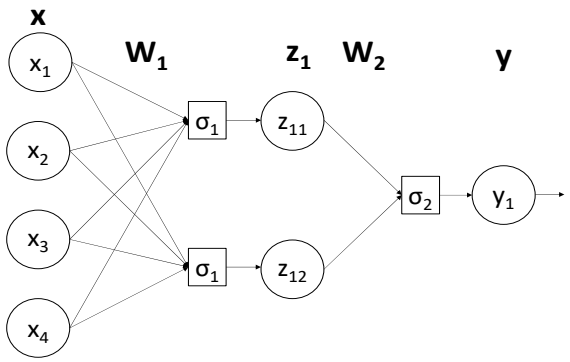


Figure 1: ANN with one hidden layer with two nodes, input layer of four nodes and output layer with one node. Each node in a layer has the same activation function σ.

The *output layer* of the ANN pictured in Figure 1, $\boldsymbol{y}$, can then be mathematically described as

$$y = \sigma_2(\boldsymbol{W_2}^\intercal \boldsymbol{z_1}) = \sigma_2(\boldsymbol{W_2}^\intercal(\sigma_1(\boldsymbol{W_1}^\intercal \boldsymbol{x}))), \tag{2}$$

according to [8]. For the example ANN in Figure 1, the input layer has four *nodes* and the hidden layer has two nodes, which makes $\boldsymbol{x}$ have the dimension 4×1 and $\boldsymbol{z_1}$ having the dimension 2×1. With these layers the weight matrix $\boldsymbol{W_1}$ will have the dimension 4×2 and $\boldsymbol{W_2}$ will have the dimension 2×1. The activation function σ has the function of adding an element of non-linearity to the network, since the multiplications of weight matrices can be condensed down to many linear operations, which in turn can be rewritten as a single linear operation. According to [9], this limits the complexity of the behaviour that is wished to be mapped. When introducing a non-linear activation function σ, the ANN becomes able to compute more complex mappings. There are many used activation functions: hyperbolic tangent $\tanh x$, the sigmoid function $sigmoid(x) = e^x/(1 + e^x)$ and Rectified Linear Unit (ReLU) $R(x) = max(0, x)$ are some commonly used.

Figure 1 is a simple ANN that compresses a vector of four components to a single-valued output, but the exact same reasoning and mathematics shown in Eq. (1) and Eq. (2) can be used for input vectors of arbitrary size and for as many layers as is needed, each with any amount of nodes in them. The amount of nodes per layer does not have to decrease with the progress of the data in the ANN, but can both decrease and increase in any way, one example being an Autoencoder which will be introduced in the next section.

In order for the ANN to learn the behaviour of the data, the weights must be updated according to a *loss function*. This loss function is the description of what the ANN should aim to minimize. There are several different sorts of loss functions, as stated by [10], that all are suitable differently for different applications, so only one will be explained. Since this project has focused on ANNs with the purpose of reconstruction, one popular loss function for this is minimizing the mean squared error, *MSE*, of the difference between the input and output. This means that the purpose of the ANN is to as accurately reconstruct the input, implying that the input and output is of the same dimension. The loss function can then be expressed as

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i||_2^2, \tag{3}$$

where $\boldsymbol{\theta}$ denotes all the weights in the ANN, N denotes the *batch size*, meaning that N inputs will be used before updating the weight matrices, $\boldsymbol{x}_i$ denotes the i-th input of the batch and $\hat{\boldsymbol{x}}_i$ is the reconstructed version of $\boldsymbol{x}_i$ that is the output of the ANN and depends on $\boldsymbol{\theta}$. $||\boldsymbol{x}||_2$ denotes the euclidean norm of the vector $\boldsymbol{x}$. Other applications of ANNs may use other loss functions, many of which are using statistics and probability.

The actual updating of the weight matrices is done by what is known in optimization as the *gradient descent* method, which is described in detail by [11]. This method consists of calculating the gradient of the loss function used and going in the opposite direction from the current parameters, expressed as

$$\boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \rightarrow \boldsymbol{\theta},$$

where $\boldsymbol{\theta}$ again denotes all weights, $\mathcal{L}(\boldsymbol{\theta})$ is the loss function in use and η denotes the *learning rate*, the size of each update to the weights. This gradient descent optimization is done and applied to the ANN after a batch of data of size N has been run through. When all data in the set of training data has been used in the loss function, it is said that one *epoch* has been completed. The learning algorithm is usually repeated several times, several epochs, until either the ANN is considered well trained or, more commonly, a set number of epochs has been iterated through.

It is common practise to divide all available data into two separate sets; training data and testing data. By only training on the training data and verifying the results of the ANN by running the testing data and comparing the results, the phenomena of *overfitting* can be detected. If a model is overfitted, it has learned the specific pattern of the training data too well, and will yield worse results for the testing data, according to [12]. This can partly be counteracted by shuffling the training data after each epoch as well as tested by splitting the available data into subsets, where one set is used for training and one for testing that the model is not overfitted.

2) Autoencoder: One specific type of ANN that is widely used for exact reconstruction is a structure called an Autoencoder. The main attributes of an Autoencoder is that the input layer and output layer have the same amount of neurons, with the intent of minimizing the difference between input and output. Another feature of an Autoencoder is that it performs a dimensional reduction in its hidden layers, as a mean to learn what specific features of the input are most significant for reconstruction, according to [13] and [8]. Due to the reconstruction purpose of an Autoencoder, the loss

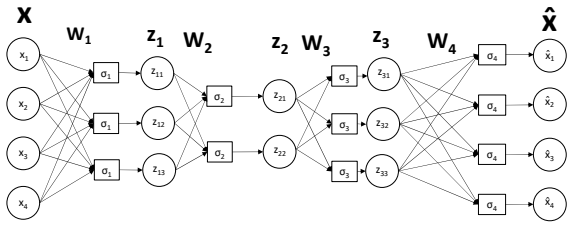


Figure 2: Autoencoder with the smallest dimension being represented by one layer, $\boldsymbol{z}_2$. The input and output layers consists of four neurons, the first and last hidden layers of three neurons and the middle layer of two neurons. Each neuron is connected to an activation function σ. This particular setup is denoted '$75-50-75$'.

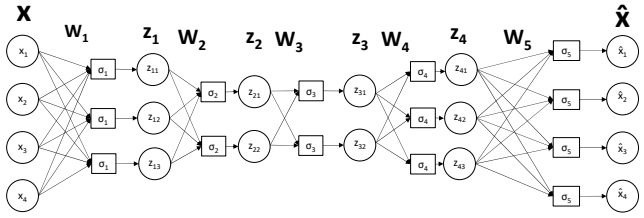


Figure 3: Autoencoder with the smallest dimension being represented by two layers, $\boldsymbol{z}_2$ and $\boldsymbol{z}_3$. The input and output layers consists of four neurons, the first and last hidden layers of three neurons and the two middle layers of two neurons. Each neuron is connected to an activation function σ. This particular setup is denoted '$75-50-50-75$'.

function is most times the MSE, as described in Eq. (3). Throughout the report, the term *reconstruction error* will refer to the difference between the input of the Autoencoder and its output. An Autoencoder is well suited to perform anomaly detection according to [4] and [14], since the learnt patterns of normal data will be disturbed in case of an anomaly.

The Autoencoders that will be used in this project will have a symmetric structure, meaning that for an Autoencoder with n layers, the amount of neurons in layer $i, 0 \leq i \leq n$ will be equal to the number of neurons in layer $n-i, 0 \leq i \leq n$. This structure can come in two different shapes, one, as pictured in Figure 2, where there is a single middle layer with the least amount of neurons, or, as seen in Figure 3, with two equally sized layers with the same number of neurons as the minimum.

Throughout this report, a certain type of notation will be made for different Autoencoder setups. This notation is based on the percentage of neurons in a particular layer compared to the input layer. It will look like '$n-m-p-m-n$', for an Autoencoder with five hidden layers, where the first and last layer has $n\%$ of the amount of neurons that the input layer has, the second and fourth hidden layers have $m\%$ and the middle layer has $p\%$. For example an Autoencoder with four neurons in the input and output layers and two equal hidden layers with two neurons each will be denoted '$50-50$'.

B. Signal processing

1) Least square estimation: When analyzing a factor Fs influence on a series of measuring points without knowing their exact relation, as in linear, quadratic, sinusoidal, etc., the equation

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{A} = \begin{bmatrix} 1 & F_0 & F_0^2 & \cdots & F_0^m \\ 1 & F_1 & F_1^2 & \cdots & F_1^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & F_n & F_n^2 & \cdots & F_n^m \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad (4)$$

is often set up, where n denotes the number of equations, for example the number of samples in a time series, and m denotes the highest order of polynomial that will be considered. Note that other functions than polynomials such as trigonometric and exponential functions can be used as well by just expanding the number of columns in $\boldsymbol{A}$. In Eq. (4), $\boldsymbol{A}$ is the matrix that contains the values of the measured factor F, which for example could be temperature, $\boldsymbol{x}$ is the solution to the equation and $\boldsymbol{b}$ is the measuring points which are being influenced by F. In most cases $n > m$, and for those cases there are more equations than unknowns, the system is most likely inconsistent and a solution $\boldsymbol{x}$ that satisfies all n functions does not exist. In such a case a least square estimator can be used to find the best approximation $\overline{\boldsymbol{x}}$, in the meaning that the best $\overline{\boldsymbol{x}}$ minimizes the Euclidean distance between each approximated point and actual value. This best solution, $\overline{\boldsymbol{x}}$ can be found by solving the normal equations,

$$\boldsymbol{A}^\intercal \boldsymbol{A}\overline{\boldsymbol{x}} = \boldsymbol{A}^\intercal \boldsymbol{b}, \quad (5)$$

according to [15], since the new system in Eq. (5) is reduced to a symmetric system, which in most cases have a solution $\overline{x}$. This $\overline{\boldsymbol{x}}$ is then the best approximation of the solution $\boldsymbol{x}$ in Eq. (4), and can be used to eliminate the factor. Since $\boldsymbol{A}\overline{\boldsymbol{x}}$ is the best approximation of factor F, the residual

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\overline{\boldsymbol{x}} \quad (6)$$

will be the difference between the original data and the approximation of the factor F, which should be the original data without the influence of factor F.

2) Anomaly detection: Since an Autoencoder is an ANN structure with the purpose of reconstructing a series of data, an input with large variations will not follow the patterns of the training data and thus generate a reconstruction that is less similar to the input. This sort of classifying is called deviation based anomaly detection. By looking at the mean square error of the reconstruction and input, and compare it to a set threshold level, an alarm signal can be activated to alert that there is something unusual in the input reading. This threshold level is usually determined by some sort of statistical property of the training data, such as the variance. The threshold level α is then defined as

$$\alpha = k \cdot \mathbb{E}\big[\mathrm{Var}(\boldsymbol{x} - \hat{\boldsymbol{x}})\big] + \mathbb{E}\big[||\boldsymbol{x} - \hat{\boldsymbol{x}}||_2^2\big],$$

where k is a factor to be determined experimentally later, usually in the interval $[2-8]$, $\mathbb{E}\big[\mathrm{Var}\big]$ is the mean of variances for all errors of the training data. $\mathbb{E}\big[||x - \hat{x}||_2^2\big]$ denotes the mean of all squared reconstruction errors. Since the mean of the reconstruction error is low for the majority of all data, the variance and mean squared error will be close to each other

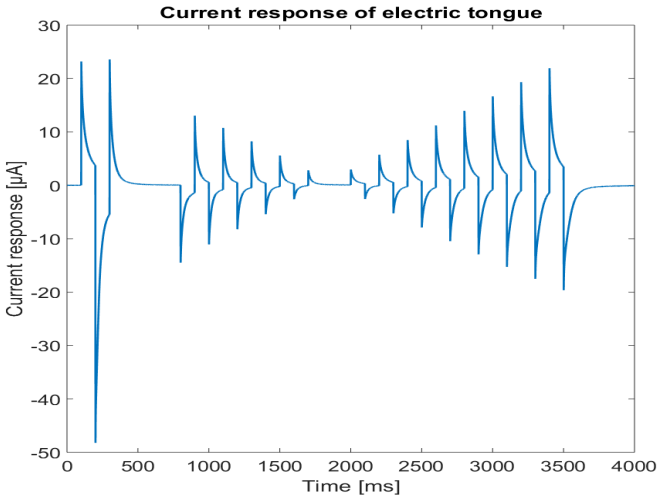


Figure 4: Current response on an excitation voltage in an electronic tongue.

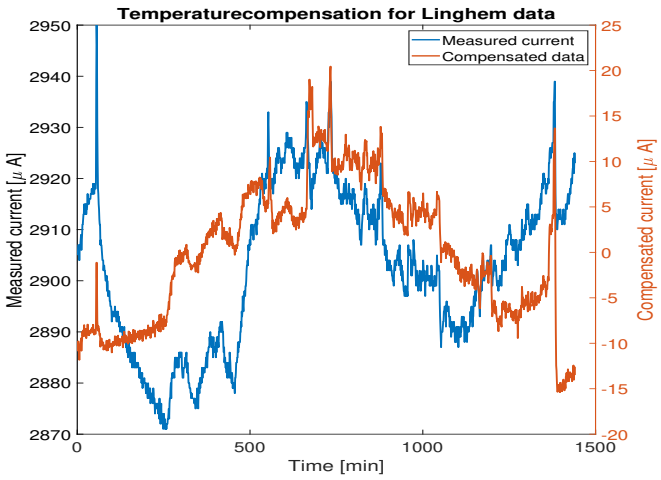


Figure 5: Measured data (blue) from Linghem 2017-06-28 and compensated data (orange) when only temperature was compensated.

according to [16], making the variance a suitable parameter to base the threshold level on.

V. Method

In this section, the different steps that were taken in order for the project to be completed are described. In Section V-A, the data used throughout the project is described, where the preprocessing of data is described in Section V-A1 and the assembly of input vectors to the ANN is explained in Section V-A2. In Section V-B, the construction, training and evaluation of different Autoencoder structures are described, as well as the threshold level being set[1].

A. Processing of sensor data

The data used to train and verify the ANN was provided by Linköping University and consisted of 90 days of measurements from the pressure booster station Linghem, as well as 4 days from the water treatment plant in Nykvarn. The Nykvarn data has some measurements where sewage water was manually added in a controlled environment and temperature was the only measured environmental factor. In Linghem, the measurements are of supposedly non-contaminated water with several natural variations such as temperature, chlorine, etc.

The actual measurements are composed of continuous 10 second series over 24 hours each day. Each 10 second series consists of a four seconds long current response of an excitation voltage, followed by six seconds of down time where no measurements are done. The four seconds current response can be seen in Figure 4. The four second measurement is sampled with a frequency of 1 kHz from three different sensor probes made of different materials, resulting in 3×4000 data points every 10 seconds.

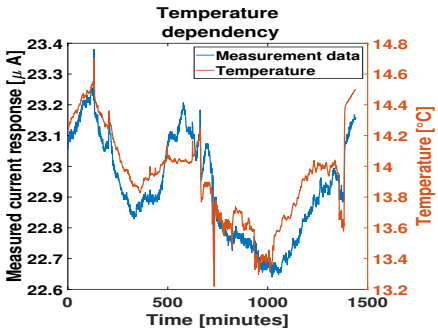
[1]The code of the project can be found at https://gits-15.sys.kth.se/mgelin/EF112X/

Because of limited processing power it was recommended by the supervisor to keep the ANN small, which led to only one point per 10 second series being used as input data for the input to the ANN. Point 101 from the first probe was used, which represents the first positive peak in Figure 4.

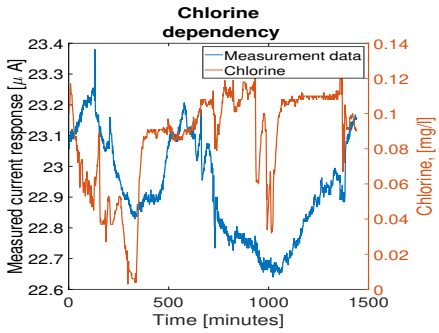
1) Preprocessing: In order to make the ANN as efficient as possible, several environmental factors that may influence the measurements were investigated. If some factors could be either ruled as not influencing readings or eliminated before being fed into the ANN, it would reduce the complexity of the ANN as well as make it more reliable. These environmental factors are: temperature, water inflow, water outflow, water pressure in, water pressure out and chlorine excess. These were plotted along with the measurement for the data, from the date $2017{-}06{-}21$, and can be seen in Figure 6. Note that all factors displayed in Figure 6 were taken from the Linghem site, the data from Nykvarn only had corresponding temperatures available since it was in a more controlled environment and the other factors did not interfere with measurements.

All environmental factors were examined and their potential correlation to the current response evaluated. As can be seen in Figure 6a, the temperature shows a strong correlation to the current response. This caused the temperature to be considered a factor that should be taken into account, either by eliminating its influence or as features to the autoencoder. Chlorine excess shows that it at most has a small impact on the current response, if any at all, which can be seen in Figure 6b. This assessment goes for the rest of the factors as well, the flow of water in and out of the measuring point which can be seen in Figure 6c and Figure 6d, and in and out pressure at the measuring point, as seen in Figure 6e and 6f respectively.

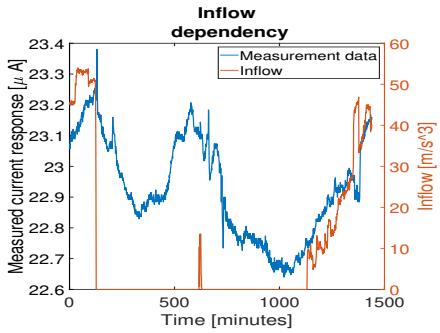
Since temperature is the factor that showed most correlation to the measurement data, a first compensation was performed on data from Nykvarn, where the only influencing factor was



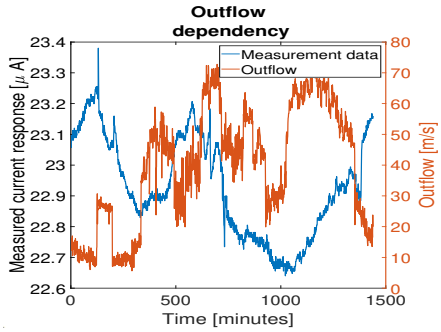
(a) Temperature of the water at the measurement point (orange) and current response (blue).



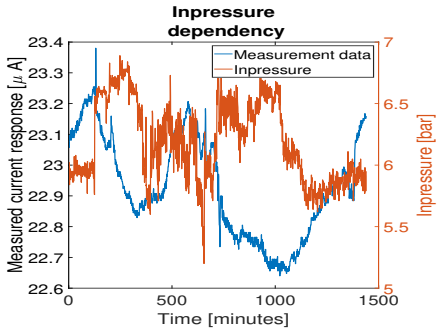
(b) Chlorine excess in the water at the measuring point (orange) and current response (blue).



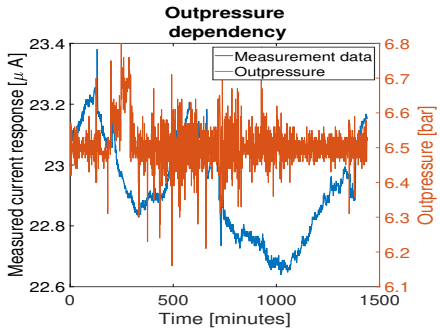
(c) Inflow rate of water to the measuring facility (orange) and current response (blue).



(d) Outflow rate of water from the measuring facility (orange) and current response (blue).



(e) Water pressure into the measuring point (orange) and current response (blue).



(f) Water pressure out from the measuring point (orange) and current response (blue).

Figure 6: All considered environmental factors plotted together with the current response. All different data on factors and measurements were taken from Linghem at 2017-06-21.

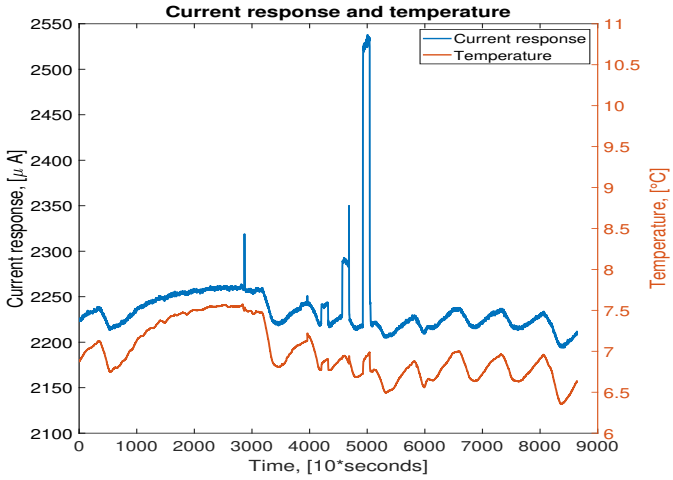


Figure 7: Current measurements (blue) and temperature (orange) at Nykvarn treatment plant. Measurements were taken at 2017-02-02.

temperature. As shown in Figure 7, the temperature shows a clear connection to the measurement. Note that in this particular measurement, at 2017-02-02, sewage water was manually injected to the water network at four different times, which can be seen in Figure 7 at times 3900, 4300, 4600 and 5000. In all preprocessing of environmental factors the degree of complexity of the model has been limited to linear least square estimators, on the advice of the supervisor because of limits in time and uncertainty of how or if the factors affect measurements. The temperature influence was compensated as described in Section IV-B1, where

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ T_1 & T_2 & \cdots & T_n \end{bmatrix}^\intercal, \boldsymbol{b} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^\intercal, \quad (7)$$

T_i and x_i are the temperature and measured data at sample i. With this, the compensated data, x', can be calculated as

$$\boldsymbol{x'} = \boldsymbol{b} - \boldsymbol{A}\overline{\boldsymbol{x}}, \qquad \boldsymbol{A}^\intercal \boldsymbol{A}\overline{\boldsymbol{x}} = \boldsymbol{A}^\intercal \boldsymbol{b}. \quad (8)$$

This compensated data can be seen in Figure 8, where it is clear that the only variation in the measured data is the added sewage water. This solidifies temperature as an important influence which will be accounted for in either preprocessing or in the actual ANN structure.

Two approaches for eliminating the environmental factors on the Linghem data were used, one where the factors were eliminated sequentially and one where all were compensated simultaneous. Both used linear least square estimation, as described in Section IV-B1.

For the simultaneous compensation the coefficient matrix $\boldsymbol{A}$ and datavector $\boldsymbol{b}$ in Eq. (7) looks like

$$\boldsymbol{A} = \begin{bmatrix} 1 & T_1 & C_1 & IF_1 & OF_1 & IP_1 & OP_1 \\ 1 & T_2 & C_2 & IF_2 & OF_2 & IP_2 & OP_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & T_n & C_n & IF_n & OF_n & IP_n & OP_n \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

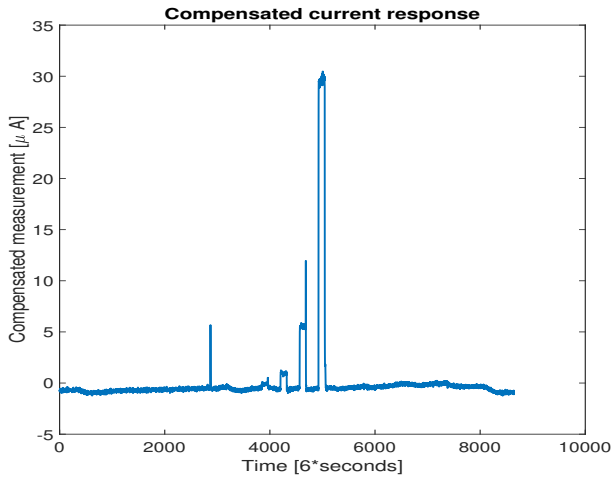


Figure 8: Compensated measurement data from Nykvarn. The data that was compensated was measured at 2017-02-02. At times 3900, 4300, 4600 and 5000 there was sewage water manually inserted into the water.

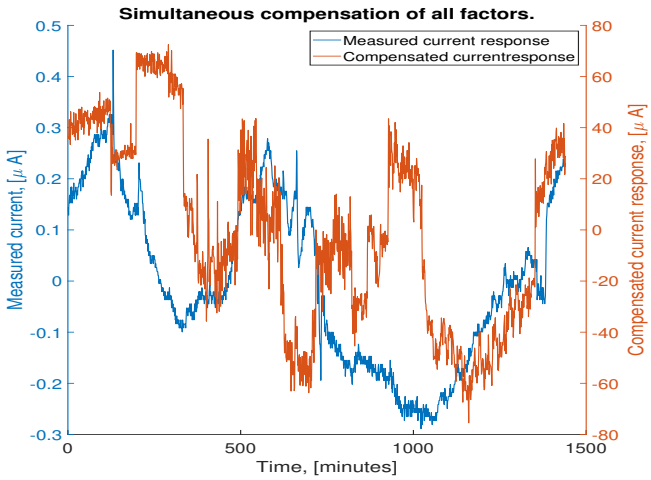


Figure 9: Current response (blue) and compensated current response (orange) when all factors were compensated simultaneous. Data from Linghem at 2017-06-21 was used.

where T_i is the temperature at timepoint i, C_i is the chlorine excess, IF_i is the inflow rate of water, OF_i is the outflow rate of water, IP_i is in-pressure at the measuring point and OP_i is the out-pressure at the measuring point, and x_i is the measured current.

With this, the normal equations in Eq. (5) can be solved and the residual in Eq. (6) will result in the best compensated data when using only linear least square estimators. The result can be seen in Figure 9. The other method used to try to compensate for environmental factors was to compensate one factor at a time, iterating through all six different influences. With this method, the coefficient matrix $\boldsymbol{A}$ and datavector $\boldsymbol{b}$ can be written as

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ F_1 & F_2 & \cdots & F_n \end{bmatrix}^\intercal, \boldsymbol{b} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^\intercal,$$

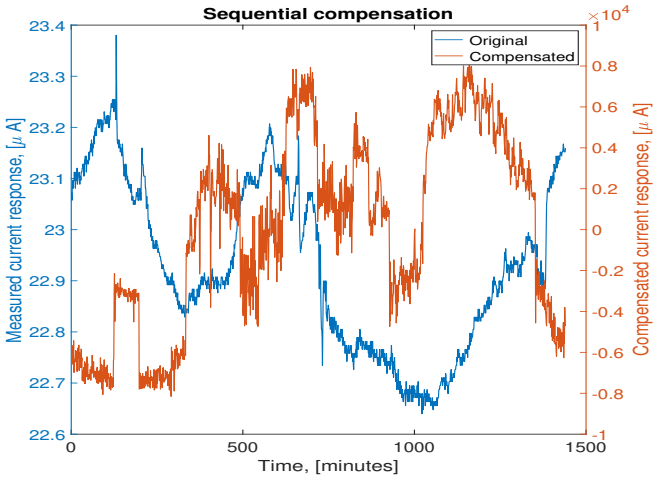


Figure 10: Current response (blue) and compensated current response (orange) when factors were sequentially compensated. Data from Linghem at 2017-06-21 was used.

where F_i is the current factor at timepoint i, which will change for each iteration, starting with temperature followed by chlorine excess, etc. With this, Eq. (5) and Eq. (6) can be solved as before and get a compensated dataseries, which can be seen in Figure 10. As can be seen in Figure 9 and Figure 10, the linear least square estimation yielded results that seem erratic, which led to the decision that all factors except temperature was not used further in this project. These results confirm the results specified in the previous tests mentioned in Section II. The temperature was deemed playing a significant role of the behaviour of the electronic tongue, so the data from Linghem was compensated for temperature influence, since the data from Nykvarn showed a good result even from a linear least square estimator, as described earlier. The same coefficient matrix $\boldsymbol{A}$ and datavector $\boldsymbol{b}$ can be used as in Eq. (7) and Eq. (8) with the data from the Linghem measurements. This resulted in compensated timeseries that were deemed credible, which can be seen in Figure 5.

2) Assembly of input vectors: If the inputs were to be a time series of 24 hours with 1 minute sampling periods, two clear downsides were found; a lack of training data and an ANN with long response time.

In order to make a well trained ANN, a lot of training data is required. In this project, the data was limited to 90 days of measurement, which is not enough for the training if the days measurements were used unprocessed, meaning that one input vector consists of one complete day, resulting in only 90 different input vectors. Several solutions to this problem was discussed and analyzed, such as using smaller time series, using more datapoints from the 4 second current response or trying to shift the time series in some way. The last option, to shift the data, was considered the most promising and decided upon. By shifting the data it was believed that the time dependency would be accounted for in some way, as well as making more datasets to train the ANN on.

The datasets were created by gradually shifting the values in the time series to the values of the next day. Let

$$\boldsymbol{v}_d = \begin{bmatrix} x_{d,1} & x_{d,2} & \cdots & x_{d,n-1} & x_{d,n} \end{bmatrix}^\intercal$$

be the vector of measurements from day d, with components $x_{d,T}$ being the measurement values on day d and at time T, with $T = 1$ the first value of the day and $T = n$ the last value. With a sampling period of 1 minute, $n = 24 \times 60 - 1 = 1439$. Notice the -1, which is due to a bug in the sensordata making some days not having measurements for all times.
The new data series were then created as

$$\boldsymbol{v}_{d,M} = \begin{matrix} [x_{d,1} & x_{d,2} & \cdots & x_{d,M-1} & x_{d,M} \\ & & x_{d-1,M+1} & \cdots & x_{d-1,n}]^\intercal, \end{matrix} \tag{9}$$

where d denotes the latest day of measuring and M denotes the M-th sample of day d. This way, if an anomaly is detected, it is clear that the anomaly happens at sample M of the day d.

Two different types of input vectors was assembled; one with data that was already processed with respect to temperature, as described in Section V-A1, where the data points $x_{d,T}$ in Eq. (9) are already compensated for temperature influence. The other sort of input vector consists of the same structure as Eq. (9) but includes both raw measurement data as well as the measured temperature, making it twice as long. This will look like $\boldsymbol{v}_{d,M}$ expanded with vector

$$\boldsymbol{t}_{d,M} = \begin{matrix} [t_{d,1} & t_{d,2} & \cdots & t_{d,M-1} & t_{d,M} \\ & & t_{d-1,M+1} & \cdots & t_{d-1,n}]^\intercal, \end{matrix}$$

where the indexing is exactly the same as in Eq. (9).

At the end of each input vector, for both the temperature compensated version and the raw data version, a value specifying the current weekday is added. This is done so that Monday corresponds to the value $1/7$, Tuesday to $2/7$, and so on until Sunday which corresponds to $7/7 = 1$.

The data was divided into two subsets, one for training consisting of 80% of all data and one for testing, with the remaining 20%.

In order to have inputs that are all in the same range of values, the input data was normalized to an interval of zero to one, like

$$\boldsymbol{v}_{i,norm} = \frac{\boldsymbol{v}_i - \min\limits_j(\boldsymbol{v})}{\max\limits_j(\boldsymbol{v}) - \min\limits_j(\boldsymbol{v})},$$

where $\boldsymbol{v}$ is the measurement data, i is the time of the latest sample and $\boldsymbol{v}_{i,norm}$ is the normalized data vector. The measurement data and temperature data was normalized separately so they don't interfere with each other. Since the indicator for the weekday is already in the interval $[1/7, 1]$, it is not necessary to normalize it again.

B. Design of the Autoencoder

The first step of building the fully functioning anomaly detection structure was to assemble an Autoencoder. This was done by using the Tensorflow [17] machine learning library and their neural network API Keras [18], all run in Python. To start off, an Autoencoder was constructed for the already existing database MNIST, available [19] and included in Tensorflow. MNIST was used as it was easy to verify that the Autoencoder worked as intended without having to worry about any errors that might be in the input data. For MNIST, an Autoencoder with the structure '$90 - 70 - 70 - 90$' was constructed and trained with the MNIST data, with a resulting reconstruction error of 2%. All activation signals used were $ReLU$ except for the very last output layer, where a $sigmoid$ function was used to normalize the data.

This Autoencoder was also used with the manually assembled input data, as described in Section V-A2, with good results, which confirms that an Autoencoder is a suitable structure for the task at hand. With this result, other Autoencoder structures were constructed to test which would yield the best results. In total six different Autoencoder structures were constructed and trained, where each version was used with both preprocessed data and raw data, as explained in Section V-A2. For these structures, the time elapsed between each input vector was increased to 60 min to decrease the training time. The structures and their respective reconstruction error after 100 epochs can be seen in Table I. The loss functions as a function of epochs for each Autoencoder structure and input structure can be seen in Figure 11 and Figure 12. Similar to the Autoencoder used for MNIST, all activation functions were $ReLU$ except the output layer, which used a $sigmoid$ function.

Table I:
Reconstruction error for different structures

Structure	Raw data	Compensated data
'$70 - 30 - 30 - 70$'	0.0012	0.0707
'$85 - 40 - 40 - 85$'	0.0017	0.0717
'$50 - 15 - 15 - 50$'	0.0016	0.0679
'$70 - 30 - 10 - 30 - 70$'	0.0015	0.0754
'$80 - 50 - 30 - 50 - 80$'	0.0026	0.0718
'$50 - 25 - 10 - 25 - 50$'	0.0050	0.0717

Loss function value after 100 epochs of training. Compensated data is when the data is preprocessed with respect to temperature and raw data is when the raw measurements and temperatures are used unprocessed as inputs. After each epoch, the data was shuffled.

With these results, it was decided to use the input structure with the raw data in combination with raw temperature, since the best reconstruction error for preprocessed data in the input vector is more than ten times larger. The structures with best reconstruction was the two with layers of 70% and 30%, with the difference being an additional 10% layer in the middle for one of them. Because of slightly better loss and a lower number of weights, the structure of $70 - 30 - 30 - 70$ was decided upon.

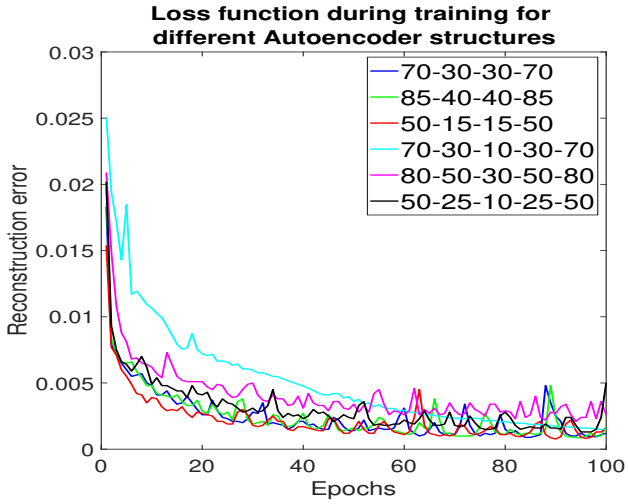


Figure 11: Reconstruction error as a function of number of epochs, for each Autoencoder structure evaluated when the input data was raw measurement data and temperatures. After each epoch, the data was shuffled.

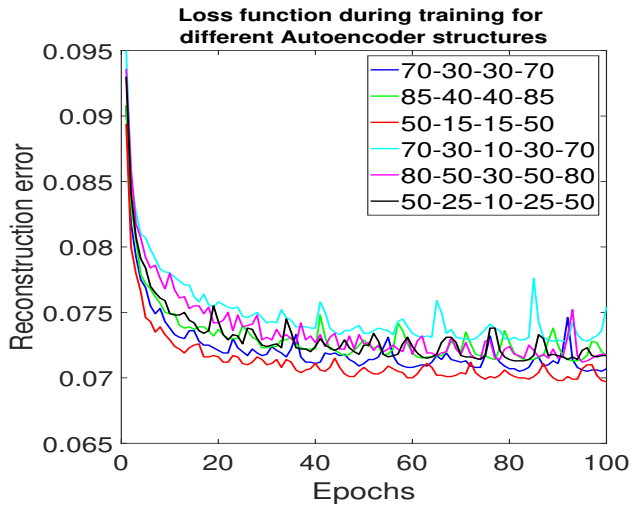


Figure 12: Reconstruction error as a function of number of epochs, for each Autoencoder structure evaluated when the input data was data compensated for temperature. After each epoch, the data was shuffled.

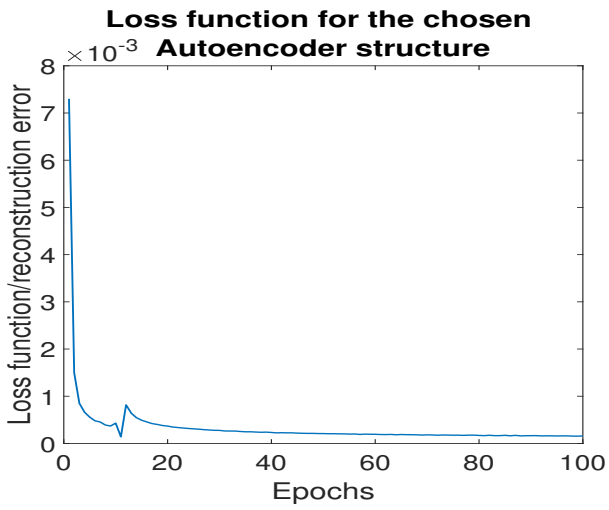


Figure 13: Reconstruction error as a function of number of epochs elapsed in training for the chosen Autoencoder. Here, a sampling rate of one minute was used. After each epoch, the data was shuffled.

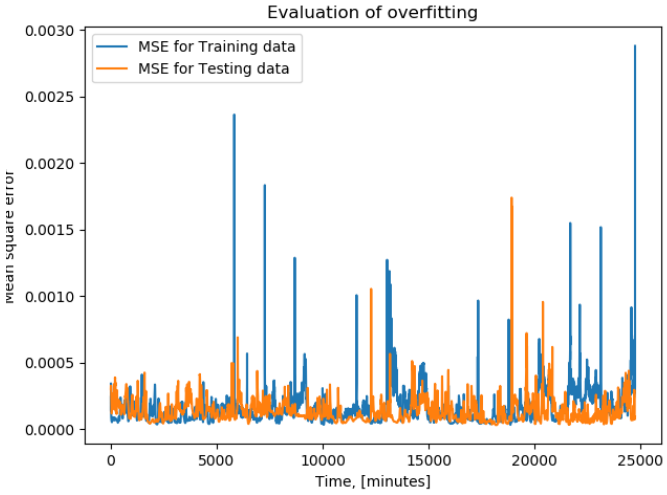


Figure 14: Reconstruction error for testing data set (orange) and training data set (blue) for the Autoencoder structure '$70-30-30-70$'. Note that all data is normal without added anomalies, making the high peaks potential false alarms.

To further improve performance, the chosen Autoencoder structure was trained again with all the data in the training set, meaning one sample for each minute in the training data set. This yielded a better reconstruction error which can be seen in Figure 13, resulting in a loss value of 15×10^{-5}, significantly less than previous training.

To verify that the model was not overfitted the testing data set, which was not used to train the Autoencoder, was reconstructed and compared to the reconstruction of the first 25% of the training data set. The result is shown in Figure 14 and the mean value of the reconstruction error is 13.26×10^{-5} for the testing set and 15.97×10^{-5} for the training set. The fact that the testing data has a better reconstruction in general indicates that the model is not overfitted.

The anomaly detection threshold was implemented by reconstructing all available data, both training and testing sets, and taking the mean of all variances of the difference between input and reconstruction. This gave a value of 17.3×10^{-5}. The mean of all mean squared errors between input and output was also computed and found to be 16.9×10^{-5}. Together, these values set the threshold level α as

$$\begin{aligned} \alpha &= k \cdot \frac{1}{N}\sum_{i}^{N}(\mathrm{Var}_i) + \frac{1}{N}\sum_{i}^{N}\mathbb{E}\left[||\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i||_2^2\right] \\ &= k \cdot 17.3 \times 10^{-5} + 16.9 \times 10^{-5}, \end{aligned} \tag{10}$$

where Var_i is the variance for the reconstruction error of input vector that ends at time i, $\mathbb{E}\left[||\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i||_2^2\right]$ is the mean of the squared reconstruction error for vector ending at time i and k is some integer. By comparing the reconstruction error with the threshold level α, an alarm signal is raised if the

reconstruction error is larger than α. If the training data is reconstructed and then evaluated with a threshold level with different k, false alarm rates of the solution can be used to determine a suitable k, in addition to its sensitivity. k was iterated through for integers between two to eight and the results can be seen in Table II.

Table II:
False alarm rate

k	2	3	4	5	6	7	8
False alarms for 40 days	588	177	72	23	12	6	4

Iterations of k in the threshold level. In order to maintain a sensitivity that is useful, a maximum of $k = 8$ was set.

In order to maintain a false alarm rate that is low and still have a sensitivity that can detect smaller anomalies, $k = 7$ was chosen, resulting in a threshold level $\alpha = 1.38 \times 10^{-3}$.

VI. Results and simulations

This project has lead to a machine learning solution that is able to detect anomalies concerning contamination in the water distribution network. It is also able to, through its training, suppress false alarms that may have arisen due to periodic time-varying variables.

The input structure consists of raw sensor data where each point represents the peak value of one minute samples over a 24 hour period which corresponds to 1439 samples. The associated weekday is represented as one element. Each one minute sample also has a corresponding temperature value. The input vector is assembled as a 2879×1 column vector.

When training, the data is updated each minute by replacing an element with the corresponding element from the next day. This maintains the time of day associated with each sample so each series covers the same time period. This was done to account for the cyclic nature of certain parameters. The Autoencoder network consists of four hidden layers with the dimensional reduction of '$70 - 30 - 30 - 70$'. This yields a reconstruction error of 15×10^{-5} as shown in Figure 13.

The detection is made through an alarm function calculating the mean square error of the output as a function of time. If the mean square error exceeds the threshold set to $\alpha = 7 \cdot \mathrm{Var} + \mathbb{E}$ it classifies as contamination and will detect an anomaly. This threshold level was calculated to be $\alpha = 1.38 \times 10^{-3}$.

The following example is useful to show where the model works well. In Figure 15, the difference between input and output when introducing an anomaly is shown. The anomaly is generated by increasing the measurements magnitude with 2.5% at specific times. In Figure 15 the anomaly is introduced at time 1350 corresponding to day 1 at 22:30. The reconstruction error is at this point 0.0126, which is significantly higher compared to inputs with no anomalies around 0.000125. The 2.5% increase is not the minimal anomaly that could be detected but yields descriptive

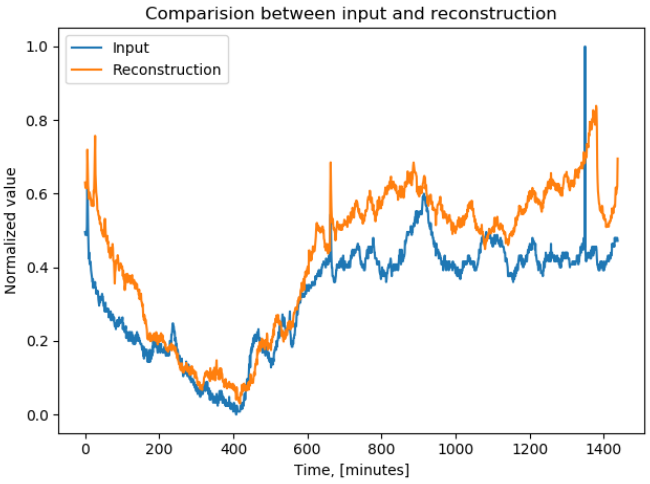


Figure 15: Input data (blue) and output data from the Autoencoder (orange) as a function of time. An anomaly is introduced at time 1350 with a magnitude of $+2.5\%$.

Figure 16: Input data (blue) and output data from the Autoencoder (orange) as a function of time. An anomaly is introduced at time 600 with a magnitude of -2%.

figures and examples. Negative peaks, where an anomaly is introduced as a decrease in magnitude, were also tested and gave equal results as a positive anomaly. An example of a negative anomaly can be seen in Figure 16.

The position of the anomaly in relation to the trend of the data, i.e. if the curve is rising fast, declining fast or is relatively stable, was evaluated as well. The behaviour of this can be seen in Figure 17, where three percentages, 2.5%, 2% and 1.5%, were added at three different places of the data. As can be observed, the anomalies at a relatively flat part of the curve were easier to detect, compared to if the values around the anomaly was changing rapidly. The figure also shows that both positive and negative peaks yield detectable reconstruction errors, which also varies with the placement.

Figure 18 shows the reconstruction error as a function of

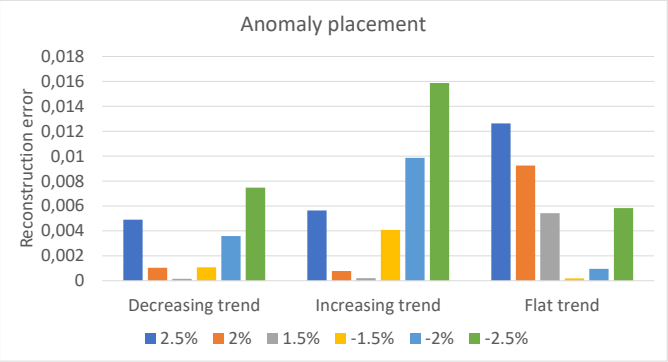


Figure 17: An evaluation of how the placement and character of the anomaly influences reconstruction. Six percentages, 1.5% (grey), 2% (orange), 2.5% (dark blue), -1.5% (yellow), -2% (light blue) and -2.5% (green), were added at three different times, each one corresponding to either a decrease or increase in measurement values, or if the values stay relatively constant.

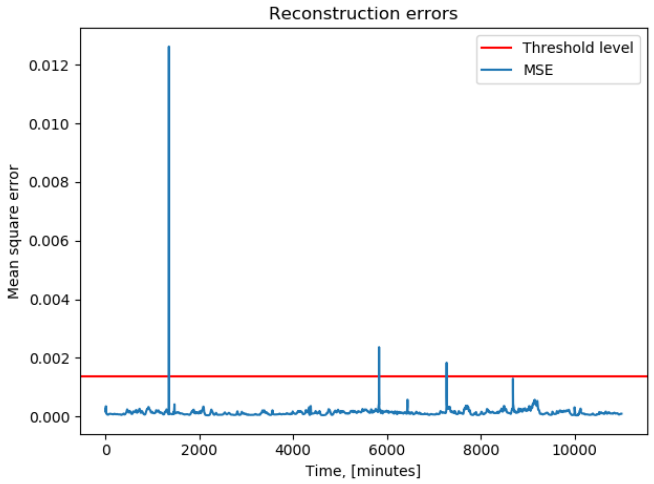


Figure 18: Alarm signal as a function of time, from 0 to 11000 corresponding to a span of 183 hours. If the reconstruction error (blue) exceeds the threshold level (red), an alarm signal is raised indicating that there is an anomaly at the current time. The peak at $t = 1350$ corresponds to an added anomaly of $+2.5\%$ and the peaks at 5800, 7300 and 8700 are naturally occurring.

time. The time period relevant for this is the first 1400 samples, which clearly classifies the peak at $t = 1350$ as an anomaly. All other samples result in low errors despite the inputs relative turbulence. The peaks between $t = 5700$ and 9000 are due to some of the samples being reconstructed poorly, resulting in false alarms. The entire graph displays an eight day period with two other anomalies detected that are considered false alarms. These will happen but with the current iteration of the solution and available data, this happens at a rate of about once per seven days, as seen in Table II.

VII. Discussion

With these results it is possible to conclude that a machine learning solution can be applied to increase the reliability of anomaly detection for this water distribution problem.

A. Network type

The Autoencoder structure used, '$70 - 30 - 30 - 70$', was picked because it had the smallest reconstruction error of the ones evaluated, as well as having a reasonable training time. The structure '$50 - 15 - 15 - 50$' was also an equally valid candidate. This structure converged faster but had a slightly higher reconstruction error after 100 epochs, as seen in Table I and Figure 11. The faster convergence in addition to having less parameters might be preferable over a more complex structure with better reconstruction that converges slower. Note that the difference in reconstruction error is minimal, so both structures would yield similar results.

Deeper Autoencoder structures gave minimal increase in performance while increasing computation time compared to this. It was therefore deemed not suitable for the problem. It is reasonable to assume that a deeper network with more hidden layers and more neurons in each layer would be able to learn the patterns of the training data even better, thus leading to a better reconstruction. A deeper Autoencoder will probably require more training data to avoid overfitting which, together with an increase in the number of weights being updated each epoch, will lead to a significant increase in training time. This may still be preferable in the case of a deeper Autoencoder decreasing the number of false alarms and the minimum level of contamination needed to trigger an alarm.

However, this does not mean that the structure '$70 - 30 - 30 - 70$' was the best structure in terms of detecting anomalies. One of the structures evaluated that was found having to big of a reconstruction error might turn out to detect anomalies with much higher sensitivity than the others but a higher reconstruction error for normal data.

B. Input structure

The current input structure consists of the raw sensordata and associated temperature of an entire day. This structure was chosen so some factors that depend on the time of day is reduced, like a higher water usage at the mornings and evenings compared to times when most people are at work. From the results it can be concluded that this input structure was successful and a viable option as it worked well with the Autoencoder. The choice of only using one datapoint from the current response to the excitation voltage, as well as measurements from only one of three probes proved to be sufficient.

The overall performance may have been increased if multiple datapoints were used. This could have been realised in ways of extending the input vector with more measurements or using some sort of weighted average between several datapoints in the four second series. Other probes or combinations of the three could be used to increase the size of the input vector. Due to the different materials of the probes, different sorts of contamination may be detected with varying precision depending on which probe is used. In the case of using several

of the probes, the solution may be more versatile in terms of which sorts of anomalies can be detected. When using more datapoints and probes, the Autoencoder will increase in complexity, which means more parameters to be optimized, leading to longer training time as well as more data to train on being needed to avoid overfitting.

C. Preprocessing

As shown in the method other time-varying parameters than temperature had clear non-linear dependencies. These dependencies of the sensor were unknown and tests with a linear least square estimator showed virtually no correlation, and was therefore deemed not useful as a way of processing the data. A way of making accurate estimators could be to separately research how the electronic tongue sensor behaves with these parameters, such as chlorine concentration, in controlled environments. The implications of eliminating all or some of these non-harmful parameters are that it would to a larger degree differentiate anomalous from normal data and make the statistical evaluation more reliable. With preprocessing, the need to include other factors other than the compensated data might not have been necessary and could work as a way of reducing the number of computations in the network.

D. Future work

There are several ways that might improve upon the current work, either through further development of the model or ways to improve performance by implementation changes.

1) Model changes: An option to increase the flexibility of the anomaly detection is to introduce a dynamic threshold that is able to adjust its value. A proposed way of doing this is to further segment the statistical evaluation of mean square error and variance when calculating the threshold. This can be done through using smaller time segments, for example the last 12 hours. The threshold level would then be able to adjust accordingly with the latest measurements, in case of the mean square error consistently being higher for some natural reason, like a dry season. The threshold level could in such a scenario be increased automatically in order to not raise false alarms frequently. While this may reduce the false alarms, it may also result in a reduced sensitivity for actual contaminations, making more potentially harmful anomalies slip by unnoticed. This might also increase the overall computational cost.

2) Implementation changes: In a real world application it might be worth to use a low-level programming language such as C or C++ to increase computational performance and give the ability to run on weaker processors. This would give room for larger networks and input structures as well as being able to use more training data without a severe time cost.

E. Future and Impact

The use of machine learning in critical infrastructure could make measurements and surveillance of large systems more reliable and less susceptible to errors while minimizing inefficiencies. Large systems can be hard to monitor so ways of condensing and adapting anomaly detection would be especially of economic and safety interest. Solutions like the one presented could be used for a wide variety of fields. From healthcare, for example detecting anomalies of oxygen concentration in the blood, to cybersecurity, for example detecting imminent attacks on servers by detecting anomalies in the data traffic. As mentioned above the use of these solutions, and machine learning in general, is believed to have great economic value so its prevalence in society is thought to increase greatly in all areas where reliable data can be extracted and used.

VIII. Conclusion

The project was meant to examine if machine learning algorithms were a viable option for reliably monitoring water quality. This has been confirmed by constructing an ANN of the sort Autoencoder that can detect anomalies in the measurement data. The Autoencoder was trained to as accurately as possible reconstruct the input, with data provided by Linköping University from a sensor called an electronic tongue. The data was evaluated with linear least square estimators to deduce if some given factors, such as temperature and chlorine excess, could be eliminated before the Autoencoder. The input vectors of the Autoencoder were assembled as a vector where each component represents data at a given time, followed by the temperature at the same time. Each instance of the input vectors then sequentially changed one value at a certain time to the corresponding data of the next day, see Section V-A2 for a more in-depth description. After training, the difference between input and reconstruction was used to classify if there was an anomaly in the input vector. The solution has a false alarm rate, where it raises an alarm despite no anomaly was in the input, of about one time for every seven days, as well as being able to detect anomalies of magnitude $\pm 1.5\%$. There are several interesting aspects to be evaluated and improved on in the future. For example, setting a dynamic, time-dependant threshold or trying different neural network structures and introducing other sorts of layers, for example recurrent layers or convolutional layers.

Acknowledgment

The authors would like to thank our supervisor Rong Du for his invaluable help and guidance. Without his feedback and tips, the project would not have been nearly as successful. The authors would also like to thank Andreas Westman for helping with various kinds of software related issues.

References

[1] Sirén, S. (2007, Dec.) Förbjudet att använda vatten i nokia. Svenska YLE, Sweden. . [Online]. Available: https://svenska.yle.fi/artikel/2007/12/05/forbjudet-att-anvanda-vatten-i-nokia-0

[2] P. Jonsson, D. Lindgren *et al.*, "Elektronisk tunga och andra onlinesensorer för detektion av föroreningar i dricksvattennätet – en utvärdering," Svenskt Vatten AB, 167 14 Bromma, Tech. Rep. 15, 2018.

[3] G. Rebala, A. Ravi, and S. Churiwala, *An Introduction to Machine Learning*, 1st ed. Springer, 2019, ch. 1, pp. 1–16.

[4] Y. Ikeda, K. Ishibashi *et al.*, "Anomaly detection and interpretation using multimodal autoencoder and sparse optimization," *ArXiv*, vol. abs/1812.07136, 2018.

[5] C. Fan, F. Xiao, Y. Zhao, and J. Wang, "Analytical investigation of autoencoder-based methods for unsupervised anomaly detection in building energy data," *Applied Energy*, vol. 211, pp. 1123–1135, 2018.

[6] H. Moeini and F. M. Torab, "Comparing compositional multivariate outliers with autoencoder networks in anomaly detection at hamich exploration area, east of iran," *Journal of Geochemical Exploration*, vol. 180, pp. 15–23, 2017.

[7] B. L. Betechuoh, T. Marwala, and T. Tettey, "Autoencoder networks for hiv classification," *Current Science*, vol. 91, pp. 1467–1473, 2006.

[8] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," 2015. [Online]. Available: http://dm.snu.ac.kr/static/docs/TR/SNUDM-TR-2015-03.pdf

[9] A. Oppermann. (2020, May) Activation functions in neural networks. [Online]. Available: https://www.deeplearning-academy.com/p/ai-wiki-activation-functions

[10] K. Janocha and W. Czarnecki, "On loss functions for deep neural networks in classification," *Schedae Informaticae*, vol. 25, 02 2017.

[11] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[12] R. Caruana, S. Lawrence, and L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, ser. NIPS'00. Cambridge, MA, USA: MIT Press, 2000, p. 381–387.

[13] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Workshop on Unsupervised and Transfer Learning*, I. Guyon, G. Dror *et al.*, Eds., vol. 27, University of California, Irvine. Irvine, CA 92697-3435: Department of Computer Science, University of California, Irvine, 2012, pp. 37–50.

[14] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *MLSDA'14: Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, A. Rahman, J. Deng, and J. Li, Eds., vol. 1, Association of Computing Machinery. Australia QLD, Gold Coast, Australia: Association of Computing Machinery, 2014, pp. 4–11.

[15] T. Sauer, *Numerical analysis*, 2nd ed. Harlow, Essex: Pearson, 2017, ch. 4, pp. 188–208.

[16] P. Händel, R. Ottosson, and H. Hjalmarson, *Signalteori*, 3rd ed. Stockholm, Sweden: Sinit, 2002, ch. 2, pp. 21–33.

[17] M. Abadi, A. Agarwal, P. Barham *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," Nov. 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[18] —. (2019, Sep.) Keras: The python deep learning library. [Online]. Available: https://keras.io/

[19] Y. LeCunn, C. Cortes, and C. J. Burgess. (2013, May) The mnist database of handwritten digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/