TÉCNICO LISBOA

Unsupervised Anomaly Detection in Water System Networks using Recurrent Neural Networks

Ana Margarida Pataca da Costa

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisors: Prof. Susana de Almeida Mendes Vinga Martins
Prof. Rui Miguel Carrasqueiro Henriques

Examination Committee

Chairperson: Prof. Paolo Romano
Supervisor: Prof. Susana de Almeida Mendes Vinga Martins
Member of the Committee: Prof. Francisco António Chaves Saraiva de Melo

July 2020

Acknowledgments

I would like to thank my supervisors, Prof. Rui Henriques and Prof. Susana Vinga, for all the support and assistance. I would like to thank Bruno Ferreira for is insight on the field of hydraulics. I want to also acknowledge my colleage Susana Gomes for the partnership when we two were go through the same. I would like to thank Paulo Lemos-Alves and Bernardo Furet, my classmates. All this obstacles along the way were worthwhile. I would also like to thank my friends Zahara Abibe, Sofia Cansado, Ricardo Cardoso and Maria Nunes. You were always there for me. In addition, I would like to thank my parents. I would not have come this far without them.

Abstract

Water is an indispensable resource for society. A major task associated with the proper management of water distribution systems is to detect anomalies to support decision-making and make contingency plans. Interesting anomalies can be bursts, unusual demands, and illegal consumption. Water leakage detection and location is a very difficult problem, due to the lack of information about the water system, and a leak might not be easily detected or confused with other events. The methodology proposed detects anomalies in the water system distributions, with a focus on bursts, through the use of deep learning architectures, in particular, encoder-decoder architectures based on LSTM such as LSTM autoencoder, CNN-LSTM, CNN-BiLSTM, and SCB-LSTM. Predictions are adjusted by using a temperature correction model. The water system studied is Quinta do Lago, located in Portugal, Algarve region. Infraquinta is the entity that manages the infrastructures. Simulation analysis and experimental results in real data show the pitfalls of the unsupervised anomaly detection task in water distribution systems. It also highlights that the proposed methodology, although yielding some properties of interest, needs to be complemented with additional principles to the targeted end. Finally, it pinpoints meaningful differences between recurrent architectures.

Keywords

Time series data analysis; Anomaly detection; Water management system; Recurrent neural network;

Resumo

Água é um recurso indispensável para a sociedade. Uma tarefa essencial associada com a gestão adequada de um sistema de distribuição de água é a detecção de anomalias de forma a apoiar a tomada de decisões e fazer planos de contigência. Anomalias interessantes podem ser fugas, consumos incomuns ou ilegais. A detecção e localização de fugas é um problema complexo devido à falta de informação sobre o sistema de água, e uma fuga pode não ser facilmente detectada pelos sensores ou confundida com outro tipo de evento. A methodologia proposta detecta anomalias em sistema de distribuição de água, com foco em fugas, através de arquitecturas *deep learning*, em particular em arquitecturas *encoder-decoder* baseadas em *LSTM* como por exemplo *LSTM autoencoder*, *CNN-LSTM*, *CNN-BiLSTM*, e *SCB-LSTM*. As previsões são ajustadas usando um modelo de correcção de temperatura. O sistema de distribuição estudado localiza-se na Quinta do Lago, Portugal, Algarve. A Infraquinta é uma entidade privada que faz a gestão destas infrasestruturas. A análise de dados simulados e resultados experimentais em dados reais mostram a dificuldade da tarefa de detecção de anomalias não supervisionada em sistemas de distribuição de água. Este estudo também destaca que a metodologia proposta, embora produza alguns resultados de interesse, precisa ser complementada com princípios adicionais para o fim pretendido. Finalmente, este estudo também permitiu identificar diferenças significativas entre arquiteturas recorrentes.

Palavras Chave

Análise de dados temporais; Detecção de anomalias; Gestão de redes de água; Redes neuronais recorrentes;

Contents

List of Figures

List of Tables

Acronyms

WDS	Water Distribution System
WME	Water Management Entities
DMA	District Metering Area
SCADA	Supervisory Control and Data Acquisition
NN	Neural Network
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
LSTM	Long-Short Term Memory
GRU	Gated Recurrent Unit
BAM	Bidirectional Associative Memories
VAE	Variational Autoencoders
SAE	Sparse Autoencoder
BN	Batch Normalization
PCA	Principal Component Analysis
HZM	Hour Zone Measurement
MNF	Minimum Night Flow
ROC	Receiver Operating Characteristic

1

Introduction

Contents

Water, once an abundant natural resource, is becoming more valuable due to climate change and over-exploitation [2]. Challenges that will be faced with climate change are droughts, flash floods, higher air temperature, and increased household water demand in the hot season [3]. In this context, it is necessary to make systems more resilient to climate changes and prepare contingency plans. Thus, there is an increasing need for optimizing water supply. This can be achieved by proper water supply management.

A Water Distribution System (WDS) is a network of pumps, pipelines, storage tanks, and other appurtenances. WDS collects a large amount of data (e.g. pressure, flow), which needs to be treated to generate useful information, not only for daily control, operation, and management of systems but also for supporting the current and future planning of the urban water infrastructures.

Water Management Entities (WME), also referred to as water utilities, are responsible for the operation of a WDS. Among other services, WMEs provide drinking water and wastewater services (including sewage treatment) to residential, commercial, and industrial sectors of the economy. Typically, public entities operate WDSs.

In Portugal, municipalities are responsible for the WDS. The data to be analyzed in this project are mostly time series and was provided by three public water management entities of different sizes and features. The time series is collected by the Municipality of Barreiro, Municipality of Beja and by the public company Infraquinta, which is partly owned by the Municipality of Loulé (Algarve). The Municipality of Barreiro (CMB) manages its urban water distribution systems with automatic meter reading (AMR) in district metering area (DMA) and manual metering at its clients. The Empresa Municipal de Água e Saneamento de Beja (Beja EMAS) manages an urban system and small rural systems, using AMR in some domestic clients with different technologies. Infraquinta manages the water supply system of Quinta do Lago, a luxury tourist destination. Infraquinta represents both a water management entity of a touristic zone, dealing with high seasonality in water consumption. Their WME is sophisticated, having hourly AMR in all its consumers. There is an alarm system implemented that is activated when maximum thresholds are crossed.

The classic definition of an outlier is given by Hawkins [4] who defines it as "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism". In WDS, outliers in data might be caused by an unusual household or non-household water consumption, leakages, changes in network system operation, sudden system faults (e.g. breaks in pipelines or service connections), meter malfunction, or problems with the telemetry or supervisory control and data acquisition (SCADA) system [5].

Water leakage in distribution systems is typically classified into background leakage and burst related leakage [6]. Burst type leakage is characterized by quick pressure drop and can be easily detected by the pressure sensors. Unusual increases in flow levels can also indicate bursts. When a burst occurs, a negative pressure wave is observed that travels in both directions away from the bursting point and is

reflected at the pipeline boundaries. The negative pressure is generated inside the pipe, propagating to the upstream and downstream, being detected by the pressure sensors [7]. They often are visually noticed, being reported by public or utility workers, thus the repair time is faster. Background leakage concerns the outflow from small cracks or deteriorated joints. They are not characterized by quick pressure drop and are not detectable by measuring instruments. Consequently, they go unreported for a long time. The most used operational method is the analysis of minimum night flow (MNF) data. MNF is the flow that occurs at night between 1 until 4 h when customer demands also are at its minimum. This implies that existent leakages (bursts and background leaks) are responsible for a large proportion of the existent flow at these hours [8]. The analysis of minimum night flow may help to identify leaks and unusual consumption.

Currently, leakage assessment techniques coupled with operational techniques are used by the WME to monitor the WDS. These techniques are generally offline and involve the application of mass balance type calculations or the monitoring of changes in night flow. These methods imply planning tests in advance, sometimes involving shutting down some parts of the WDS to make simulations, thus having a considerable cost of manual labor. Interpreting all data coming from the SCADA sensors is a complex task due to [9]:

- huge data volume;
- presence of missing data, noise, and gross errors;
- the variation consumption pattern being specific to a particular location, often with a predictable but changing fingerprint due to various forms of seasonality (daily, weekly and seasonal)
- inaccuracies in hydraulic models due to imprecise and outdated asset information, poor calibration, and lack of system operational feedback.

In order to reduce the efforts and costs associated with the identification and characterization of outlier events, there is a vast work in this field on traditional machine learning methods, either in an online or offline manner. Examples of traditional models for anomaly detection are Support Vector Machine (SVM) [9, 10], Principal Component Analysis (PCA) [11] and various types of neural networks [12–14]. These methods have reasonable results. Usually, traditional machine learning techniques require separate data pre-processing before training, which tends to be very time-consuming and often requires domain knowledge [15].

The deep learning field provides principles to allow a machine to be fed with raw data and automatically discover the representations needed for regression and classification [16]. Generally, deep learning methods outperform traditional machine learning methods when a large amount of data is available [16]. Traditional machine learning failed to generalize in other tasks such as recognizing speech or recognizing

objects [16]. The development of deep learning was motivated in part by the failure of traditional algorithms to generalize well on such tasks [16]. Recent deep learning approaches have shown to perform well on raw time series data, eliminating the need for pre-processing. Common deep learning architectures have convolutional layers and recurrent layers. Recurrent neural networks (RNN) have the advantage to persist information for later use in the network. This makes them particularly suited for the analysis of temporal data. Convolutional neural networks (CNN) are appropriate for automating spatial feature extraction from time series raw data through sensory signals [17]. Only recently, it was proposed models based on recurrent neural networks (RNN) for time series anomaly detection [18]. Long-Short Term Memory (LSTM) has gained popularity, due to automatic feature extraction abilities, to represent the relationship between a current event and previous events, and handle of time series problems [19]. Commonly, LSTM is not used alone, they are often used in hybrid models, particularly in encoder-decoder architectures. There is a scarcity of works aiming at detecting anomalies on this type of data.

1.1 Objectives

This works aims to compare multiple deep learning architectures, joining the efforts of multiple works, without the need for major preprocessing. These architectures will be used as reconstructions models. Thus, first the models are trained with normal sequence data. If the deviation is bigger than the threshold, then an anomaly is detected. Typically, these models are tested in simple anomaly detection scenarios, where data is labeled, the environment is more static, and there are few different types of anomalies. This research addresses this gap by examining the performance of these models in a complex and dynamic scenario such as anomaly detection in water system networks, in an unsupervised way. The events occurring in the WDS are reported by workers on site. Workers report events such as bursts or maintenance activities. Not all events are reported or are reported with incomplete information. Burst events are usually reported with the date of perception, date when water was cut off and turned on again, and infrastructure intervened. Thus, is not possible to know exactly when or where a burst occurred. These approaches are further assessed using synthetic data too. Since real data is not always labeled with the target events of interest (such as bursts) and labeling is time-consuming and expensive, artificial sequences can be generated by a mathematical model and considered to make parameter adjustments and make preliminary results. Furthermore, it is hard to differentiate a consumption from a leak event, or valves opening or closing, due to maintenance activities, from a burst. The target problem is intentionally difficult to attack from this unsupervised stance. Thus, good results are not expected. The main objective of this research is to test whether anomalies characterized by leaks can be isolated or not. In the future, the target approaches can help to reduce the number of false alarms on the already existent alarm system or help a domain expert, to make decisions or to find interesting patterns.

1.2 Contributions

The major computational contributions of this thesis are the following:

- Detection of anomalies on time series using recurrent neural networks.
- Comparison between deep learning models: CNN-LSTM, CNN-BiLSTM, SCB-LSTM, stacked LSTM, and BiLSTM.
- Combine the advantages of CNN and Bi-LSTM models.
- Exploration and preprocessing of the gathered time series from real and simulated WDSs.
- Create a temperature correction model to correct predictions from deep learning models.
- Principles to set thresholds in reconstruction models.

The major applied contributions to the management of water distribution systems are:

- Detection of anomalous events of potential interest, including leaks, unusual demands, and system events.
- Early detection of pipe bursts. Detection windows are smaller than the average time of perception.
- Thresholds to relax the already existent alarm system.

1.3 Organization of the Document

This thesis is is organized as follows:

- chapter 2 presents definitions of key topics of water system management and deep learning.
- chapter 3 offers a compilation of related work.
- chapter 4 describes the proposed solution.
- chapter 5 presents and discusses results in simulated data and real data.
- chapter 6 summarises the main conclusions of this work, as well as future directions.

2

Background

Contents

2.1 Hydraulic concepts

This section provides essential background about hydraulics systems. First, basic concepts about components of the WDS are introduced, followed by methods of acquisition and modeling of data.

Water System: A system of supplying water, as throughout a metropolitan area. The system includes pumps, pipelines, and nodes (junctions, reservoirs, or tanks).

Telemetry System: A collection of meters and monitoring devices that relay information about flows and pressures at some discrete points of the distribution network.

Node Pressure: Any pressure data collected at pressure monitoring sensors in the distribution system. The pressure is the force per unit area applied in a direction perpendicular to the surface of an object.

Pipe Flow: Any in-line flow data measured in the distribution system. According to the law of conservation of matter, for permanent conditions (i.e. not transient conditions such as water hammer), the mass flow rate is constant along a stream.

Pressure Sensor: Used to measure the pressure in pipes where water is flowing – for example, in water distribution systems, to automatically determine whether pumps need to be activated to increase the flow rate. An example of static water pressure is measuring the water level in a storage tank, and an example of dynamic water pressure is measuring the water pressure in a water line in the distribution system, to monitor transient pressures or pipe bursts.

Flow Sensor: To monitor the amount of water being supplied and used, the rate of flow of water has to be measured. Water flow sensors are used for this purpose. Water flow sensors are installed at the water source or pipes to measure the rate of flow of water and calculate the amount of water flowed through the pipe. The rate of flow of water is measured as liters per hour or cubic meters.

Valves: There are essentially two types of valves: pressure regulating valves and flow control valves. Pressure-regulating valves have different functions: from keeping system pressures safely below a desired upper limit to maintaining a set pressure in part of a circuit. The purpose of flow control in a hydraulic system is to regulate speed.

Pump: A pump produces liquid movement or flow.

Leakage: In WDS, leakage through pipes has two major types, *burst* and *background* type leakages [6].

Burst type leakage is characterized by quick pressure drop and can be easily detected by the pressure sensors. They often are visually noticed, being reported by public or utility workers, thus the repair time is faster.

Background leakage concerns the outflow from small cracks or deteriorated joints. They are not characterized by quick pressure drop and are not detectable by measuring instruments. Consequently, they go unreported for a long time.

Leakage in distribution systems can be caused by several different factors [20]. Some examples include bad pipe connections, internal or external pipe corrosion, or mechanical damage caused by excessive pipe load (e.g. by traffic on the road above or by a third party working in the system). Other common factors that influence leakages are ground movement, high system pressure, damage due to excavation, pipe age, winter temperature, defects in pipes, ground conditions, and poor quality of workmanship. Therefore, the presence of leakage may damage the infrastructure and cause third party damage, water and financial losses, energy losses, and health risks. Leakage is dependent on system pressure. The higher the pressure, the larger the leak flow and vice versa [20].

2.2 Acquisition and modeling of data

Supervisory Control and Data Acquisition (SCADA): The Supervisory Control and Data Acquisition (SCADA) system is used to manage in real-time the water supply system in a utility. This can be done by monitoring the whole system from water sources to the customer. All the operations are done at the command center allow remote control of the WDS. Examples are to make setpoint changes on distant process controllers, to open or close valves or switches, to monitor alarms, and to gather measurement information [21].

EPANET: EPANET is a software application used throughout the world to model water distribution systems. EPANET abstracts an actual distribution system into a network of links connected at their endpoints called nodes.

Links can represent pipes, pumps, or control valves (shutoff valves and check valves are considered to be properties of pipes and are not represented as separate valves). Nodes can be junctions, reservoirs, or tanks. Junctions are points where pipes join together and where water either enters or leaves the network. Reservoirs represent fixed head boundary locations, such as treatment works, groundwater aquifers, or tie ins to other networks or pressure zones. Tanks are storage facilities where the volume and water level can change over time.

EPANET receives input data on the network being modeled through a text file written using a Problem Description Language (PDL).

Modeling leakage in EPANET: Although EPANET is primarily designed for modeling network supply and water quality issues, the emitter property in EPANET, which is designed to model fire hydrants and sprinklers can be adapted to model leaks [10]. The emitter behavior equation is very simple [22]: $Q = K \times P^N$ where Q is the leakage flow rate at node j, P is the pressure, K is the emitter coefficient, and N is the pressure exponent.

2.3 Time Series data analysis and Machine Learning

Time Series data: A time series is a set of observations x_t, each one being recorded at a specific time t [23]. A discrete-time time series is one in which the set T_0 of times at which observations are made is discrete, as is the case, for example, when observations are made at fixed time intervals. Continuous time series are obtained when observations are recorded continuously over some time interval, e.g., when $T_0 = [0, 1]$. Discrete time series are often obtained by observing a continuous-time process at a discrete sequence of observation times. It is then natural, even though the observations are made at discrete times, to model the underlying process as a continuous-time series. Domain values are also described in terms of continuous/numerical or discrete/categorical.

Univariate time series: Refers to a time series that consists of single (scalar) observations recorded sequentially over equal time increments.

Multivariate time series: A multivariate time series is a series of m-dimensional vectors $(x_t[1], x_t[2], \ldots, x_t[m])'$ observed at point in time t (usually $t = 1, 2, 3, \ldots$) [23]. The multivariate time series has not only serial dependence within each component series $\{x_t[i]\}$ but also interdependence between the different component series $\{x_t[i]\}$ and $\{x_t[j]\}$, $i \neq j$.

Time series data: Time series data is a set of multivariate time series.

2.4 Time series anomaly detection

Outlier: An outlier is classically defined as "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [4].

In WDS, outliers in flow data might be due to [12]:

- unusual water demands (e.g. holidays, summer season)
- changes in network system operation - pump or valve maneuvres.
- sudden system faults such as breaks in pipelines or service connections.

- meter malfunction, or problems with the telemetry or supervisory control and data acquisition (SCADA) system (including local meter processing, local storage, transmission, and final data storage issues). These are noticeably different from the rest of the data.
- pipe bursts or leaks.
- illegal consumption.

Generally, outliers are grouped into three major categories [24]:

- Global outliers (or point anomaly): A data point significantly deviates from the rest of the data set.
- Contextual outliers (or conditional outlier): it a data point deviates significantly based on a selected context
- Collective outliers: A subset of data objects that collectively deviate significantly from the whole data set, even if the individual data objects may not be outliers.

Time series anomaly detection is the process of finding outlier data points relative to some normal or usual time series.

Neural networks are adaptive statistical models based on an analogy with the structure of the brain [25]. They are built from simple units (neurons). These units are interlinked by a set of weighted connections. Learning is achieved by adjustment of the connection weights. Each unit codes correspond to a feature or characteristic of a pattern that we want to analyze or predict. The units are organized in layers.

Autoencoder: An autoencoder is an artificial neural network that applies an unsupervised learning algorithm that uses backpropagation, setting the target values to be equal to the inputs [26]. An autoencoder aims to produce an output that reconstructs its input with noise removed.

A classic autoencoder is a feedforward fully-connected neural network, where the number of neurons in the input layer and the output layer is the same and where there are much fewer neurons in the hidden layers than in the input and output layers [27].

Internally, it has a hidden layer h that describes a code used to represent the input (latent representation). This kind of network is composed of two parts: an encoder function $h = f(x)$ and decoder that produces a reconstruction $r = g(h)$ [28].

If the only purpose of autoencoders was to copy the input to the output, learning to set $g(f(x)) = x$ they would be useless. Instead, autoencoders are designed to be unable to copy perfectly [28]. Usually, they are restricted in ways that allow them to copy only approximately, and to copy only input that

resembles the training data. Because the model is forced to prioritize which aspects of the input should be copied, it often learns useful properties of the data.

This can be achieved by creating constraints on the copying task. One way to obtain the most salient features from the training data is to constrain h to have smaller dimensions than x, in this case, the autoencoder is called *undercomplete*.

If the autoencoder is given too much capacity (high ability to model complex relationships), it can learn to perform the copying task without extracting useful information about the distribution of the data [28]. This can occur if the dimension of the latent representation is the same as the input and in the *overcomplete* case, where the dimension of the hidden code is greater than the input.

Another method to constrain the reconstruction of an autoencoder is to use *regularized autoencoders* [28]. Rather than limiting the model capacity by keeping the encoder and decoder shallow and the code size small, regularized autoencoders use a loss function that encourages the model to have other properties besides the ability to copy its input to its output. In general, there are two types of regularized autoencoder: the *sparse autoencoder* and the *denoising autoencoder*. A sparse autoencoder is an autoencoder whose training criterion involves a sparsity penalty $\Omega(h)$ on the code layer h , in addition to the reconstruction error. This means that a regularization term is added to the loss function. This penalizes activations of hidden layers, so only a few nodes are active. A denoising autoencoder is an autoencoder whose reconstruction error term of the loss function is changed. This can be done by adding some noise of the input and make the autoencoder learn to remove it.

Convolutional Neural Network: A convolutional neural network (CNN) is a type of deep neural networks specialized in processing data with a known grid-like topology [28]. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.

Recurrent Neural Network: Recurrent Neural Networks (RNN) is a type of neural networks for processing sequential data [28]. RNN differs from standard neural networks by allowing the output of hidden layer neurons to feedback and serve as inputs to the neurons. In this way, the network can use the past as a way to understand the sequential nature of the data.

The notation for the hidden state update is:

$$\mathbf{h}_t = \phi\left(W\mathbf{x}_t + U\mathbf{h}_{t-1}\right) \tag{2.1}$$

The hidden state at time step t is h_t. It is a function of the input x_t at the same time step t, modified by a weight matrix W (like the one we used for feed-forward nets) added to the hidden state of the

previous time step, h_{t-1} , multiplied by its own hidden-state-to-hidden-state matrix U, otherwise known as a transition matrix and similar to a Markov chain. The weight matrices are filters that determine how much importance to accord to both the present input and the past hidden state. The error they generate will return via back-propagation and be used to adjust their weights until error can't go any lower.

The sum of the weight input and hidden state is squashed by the function ϕ which condensates very large or very small values, as well as making gradients workable for back-propagation. Because this feedback loop occurs at every time step in the series, each hidden state contains traces not only of the previous hidden state but also of all those that preceded h_{t-1} for as long as memory can persist.

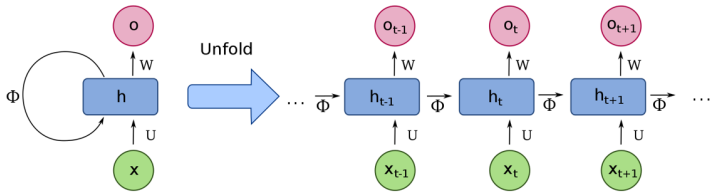

Figure 2.1: The unrolled recurrent neural network, adapted from the original figure of François Deloche [CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0)]

LSTM: A Long Short-Term Memory (LSTM) network is a type of recurrent neural network. LSTMs aim at facilitating the learning of long term dependencies in the input [29].

LSTMs also have this chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

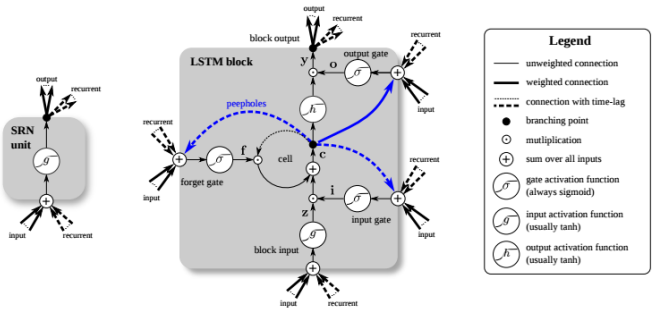

Figure 2.2: Schematic of Simple Recurrent Neural Network (left) and LSTM (right) used as hidden layers of a recurrent neural network. Figure from Greff et al., 2017 [1]

The LSTM architecture consists of a set of recurrently connected structures called memory blocks. Each block contains one or more self-connected memory cells each with an associated cell state. The memory block is built around the cell(s) which ensure constant error flow through them by using an identity function and always getting incoming unit weights. Thus these units solve the vanishing/exploding gradient problem commonly observed in traditional recurrent neural networks [30, 31]. The LSTM

mitigates this problem via input, forget, and output gates; the input gate regulates how much of the new cell state to keep, the forget gate regulates how much of the existing memory to forget, and the output gate regulates how much of the cell state should be exposed to the next layers of the network.

LSTM avoids the need for a pre-specified time window and is capable of accurately modeling complex multivariate sequences.

LSTM has two limitations. First, the number of memory cells is linked to the size of the recurrent weight matrices. An LSTM with N_h memory cells requires a recurrent weight matrix with $O\left(N_h^2\right)$ weights. Second, LSTM is a poor candidate for learning to represent common data structures like arrays because it lacks a mechanism to index its memory while writing and reading [32].

Gated Recurrent Unit: Gated Recurrent Units (GRU) are special variants of LSTMs that merge the forget and input gates into a single update gate resulting in a simpler model than standard LSTM models [33]. The GRU operates using a reset gate and an update gate. The reset gate sits between the previous activation and the next candidate activation to forget the previous state, and the update gate decides how much of the candidate activation to use in updating the cell state.

Both LSTMs and GRUs can keep memory/state from previous activations rather than replacing the entire activation like RNNs, allowing them to remember features for a long time and allowing backpropagation to happen through multiple bounded nonlinearities, which reduces the likelihood of the vanishing gradient. LSTMs control the exposure of memory content (cell state) while GRUs expose the entire cell state to other units in the network.

Associative Memories is a type of memory that allows for the recall of data based on the degree of similarity between the input pattern and the patterns stored in memory [34]. A network with associative memories stores mappings from specific input representations to specific output representations. The linear associator is the first studied and simplest associative model. It is a feedforward type network where the output is produced in a single feedforward computation. There are two types of associative memory: auto-associative and hetero-associative.

Heteroassociative memories recall an associated piece of data from one category upon the presentation of data from another category. *Autoassociative memories* are capable of retrieving a piece of data upon the presentation of partial information only. Auto-associative mapping can be created by training an ANN to reproduce its input at its output [35]. A set of reference signal patterns (e.g. parts of a time series) are learned by the auto-associative network. Recurrent networks allow recurrences through feedback connections. This feature is used in associative memories such as the Bidirectional Associative Memory and Hopfield network.

Hopfield Networks: The Hopfield model was proposed by John Hopfield of the California Institute of Technology during the early 1980s. The dynamics of the Hopfield model are different from that of the linear associator model in that it computes its output recursively in time until the system becomes stable [36]. Boltzmann Machines and Deep Belief Networks are paradigmatic neural networks built upon Hopfield's work.

Bidirectional Associative Memory: Bidirectional associative memories (BAM) are artificial neural networks that have long been used for performing a heteroassociative recall. Given a pattern (e.g. parts of a time series), BAMs return another pattern that may have a different window length [37].

3

Related Work

Contents

This chapter displays a compilation of relevant work to the targeted problem. Section 3.1 presents different outlier detection methods applied to time series produced by sensor systems in general. Section 3.2 covers outlier detection techniques performed in distribution networks, mainly water. Both sections focus on deep learning architectures.

3.1 Outlier analysis in time series

Many types of systems are equipped with sensors to monitor them, producing large volumes of time-ordered observations that form time series [38]. Such time series are produced widely and are used in many application domains: industrial, finance, biology, transportation, and healthcare [38]. This section, briefly describes previous work on anomaly detection on sensory data, with a special focus on RNN.

A few examples of statistical machine learning methods applied to outlier detection:

1. One-class SVM. For anomaly detection, there is a semi-supervised variant of one-class SVM. Only normal data is required for training before anomalies can be detected [39].

2. Local Outlier Factor (LOF). A well-known density-based outlier detection method [40].

3. A kernel-based method Isolation Forest (ISF) [41]. ISF is a randomized clustering forest.

4. Matrix Profile I (MP). MP is considered to be the state-of-the-art similarity-based outlier detection method [42].

5. Dynamic Bayesian Network. Serras proposed a method based on Dynamic Bayesian Network for multivariate time series [43].

Until recently, only a few works in literature proposed the use of neural networks in outlier detection. Neural Networks have computational challenges such as overfitting and slow use when using large data sets [18, 44]. Many methods have been proposed and investigated for these problems. Overfitting occurs when a model adjusts excessively to the training data, seeing patterns that do not exist, and consequently performing poorly in predicting new data. This can be overcome by changing connectivity between layers (dropout) [18, 44] and other implicit or explicit regularization strategies [45]. Training deep neural networks was also challenging due to the vanishing gradient problem that implied that weights in layers close to the input layer were not updated in response to errors calculated on the training dataset. An important milestone of the field of deep learning was greedy layer-wise pretraining [46] that allowed very deep neural networks to be successfully trained, achieving state-of-the-art performance. The development of deep learning was motivated in part by the failure of traditional algorithms to generalize well on tasks such as recognizing speech or recognizing objects [16]. Until recently, deep learning has been applied to fault or anomaly detection due to these developments over the years.

In particular, Recurrent Neural Networks (RNN) have received some attention [18]. Using RNN the prediction of a current event will depend on both the current event and its previous events. This type of neural network is often used in an unsupervised context.

The majority of models based on RNN are trained on normal sequential data. To detect anomalies, the degree of deviation between observed data and the normal state is then computed. An advantage of this method is that anomalous data are not needed to build the model, which is harder to get. The encoder-decoder was trained with normal instances during training and learned to reconstruct them. When given an anomalous sequence, the model may not be able to reconstruct it well, and hence would lead to higher reconstruction errors compared to the reconstruction errors for the normal sequences. This method has a disadvantage that it depends a lot on the chosen threshold. Searching the ideal threshold can be very exhaustive even though it gives some freedom to adjust it according to the needs of the system e.g. to generate less false alarms.

An increasingly popular type of RNN is Long-Short Term Memory network (LSTM) due to their capacity for classifying, processing, and making predictions based on time series data. Intuitively, this means that when given certain input samples, they can remember the context of the samples, and predict a coherent output according to that context, thus learning the behavior of the training set. LSTM has shown its advantages in numerous applications through dealing with the exploding and vanishing gradient problem and keeping short-term "memory" for a long time [47]. LSTM compared with traditional RNNs and other sequence learning methods such as hidden Markov models perform better [48]. Typical architectures are commonly used with LSTM: stacked LSTM, Bi-LSTM, and LSTM encoder-decoder architectures.

Stacked LSTMs or Deep LSTMs were introduced by Graves, et al. in their application of LSTMs to speech recognition, beating a benchmark on a challenging standard problem. In the same work, they found that the depth of the network was more important than the number of memory cells in a given layer [49].

Graves et al. [19] proposed a stacked LSTM for anomaly detection in time series. A network is trained on normal data and used to forecast.

An encoder-decoder LSTM architecture is a model comprised of two sub-models: one called the encoder that reads the input sequences and compresses it to a fixed-length internal representation, and an output model called the decoder that interprets the internal representation and uses it to predict the output sequence. When the series is inherently unpredictable, the LSTM encoder-decoder can produce superior results in comparison to the LSTM predictor [50]. Typical LSTM encoder-decoder architectures are CNN-LSTM and LSTM autoencoders.

Classic autoencoders based on feedforward neural networks are often used for non-sequential data [18]. To perform outlier detection in sequential data such as time series, autoencoders based on recurrent neural

networks are proposed while reusing the idea that large reconstruction errors indicate outliers [50]. The encoder-decoder model was trained with normal instances during training and learned to reconstruct them. When given an anomalous sequence, the model may not be able to reconstruct it well, and hence would lead to higher reconstruction errors compared to the reconstruction errors for the normal sequences. A LSTM-based Autoencoders [50] are the state-of-the-art deep learning outlier detection method for time series. Autoencoder ensembles aim to further improve the accuracy of outlier detection based on autoencoders [18] but were only applied to non-sequential data until recent years [27]. The main idea is to build a set of autoencoders and to consider the reconstruction errors from multiple autoencoders when detecting outliers. Using a set of classic, fully-connected autoencoders is not helpful since the network structures are the same across the different autoencoders. Instead, for each autoencoder, it is helpful to randomly remove some connections to obtain a sparsely-connected autoencoder. Then, an autoencoder ensemble consists of multiple sparsely-connected autoencoders with different network structures, which helps reduce the variances of the overall reconstruction errors.

Despite the success of LSTMs for sequence modeling, they still can not integrate information from future timesteps. To solve this problem, Bidirectional Long Short-Term Memory networks (Bi-LSTMs) [51] were proposed. Bi-LSTM can capture the past and future contexts in time series sequence data in case of the signals of machinery.

Lee et. al [17] proposed a Convolutional Neural Network layer, Bidirectional and Unidirectional Long Short-Term Memory Recurrent Neural Networks, which is one of a novel deep architecture named stacked convolutional bidirectional LSTM network (SCB-LSTM). The baseline models for which SCB-LSTM was compared was stacked uni-LSTM and stacked bi-LSTM. SCB-LSTM outperformed these baselines.

The combination of CNN and LSTM layers is being studied to extract temporal and spatial features. Since speech recognition and natural language processing have temporal and spatial information, the combination of CNN and LSTM can effectively extract features. Currently, these advantages are being applied to classifying and predicting sensor data occurring in the industrial domain [52]. A CNN-LSTM is proposed to predict power demand [53].

Canizo et al. [15] proposes a deep learning based approach for supervised multi-time series anomaly detection that combines a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN) in different ways. Unlike other approaches, it uses independent CNNs, so-called convolutional heads, to deal with anomaly detection in multi-sensor systems. Each sensor is addressed individually avoiding the need for data pre-processing and allowing for a more tailored architecture for each type of sensor. This architecture is called Multi-head CNN–RNN. CNN is also typically used with autoencoders. Convolutional Neural Networks based Autoencoder (CNN) [54], which treats time series as images and employs a CNN autoencoder to reconstruct the image.

3.2 Water distribution network data analysis

In general, there are three categories of methods for detecting pipeline leakage, namely leakage assessment techniques, operational techniques, and machine learning techniques [20]. Among conventional solutions to the burst detection and location, WMEs currently use leakage assessment techniques to assess leakages coupled with operative methodologies [12, 20].

The objective of leakage assessment (i.e. water audit) is to estimate the quantity of water loss in the system analyzed without worrying where the leaks are located [20]. The assessment methods developed so far can be broadly classified into (1) top-down or (2) bottom-up.

The objective of top-down leakage assessment approaches is to estimate the leakage in a particular system by evaluating different components of the overall water balance, primarily the water consumed for different purposes. Generally, the equation used to this end:

$$\text{Volume of Real Losses} = \text{System Input Volume} - (\text{Authorized Consumption Volume} + \text{Apparent Losses Volume})$$

Bottom-up procedures are implemented when the company has confirmed the data used in the top-down part. Bottom-up real loss assessment can be carried out in two different ways [12, 20]:

1. 24 Hour Zone Measurement (HZM)
2. Minimum Night Flow (MNF) analysis

HZM needs a temporary isolated area of the distribution network that is supplied from one or two inflow points only. In these areas, 24h inflow measurements shall always be logged along with pressure measurements.

MNF in urban situations normally occurs during the early morning period, usually between 01:00 and 04:00 h. The estimation of the real loss component is carried out by subtracting legitimate night uses from the MNF. By monitoring MNF, unusual changes in water volumes can be detected.

Operative methodologies locate where the leak is. Operative methodologies usually employ highly specialized hardware equipment, such as leak-noise correlators and pig-mounted acoustic sensors.

Many approaches from the fields of artificial intelligence and statistics have been applied for detecting abnormality in WDSs from time series data. At the present, in the literature, there are mainly traditional models for anomaly detection in WDS, such as support vector machine (SVM) [9, 10], Principal Component Analysis (PCA) [11] and various types of neural network (NN) [12]. For anomaly detection in WDS, methods based on deep learning have currently been the focus of researchers [7, 55].

Alert systems that convert flow and/or pressure sensor data into usable information in the form of timely alerts have been developed with a focus on burst detection to help with the issue of leakage reduction [14]. One of the disadvantages of the methods applied either in traditional machine learning or deep learning is the need to make a database of interesting patterns or leak labeling a historical

database [13, 56]. To create the database there is the need for domain knowledge. The database also needs to be large. Another disadvantage is time and resources spent at preprocessing the timeseries [7,57]. To avoid the cost and time of labeling data by experts, many approaches use simulated data [12,58]. Since real data is not always labeled with the target events of interest (such as bursts) and labeling is time-consuming and expensive, many approaches use artificial sequences.

Mathematical hydraulic models can be used for the calibration of a water distribution system model. They can be used to simulate leakage in terms of pressure changes [12]. The problems to develop this type of model are: first, understanding of complex concepts related to the field of hydraulic, secondly, wasting time for deriving a suitable mathematical relation. Besides that, the model is very static. The model does not take into account dynamic changes and non-stationary environments.

To overcome the development of a complex hydraulic model, this type of model is instead replaced by a neural network that is trained with data generated by EPANET. The software EPANET simulates the hydraulic dynamics of the WDS (in terms of pressures at nodes and flows through pipes) generally in normal conditions (i.e., assuming no leakages are present). By changing the water demand, a reasonable subset of the possible operating conditions that may occur in the water network is created. It is preferred to use synthetic data because on real data there is no certainty if the sequence is normal or not. Besides that, labeling the data by experts is time-consuming and expensive.

The neural network makes the approximation of the dynamic behavior of selected physical variables in a water distribution network [59].

A conventional hydraulic simulation model of a water-distribution network created in EPANET can be described in discrete time as an input-output system. It generates the vectors:

- p_t - a vector of control variables representing n_p pump setting at time t;
- V_t - a vector of control variables representing n_v valve settings at time t;
- D_t - a vector of variables representing values of demands at the n_d consumer nodes in the network between times t and $t + \Delta t$;
- S_t - a vector of variables representing n_s storage tank water levels at time t;
- E_t - a vector of variables representing n_p power consumption of pumps during the interval between times t and $t + \Delta t$;
- H_t - a vector of variables representing values of pressures n_h at specific nodes during the interval between times t and $t + \Delta t$;
- Q_t - a vector of containing the values of flow n_q for specific pipes during the interval between times t and $t + \Delta t$;

- $S_{t+\Delta t}$ - a vector of variables representing n_s storage tank water levels at time $t + \Delta t$.

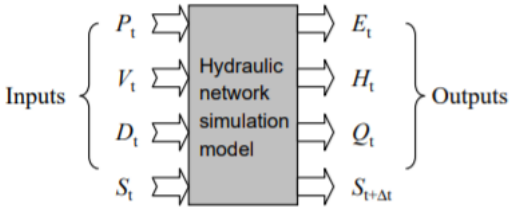

Figure 3.1: Input–output model of a water-distribution network
Source: Rao and Alvarruiz, 2007 [59]

The input set given to the ANN may contain the combination of pump/valve settings, demands, and initial water levels in storage tanks whilst the output set correspond to the power consumption of pumps, resulting in water levels in storage tanks, pressure, and flow rates at critical locations throughout the network [59]. The procedure for training and testing the ANN is [59]:

1. Model the WDS network using a conventional hydraulic simulation model.
2. Select critical points for storage levels, pressures, and flow rates.
3. Run model with different combinations of starting conditions, demands, and control settings.
4. Train ANN with generated input/output sets to predict output values at critical points.
5. Test ANN with generated input/output sets by comparing predicates values with observed.

Jalalkamali et. al in 2011 [58] described the application of a hybrid model of artificial neural network (ANN) coupled with a genetic algorithm (GA) and Radial Basis Function Neural Network (RBF). The ANNs used were feedforward networks (FFN) and recurrent neural networks (RNN). It was only used pressure data coming from 3 sensors to estimate leakage rates. GA is used to determine how many neurons to use in each hidden layer of the ANN. Obtained results show that the ANN-GA models can be used successfully to estimate leakage rates in water distribution networks. The research concluded that FNN-GA had the best results.

Mounce et al. [13,56] presented the online application of artificial neural network and fuzzy inference systems for the detection of leakage in real water distribution system. The model was used to monitor and analyze flow time series data from DMA flow meters in a water distribution network. The ANN model was trained with a continually updated historic database. It constructed a probability density model. The Fuzzy Inference System was used for classification (abnormal/normal). Compared observed flows with the probability density function of predicted flows over time windows such that confidence intervals could be assigned to alerts and provide an estimate of likely burst size. The AI analysis system

was applied to a validation data set (generated by simulated flushing) to verify the burst estimation capability, and then to data from eight months of historic operational flow data. The system did not differentiate abnormal events such as industrial demands, closure, and opening of valves from bursts. It still detected events of interest such as illegal consumption. 56% of alerts were correlated to definite bursts.

Mounce et al. [57] proposed a leakage detection method based on an artificial neural network to harmonize data obtained from different sensors to classify different types of leakage in a WDS. The developed methodology is based on sensor time series data and thus requires a large monitoring database.

A theoretical basis was described for modeling an arbitrary DMA flow sensor, allowing a prediction to be made of future values based on a lag of time series values. This concept was instantiated employing an Mixture Density Network (MDN) used to predict a mixture model representation of the conditional probability. A classification module analyzing this mixture model and the observed value over a suitable window provides a state classification. The configuration of a set of DMAs is described by a logic rules-base capable of assessing the context of normal/abnormal signals. This system was assessed by using a real data set from an experimental site in a water distribution system in which bursts were simulated by hydrant flushing.

In many of these research works, the location of the pipe burst is identified by using the arrival times and magnitudes of burst-induced transient waves at two or more points where the pressure sensors were stationed.

Analysis of the pressure gradient across a DMA during the series of burst simulations was also presented, based on using a seasonal differencing technique to measure the pressure drop present and how this varies over the spatial map. The resultant three-dimensional pressure drop map gave an accurate indication of the location of the flush. This type of pressure sensor data fusion is dependent on the deployment of pressure sensors within a DMA.

Zhou et al. in 2019 [7] proposes leak Detection and Location Based on improved spline-local mean decomposition (ISLMD) and CNN. ISLMD is a signal processing method. First, ISLMD is used to process the signal, thus eliminating noise interference. Second, CNN is used for leak detection, which can better address the problem. For the CNN, LSSVM and AlexNet were used. This method was tested only in a simulation environment, where leaks were generated. AlexNet requires a large amount of training. AlexNet can distinguish different leak apertures in different leak points, has better generalization and accuracy.

Hu et al. in 2019 proposed a hybrid model based on CNN and Bi-LSTM for urban water demand prediction [55]. The model CNN-Bi-LSTM combined the advantages of both models. The model was received as input water consumption data and climate data. After the model predicts water consumption, a temperature and holiday correction model is used. This temperature and holiday correction model is

based on statistical analysis. It receives 5 days of water usage data and the daily maximum temperature. After the temperature correction model, accuracy improved 1%-3%. After the holiday correction model accuracy improved 2%-5%.

The model CNN-Bi-LSTM was compared with other 5 models: long short-term memory networks (LSTM), bidirectional long-term memory networks (Bi-LSTM), CNN, sparse autoencoder (SAEs), and CNN-LSTM. It was concluded that CNN-Bi-LSTM had less prediction error that the others. The training time of CNN-Bi-LSTM is less than LSTM, Bi-LSTM, CNN, and CNN-LSTM, but larger than SAEs. The training convergence of CNN-Bi-LSTM was set in 125 times, which is smaller than the training times of the other five models.

In 2014, Mounce used a pattern matching technique and a binary associative artificial neural network for novelty detection [14]. AURA Alert software was considered to this end, which has been used for the detection of irregularities in highly complex assets in a variety of different industries [60].

First, a database of interesting patterns was manually built. To search these databases a similarity-based search is used so that similar-shaped profiles of different amplitudes are matched (within a given scaling factor). Matches that are over a given threshold score are returned. For this task of pattern matching, Signal Data Explorer Software (SDE) is used. SDE is a general purpose data browser and search engine for time series signal data. The SDE allows a user to specify the signal event to be searched by supplying a short example of that event (query by content). This can be specified using manually created examples, historical sample inputs, or examples imported from other systems. The user is then able to select the (possibly large) datasets for the search.

The type of binary neural network used is AURA. AURA is built on correlation matrix memories (CMMs), that store normal operating behavior. The advantages over a standard neural network are: high levels of data compression, one-pass training for online training, scalable architecture that can be readily mapped onto high-performance computing platforms, and a sound theoretical basis to determine the bounds of the system operation.

AURA Alert is the implementation of AURA within the SDE. The system will locate the nearest k matching patterns. The score associated with the most closely matching patterns can then be used to determine how different the current state is to any that have been seen before, to provide a measure of the novelty of the event.

For flow data analysis, the performance was comparable to the AI detection as reported in previous work from Mounce using an Artificial Neural Network and Fuzzy Logic system [13]. The pattern matching technique was not as good as using outlier detection based methods. It was shown that AURA Alert can rapidly learn and model the normal behavior for a system, with the ability to search through complex high-dimensional multivariate spaces to detect deviations from normal conditions. It can automatically calculate a continuous novelty score for every time step and hence enable the detection of any type of

event.

4

Solution

Contents

This thesis addresses the problem of detecting events of interest in time series through deep learning, particularly architectures based on Long-Short Term Memory (LSTM). Given time series data of flow and pressure collected from key nodes of the WDS, our main objective is to detect anomalous behavior in a semi-supervised way. For that matter, this chapter introduces important learning principles, together with an unsupervised deep learning approach, to answer the identified problem.

4.1 Case Study: Infraquinta

Consider the water distribution system managed by the WME Infraquinta. Given its touristic nature, this network as some particular characteristics. It shows high seasonality in water consumption. It has a high demand at night because of the irrigation system. Infrastructures are also brand new, yet not free of failures affecting data quality.

The SCADA system has 6 sensor pressures and 7 sensor flows. The number of sensor pressures was identified has to be scarce and not covering all zones. Due to that, research was more focused on flow data.

In this work, two different datasets were used, one with synthetic data and another with real data. Synthetic data was produced by the software EPANET by civil engineers. Simulated data has a granularity of 10 minutes. Every simulated leak has a duration of 4 hours. Simulated data were used for preliminary results since it was possible to know exactly when and where a leak occurred. It is also possible to know for sure the flow direction 4.1. This has high importance since if the flow is upstream a certain location leak, the flow sensor will notice more a flow increase. If the flow direction is downstream, the pressure sensor will notice it more, since there will be a pressure drop. The simulation was only done during summer because it is the time of the year with most water consumption in this particular water network. Leak locations were done in several parts of the network.

Real data has a granularity of 1 minute. The data used was from 2017. It is not possible to know exactly when or where a leak occurred. For some leaks, it is possible to obtain some information from the WME. Interesting information is the perception of burst dates, and also the dates when valves were open or closed. It is also possible to know what infrastructures had an intervention. Some sensors were signalized as close to the leak depending on the infrastructures intervened. In red are the sensors considered as such in Figure 4.2, Figure 4.6, and Figure 4.4.

4.2 Anomaly detection

The main goal of this master thesis is to detect anomalies in time series with the main focus on bursts. The proposed solution is divided in the following steps:

Figure 4.1: Water flow direction on simulated data

1. Process data using the sliding window algorithm. Input can be described as a time series $X = (x_t[1], x_t[2], \ldots, x_t[m])$, where each point $x_t[i]$ is a m-dimensional vector of readings from sensors for m variables (e.g. flow, pressure) at time instance t. The time series is split into equal sized batches of length w denoted as $x_t[i] = \{x[j], x[j+1], \ldots, x[j+w-1]\}$. Here i is the batch number and $j = w(i-1)+1$ of first point in the batch. This value w needs to be optimized.

2. Implement encoder-decoder architectures based on LSTM: LSTM autoenconders [50], CNN-LSTM [55], CNN-Bi-LSTM [55], SCB-LSTM [17].

   A LSTM autoencoder uses a LSTM as decoder and other LSTM network as decoder. As mentioned in sections above, an LSTM-based encoder is used to map an input sequence to a vector representa-

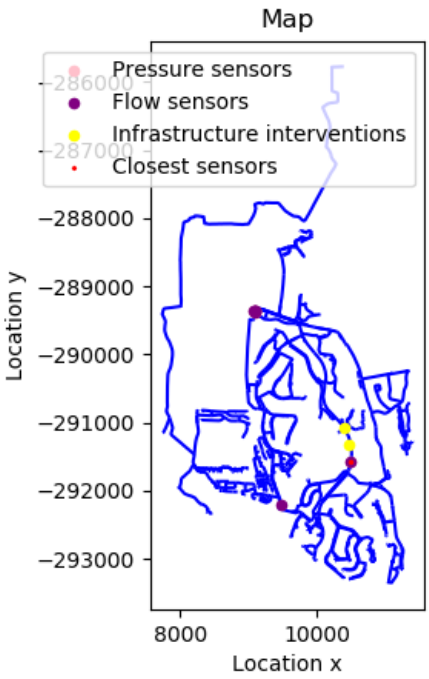

Figure 4.2: Map of WDS with marked sensors and possible location of the leak (in yellow)

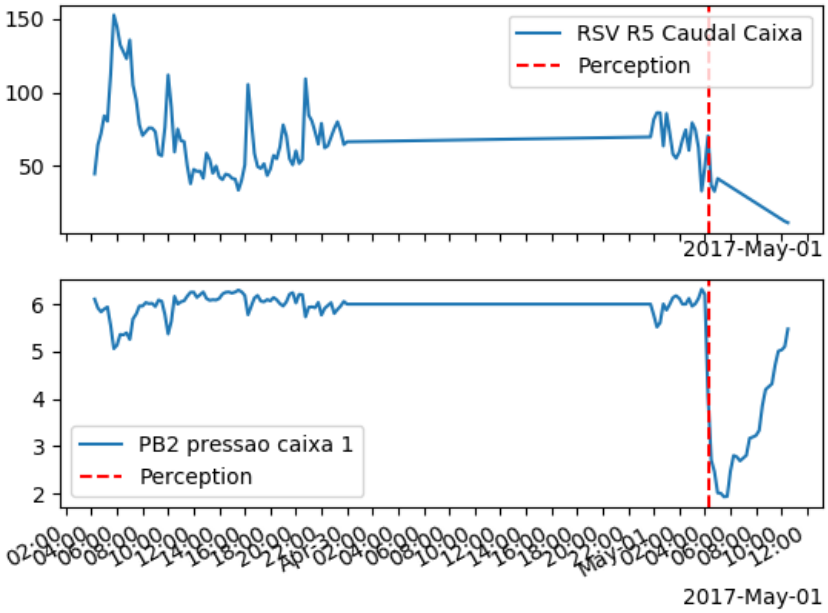

Figure 4.3: Time series of flow and pressure sensors considered as close in Figure 4.2. In red, time of perception reported by the WME.

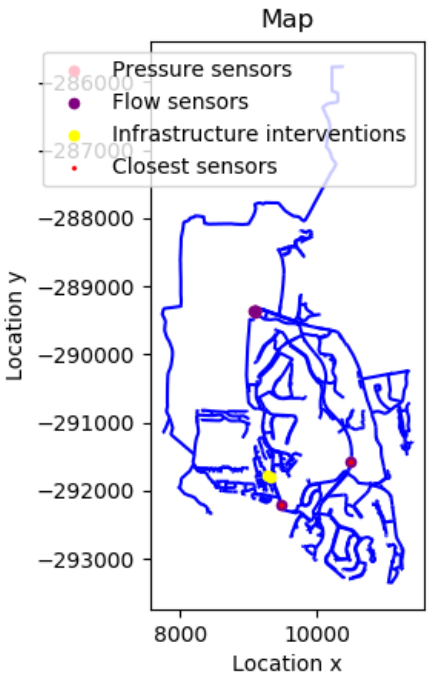

Figure 4.4: Map of WDS with marked sensors and possible location of the leak.

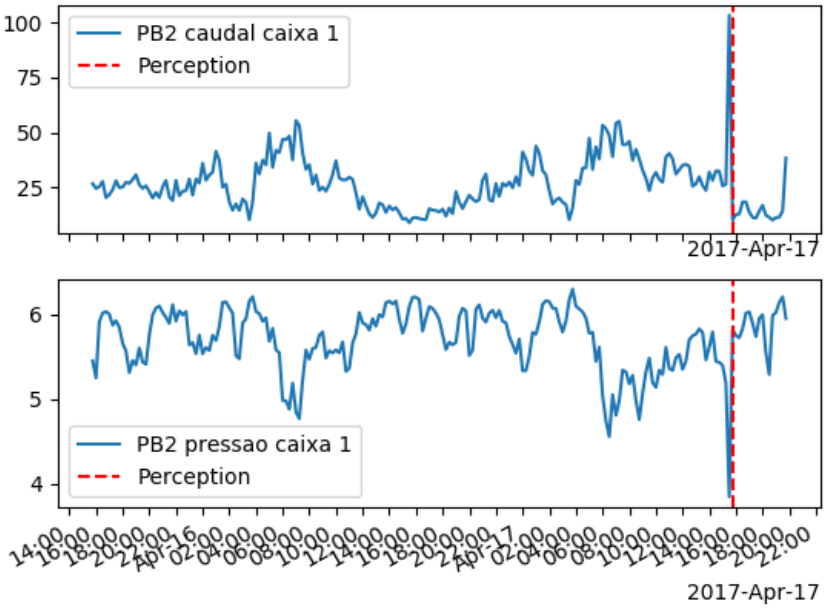

Figure 4.5: Time series of flow and pressure sensors considered as close in Figure 4.4. In red, time of perception reported by the WME.

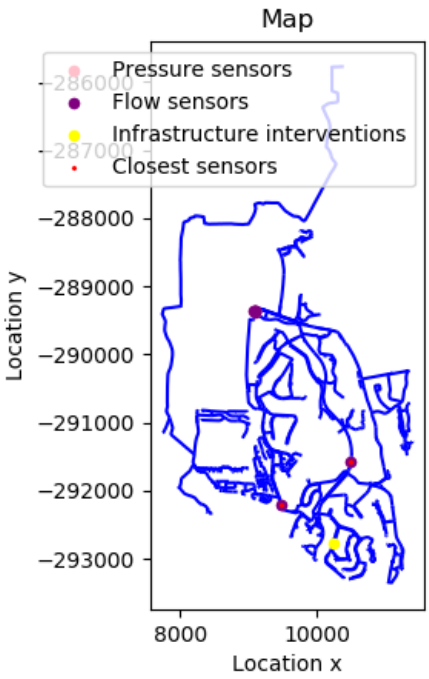

Figure 4.6: Map of WDS with marked sensors and possible location of the leak.

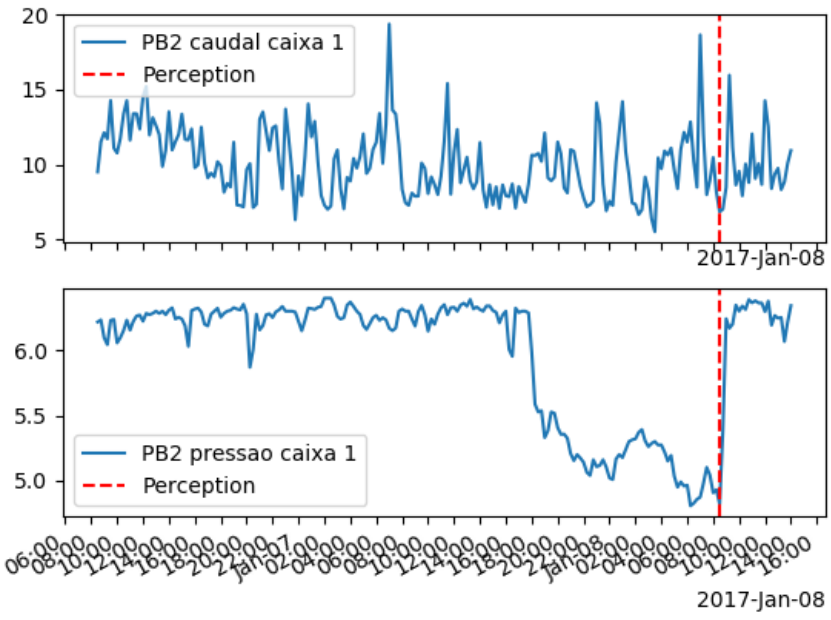

Figure 4.7: Time series of flow and pressure sensors considered as close in Figure 4.6. In red, time of perception reported by the WME.

tion of fixed dimensionality. In similarity with [27], to force an autoencoder to learn useful features, the autoencoder is regularized by using a sparsity constraint such that only a fraction of the nodes would have nonzero values, called active nodes. To achieve this, a penalty term is applied to the loss function such that only a fraction of the nodes become active.

Another approach is to replace the LSTM-based encoder by a deep convolution neural network (CNN). Some of the architectures that rely on convolution operators include: SCB-LSTM, CNN-LSTM, and CNN-Bi-LSTM. CNN performs feature extraction and the result is used to feature learning with these stacked bidirectional LSTMs. After this procedure, stacked layers of Bi-LSTM and/or LSTM enhance feature learning, and finally, the regression layer predicts. SCB-LSTM corresponds to a stacked CNN, then a stacked Bi-LSTM, and then a stacked LSTM.

The main idea of the proposed fault detection consists of error reconstruction. If the error is above a certain threshold, an outlier is detected. We first train the LSTM Encoder-Decoder model to reconstruct the normal time-series. The reconstruction errors are then used to obtain the likelihood of a point in a test time-series being anomalous. In this context, for each point $x_i[t]$, an anomaly score $a_{(i)}$ of the point being anomalous is obtained. A higher anomaly score indicates a higher likelihood of the point being anomalous [50].

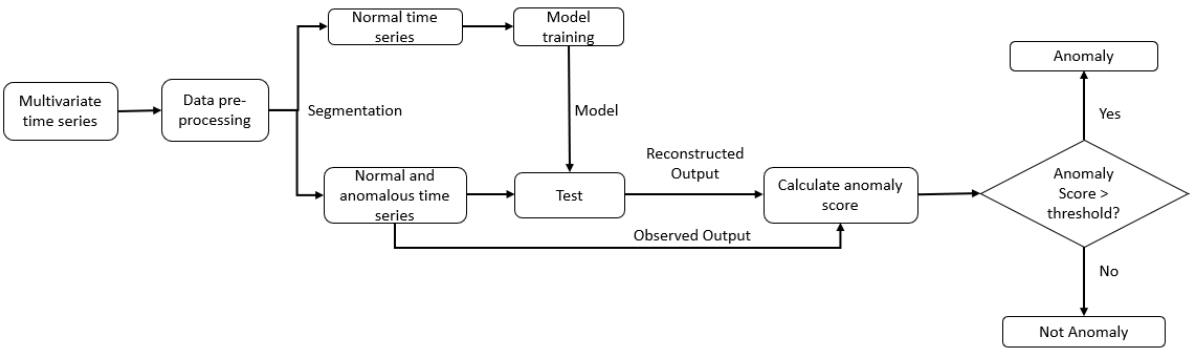

Figure 4.8: Pipeline for anomaly detection

3. Compare the models mentioned above with baseline models such as: stacked LSTM [19] and stacked Bi-LSTM. These architectures are also reconstruction models.

4.3 Data preprocessing

The data is preprocessed, such that does not have missing data or censored data (including left-censored data such as values below detection limits). Raw time series are unevenly spaced. For the research, interpolation was used, so measurements have exact intervals of 1 minute. Other methods are being researched by other parallel research. For the missing values, interpolation was also used. Before training, data is compressed by the MinMaxScaler normalization method, being compressed between the range [-1, 1]. Data was split as train data, normal validation data, v_N, and abnormal validation data, v_A. v_N was used for early stopping. v_A was used to find thresholds. v_A is a sequence of data that has subsequences of normal and abnormal data. v_N has only data considered normal. Normal data is considered the one without leaks or no reported burst events at the WME database, in the case of real data. In real data, experiments were done so seasonality was removed. The window size is 90 minutes. The number of sequences is 7. In some experiments in real data, seasonality was removed. To remove seasonality from the data, the seasonal component is subtracted from the original series and then differentiates it to make it stationary. In some experiments in synthetic data, flow consumption was subtracted according to the flow direction. To find the ideal percentage of splits, between validation and training, the walk-forward approach was used, using an expanding window. This method is also referred to as a rolling window analysis. This approach was used instead of traditional cross-validation because it preserves temporal order of data [61], being a more realistic view of how effective the models would truly have been in the past, and helps to avoid overfitting [61].

4.4 Architectures

In this section architectures of the deep neural networks are described. The proposed encoder-decoder architecture has an encoder part, typically a recurrent and/or convolution layers, followed by a decoder, typically recurrent layers. Other architectures, such as stacked, have LSTM or BiLSTM layers in sequence. Regarding the convolutional part, two types of layers are considered: multi-channel convolutional and multi-head convolutional. Multi-head was only considered for multi-variate time series, on the real dataset. On the other hand, two types of recurrent layers are examined: LSTM and Bi-LSTM.

A typical neural network, has a change in the distribution of network activations due to change in network parameters during training - internal covariate shift [62]. To minimize that a layer of Batch Normalization (BN) is applied. The batch normalization (BN) algorithm tries to normalize the inputs to each hidden layer so that their distribution is reasonably constant during training. This has advantages speeding up convergence, having much higher learning rates, and be less careful about parameter initialization. For each convolution, a Batch Normalization is applied followed by an Activation Layer (Relu). Batch normalization was used. This method worked better than pooling.

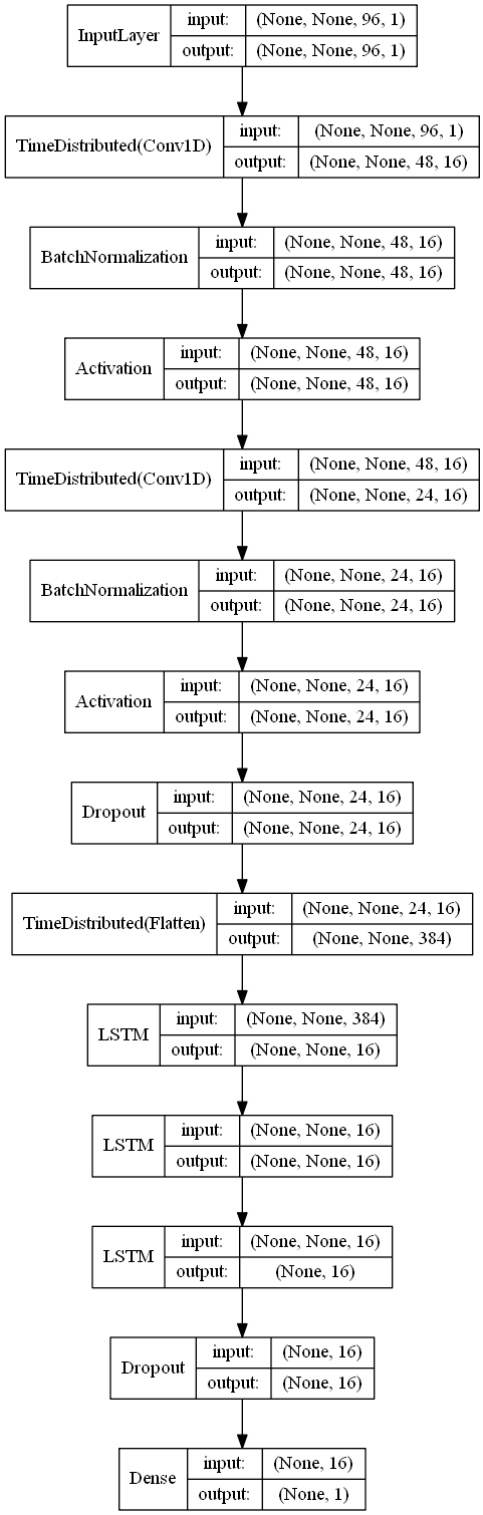

Figure 4.9: CNN-LSTM model diagram

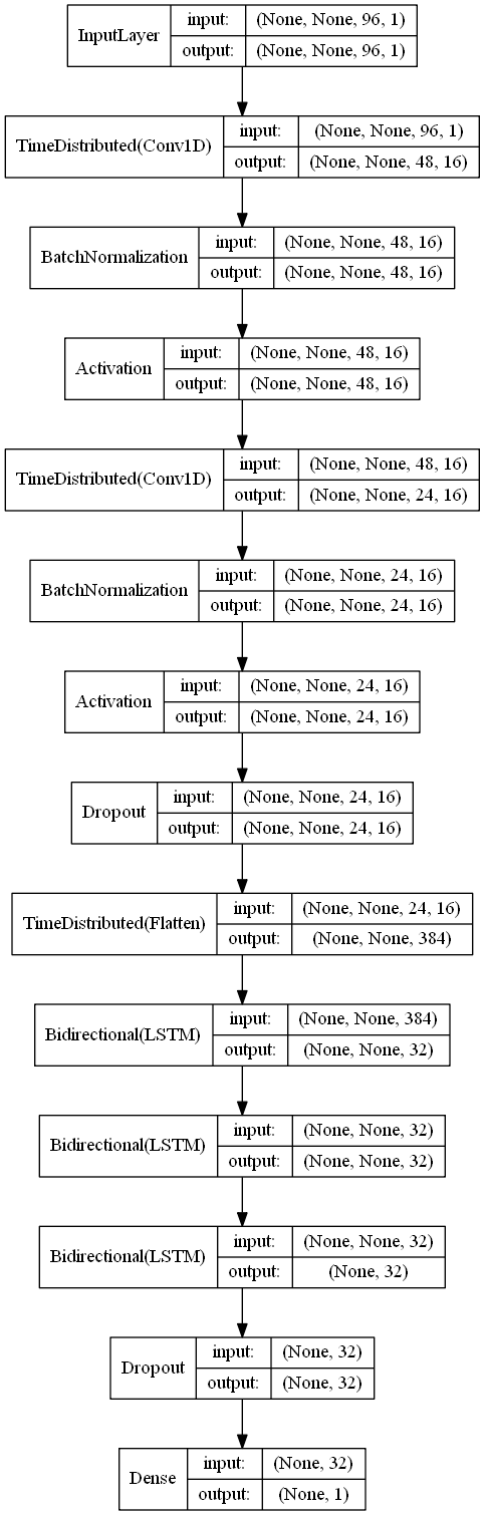

Figure 4.10: CNN-BiLSTM model diagram

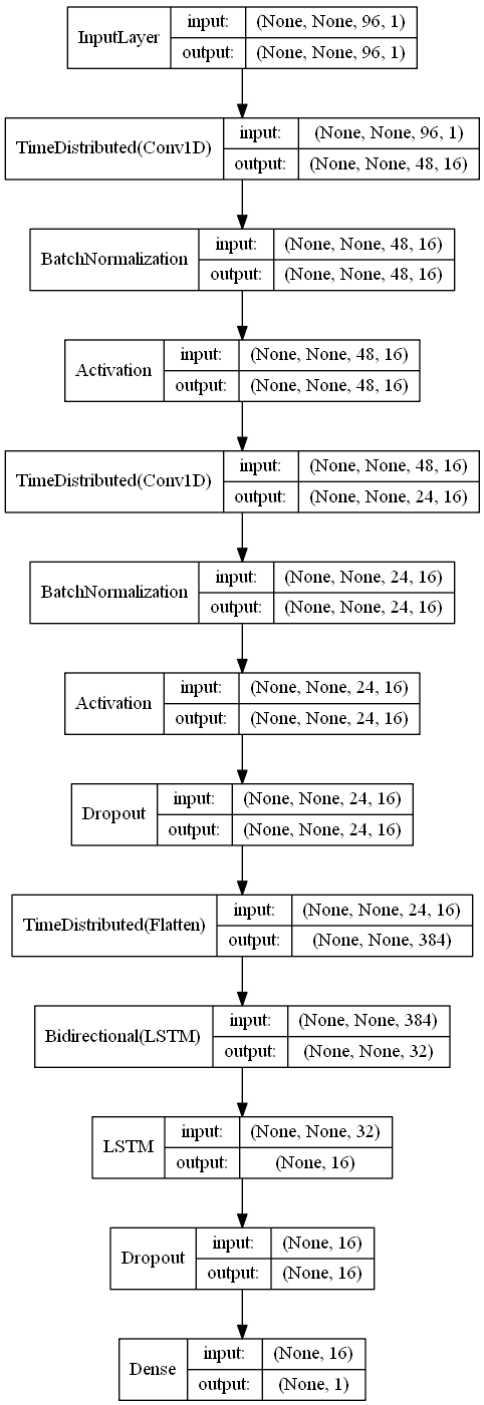

Figure 4.11: SCB-LSTM model diagram

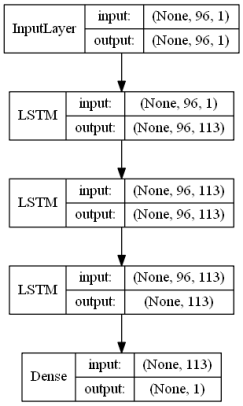

Figure 4.12: Stacked LSTM model diagram

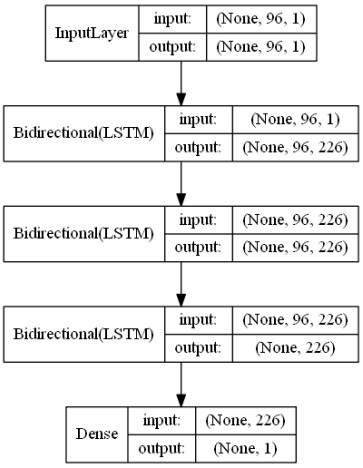

Figure 4.13: Stacked BiLSTM model diagram

A simple autoencoder LSTM was also implemented, where the encoder was an LSTM, and the decoder also an LSTM. It was tested multiple cases, such as an overcomplete, an undercomplete, and sparse. Preliminary results showed a bad performance in all cases.

4.5 Hyperparameter Tuning

Bayesian Optimization using Gaussian Processes was used to find the best set of parameters. The python library used was scikit-optimize. The objective was to minimize validation loss and time expend to train. Blocked Rolling out cross-validation was applied [63]. The optimal parameters found for the encoder-decoder with an initial CNN part are indicated in Table 4.1.

Metrics	Size
Stride size	2
Batch size	64
Encoder layers	2
Decoder layers	3
Number Kernels	5
Number filters	16
Dropout rate	0.25

Table 4.1: Parameters for CNN

The optimal parameters found for the stacked architectures are indicated in Table 4.2.

Metrics	Size
Batch size	128
Stacked layers	3
Learning rate	0.01

Table 4.2: Parameters for Stacked Architectures

4.6 Temperature correction model

Temperature changes influence water demand, mostly in summer [55,64]. In summer, the water consumption changes significantly when the temperature changes 1°C [55]. The predicted value by the models is compared with the average water used for the past five days. If the difference between the predicted value and the average of the past five days flow readings is within the set deviation range, the predicted value is accepted, otherwise, the predicted value is corrected. To be corrected, the prediction value is multiplied by the correction coefficient. The correction coefficient is calculated using the least-squares fit between the forecast errors.

4.7 Evaluation Methodology

In this work, the problem is approached as an unsupervised task. Even though models are trained under the assumption that the train sequence is normal, it does not use abnormal behavior as an attempt to learn a discriminative model. Multiple metrics and methods are discussed to evaluate this work in the following subsections.

Anomaly score is given by the reconstruction error for each point t_i:

$$\mathbf{a}^{(i)} = \left| \mathbf{x}^{(i)} - \mathbf{x}'^{(i)} \right| \tag{4.1}$$

where x is the observed time series and $x\prime(i)$ is the reconstructed time series.

If above a certain threshold, the point in a sequence is considered an anomaly.

To calculate that threshold in the validation set, it maximizes F-beta score:

$$F_\beta = \left(1 + \beta^2\right) \times P \times R / \left(\beta^2 P + R\right) \tag{4.2}$$

Initially, it was thought to use Maximum Likelihood Estimation for the anomaly score [19]. However, that assumes that reconstruction is a normal distribution and that is not true, as can be seen in Figures 4.14, 4.15. Even after a box-plot transformation p-value would be less than 0.05.

Reconstruction loss: The decoder's output is evaluated by comparing the reconstructed image with the original one, using a loss function, for example, Mean Square Error (MSE). Then, it is propagated backward and update all the weights from the decoder to the encoder.

$$\text{MSE} = \frac{1}{n} \sum_{t=1}^{n} (x_t - \hat{x}_t)^2 \tag{4.3}$$

with x_t being the observed value of the variable being predicted and $\hat{x}_t$ being the predicted value.

Precision is the number of outliers correctly identified (TP) divided by the sum of the number of outliers correctly identified (TP) and the number of normal points identified as an outlier (FP).

$$\text{Precision } = \frac{TP}{TP + FP} \tag{4.4}$$

Recall is the number of outliers correctly identified (TP) divided by the sum of TP and the number of outliers that are classified as normal (FN).

$$\text{Recall } = \frac{TP}{TP + FN} \tag{4.5}$$

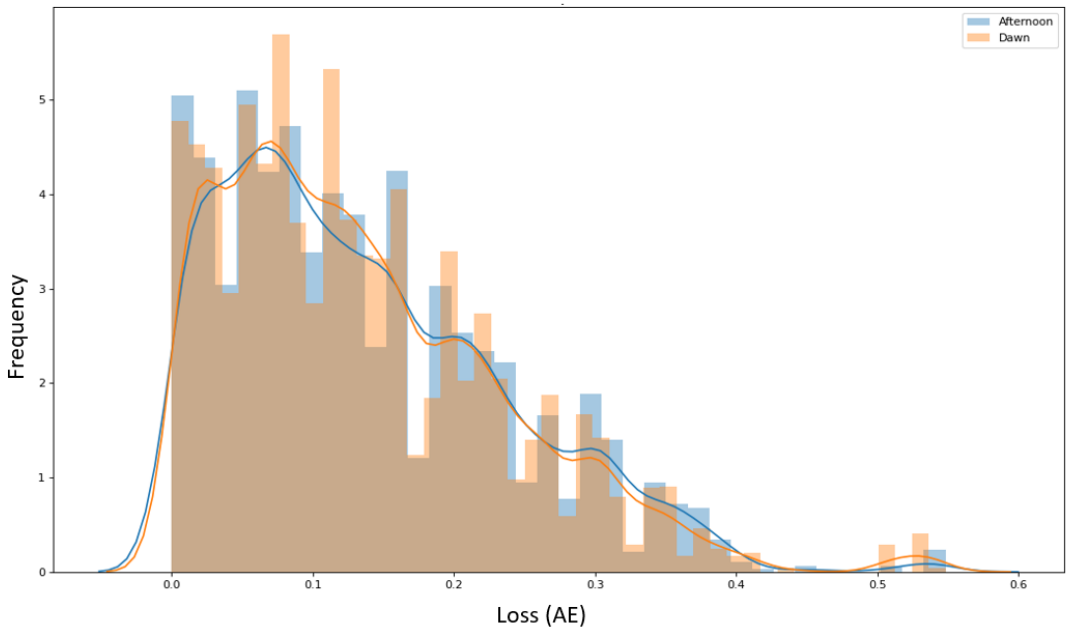

Figure 4.14: Loss absolute error by times of the day

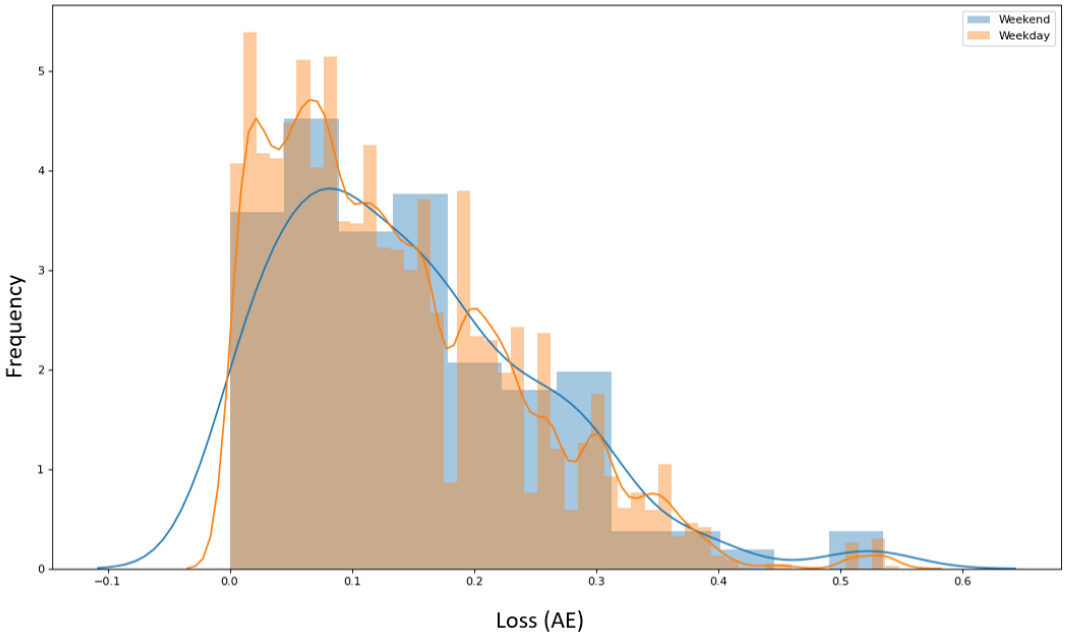

Figure 4.15: Loss absolute error by business days and weekends

F1-score a combination of recall and precision, in order to consider both metrics.

$$F1 \text{ -score } = 2 \cdot \frac{\text{Precision } \cdot \text{Recall}}{\text{Precision } + \text{Recall}} \tag{4.6}$$

Cross validation: The synthetic test set is used to tune parameters and is used as a baseline. Validation data is a combination of normal and anomalous data. The hyper-parameters are tuned (e.g., dropout rates and the number of convolution kernels) until the validation loss on normal data had converged. The error difference between train and validation sets can be used to evaluate the capacity of the models to generalize thus avoiding overfit.

5

Results

Contents

This chapter first presents a data exploratory analysis, where some aspects of the data are drawn. Then, the proposed methods are evaluated on simulated data, which is a much more controlled scenario giving a sense of what to expect on real data. After that, results are obtained from real data and some factors are presented that may have influenced them.

5.1 Data Exploratory Analysis

The analyzed data was collected by the SCADA system on Infraquinta in the year of 2017. In general, flow sensors and pressure sensors measure and record the flow and pressure every minute. Time series decomposition and descriptive statistical analysis are made. The decomposition of time series is a statistical method that splits a time series into several components, each representing one of the underlying processes. Time series decomposition was made using seasonal and trend decomposition using Loess package. Then a descriptive statistical analysis is made.

During the year, there is higher water demand in summer, having the highest peak in August, and lower demand in winter, having the lowest demand in December. Two weeks were analyzed, from Monday to Sunday, corresponding to winter, January 16th to January 22nd, and corresponding to summer, August 7th to August 13th. The sensors analyzed were sensor RPR Caudal Pre, which corresponds to a flow sensor, and Sensor RPR Pressao Pre, which corresponds to a pressure sensor. During the winter week, the flow trend decreases from Monday to Tuesday, and then increases until the weekend where it decreases, as seen in Fig. 5.2. The highest picks are on January 16th (Monday) and January 20th (Friday). The pressure trend has different spikes along the day. During the summer week, the highest demand occurs between Friday and Saturday, as seen in 5.1. The flow trend increases until the weekend when it starts to decrease. The pressure trend decreases throughout the week.

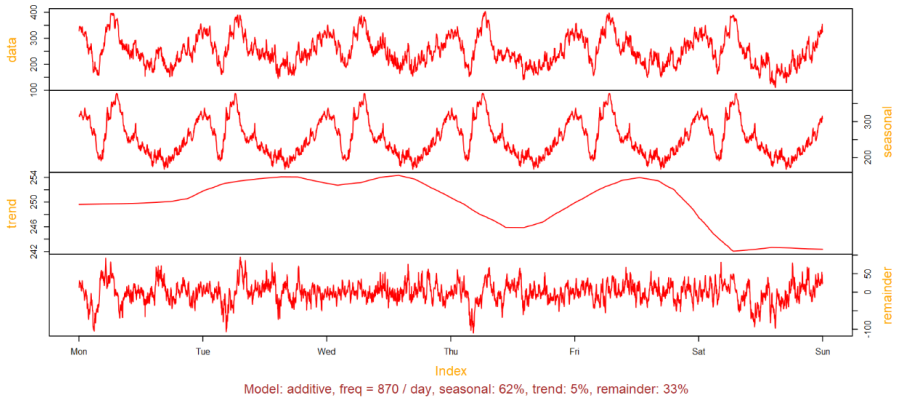

Figure 5.1: Time series decomposition of a flow sensor during the summer week August 7th to August 13th.

Weekdays of winter and summer were also analyzed. During a winter day of the week, the flow

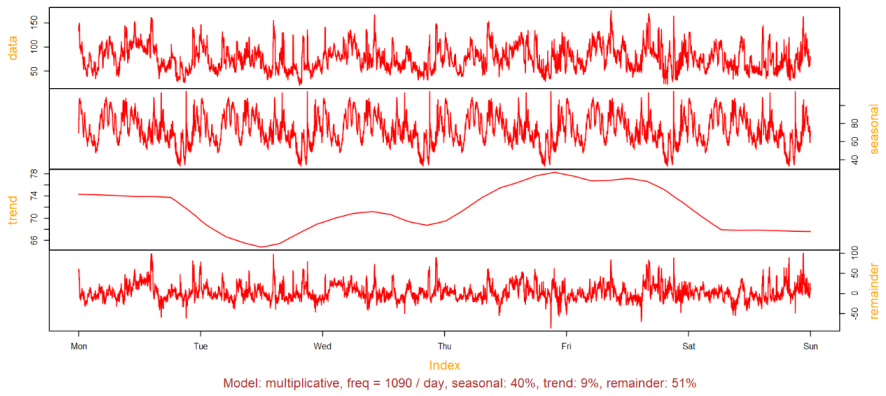

Figure 5.2: Time series decomposition of a flow sensor during the winter week January 16th to January 22nd.

trend increases from 3:00 to 12:00 and decreases until 18:00 where it starts to increase again. During the weekend, the flow trend increases from 3:00 to 8:00, then decreases until 18:00 and starts to increase again. During the summer weekday, the flow trend increases until 8:00, decreases after that, and starts to increases slowly again at 15:00. The weekend has a similar behavior. It is to remark that there is higher consumption at night because it has irrigation systems turned on at that time. The concept of minimum flow does not apply here. In conclusion, the decomposition analysis strongly suggests that the time series is seasonal.

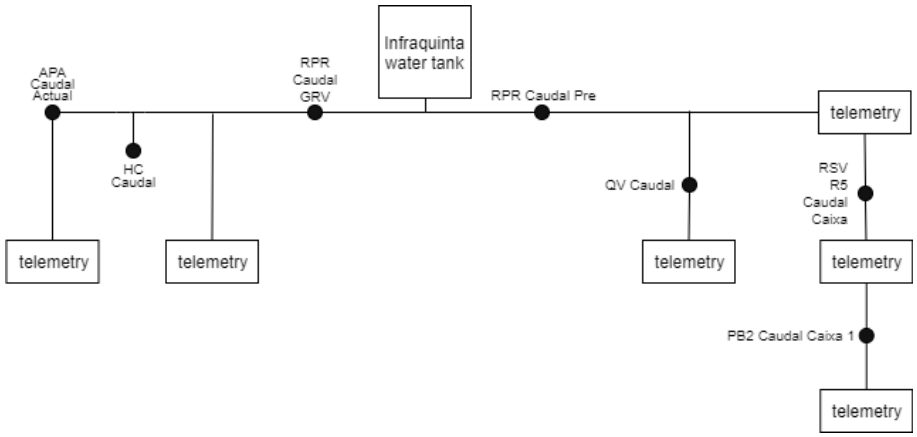

Figure 5.3: Map of the network with flow sensors

The correlation between flow sensors were analyzed to understand how the flow is distributed along the network. The highest correlations between sensors were between sensor RSV R5 Caudal Caixa and sensor RPR Caudal Pre (0.8367), sensor QV Caudal, and sensor APA Caudal Actual (0.8498), sensor APA Caudal Actual and sensor RPR Caudal Grv (0.941).

The correlation between pressure sensors and flow sensors were also analyzed. The sensor flow RPR Caudal Pre and RPR Caudal Grv have a correlation of 0.75. The pressure sensor RPR Pressao Pre and

flow sensor RPR Caudal Pre have a correlation of 0.19. The flow sensor RPR Caudal Grv and pressure sensor have a correlation of 0.003159. Even though there is a slight correlation between each variable, it is hard to infer values from a variable using the other variable. In table 5.1 a statistical analysis is made for flow (sensor RPR Caudal Pre and RPR Caudal Grv) and pressure sensors (sensor RPR Pressao Pre and RPR Pressao Grv). The pair of sensors RPR Caudal Pre and RPR Pressao Pre, and RPR Caudal Grv and RPR Pressao Grv measure the same zone.

Metrics	RPR Caudal Pre	RPR Caudal Grv	RPR Pressao Pre	RPR Pressao Grv
Mean	128.32	56.44	2.23	0.40
Std. Deviation	73.27	42.52	0.10	0.001093
25% percentile	69.0	23.60	2.10	0.40
50% percentile	114.0	45.30	2.30	0.40
75% percentile	173.0	78.20	2.30	0.40

Table 5.1: Statistical analysis of flow and pressure sensors.

Leaks were also analyzed. The burst occurred on December 12th is given as an example. The sensors RPR Caudal Grv and RPR Caudal Pre noticed the leak a lot, having the flow increased around noon, as observed in Figure 5.4. As a matter of pressure, there was no obvious pressure drop as theoretically expected, as seen in Figure 5.5. In other bursts analyzed, an obvious drop was not seen either. They have a noticeable jump when a valve is open or closed. This may be a problem for differentiating between valves maneuvers and bursts.

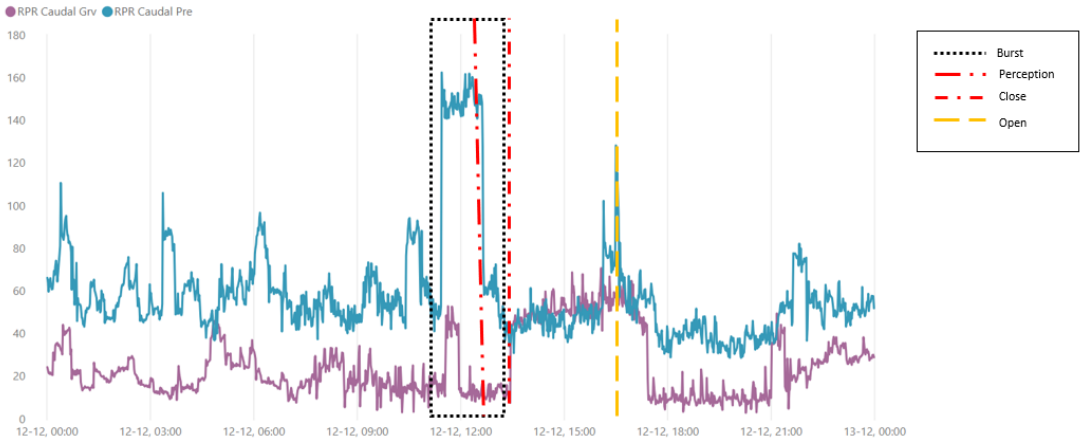

Figure 5.4: Time series of flow sensors RPR Caudal Pre and RPR Caudal Grv during the day December 12th when a burst occured.

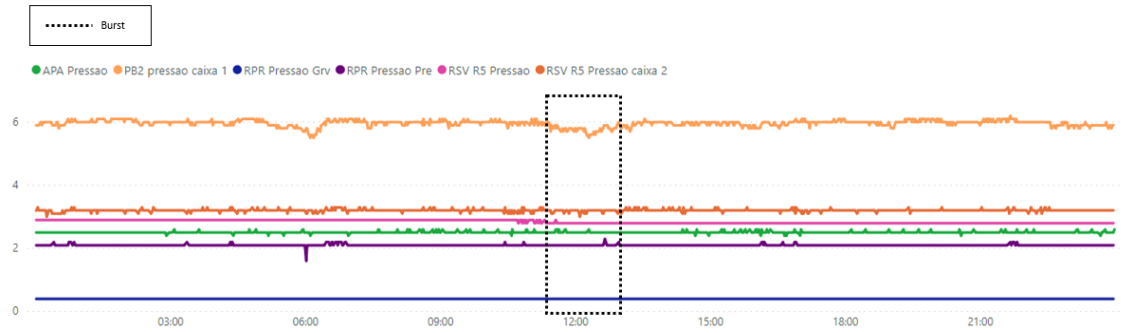

Figure 5.5: Time series of pressure sensors during the day of December 12th when a burst occured.

5.2 Experiments

All the experiments were done on a processor Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz. Python libraries used were Keras 2.4.3 and Tensorflow 2.3.0, on operating system Ubuntu 20.04 LTS.

5.3 Experiments in Simulation Data

Simulated data is tested to get preliminary results and get a sense of what to expect in real data. This data has a more controlled scenario, being possible to know when and where a certain leak occurred. Leaks were simulated in different locations in the WDS. Thus, only one real pressure sensor was simulated, being hard to detect negative pressure waves so it was not considered for the study. Only flow sensors were considered.

For the first experiment, all events in all locations were considered. The chosen sensor to be studied was sensor 2 (PB2 caudal caixa 1) because it has more events nearby. For the expanding window method, the percentage of validation sets changes. For this method, 5 partitions were considered. Percentages for validation data were 20%, 16%, 14% , 12% and 10%. v_A was used to find thresholds. The train size used was 7 months. Validation set,v_A, has 17% of leaks.

It can be seen from table 5.2 that the models with less error are stacked LSTM and stacked BiLSTM. These architectures also are the ones that take longer to train, taking 356s for the stacked BiLSTM to converge and 197s for the stacked LSTM. CNN-LSTM is the one that converges faster.

Bootstrap method was then applied to get confidence intervals as shown in Tables 5.3, 5.5 and 5.4. The block size for bootstrapping is 20 days. Each block has 15% of leaks. There is a 95% of likelihood that the range mentioned at the tables covers the true statistic of the metric.

As can be seen in table 5.3 the model with the highest mean precision is SCB-LSTM. The one with the maximum confidence interval is CNN-LSTM. From table 5.4, recall is low for all architectures. This is expected since the data points labeled as anomalous are not high [19]. Accuracy is high, due to high

Results	Validation error	Training error	Training time (s)	Epochs
CNN-LSTM	2.576×10^{-2}	4.304×10^{-2}	**33**	**9**
CNN-BiLSTM	2.020×10^{-2}	3.253×10^{-2}	68	12
SCB-LSTM	1.747×10^{-2}	3.706×10^{-2}	44	10
Stacked BiLstm	4.721×10^{-3}	7.955×10^{-3}	386	13
Stacked LSTM	$\mathbf{2.205} \times 10^{-3}$	$\mathbf{4.910} \times 10^{-3}$	197	14

Table 5.2: Analysis of convergence for all architectures using synthetic data

Results	Range	Mean ± Standard Error
CNN-LSTM	[0.200, **0.680**]	0.35 ±0.17
CNN-BiLSTM	[0.233, 0.5257]	0.337 ±0.111
SCB-LSTM	[**0.274**, 0.481]	**0.37362** $\pm 7.344 \times 10^{-2}$
Stacked BiLSTM	[0.0835, 0.4275]	0.2439 ±0.127
Stacked LSTM	[0, 0.25]	0.15 ±0.122

Table 5.3: Confidence interval for precision

Results	Range	Mean ± Standard Error
CNN-LSTM	$[9.800 \times 10^{-3}, 3.149 \times 10^{-2}]$	$7.200 \times 10^{-3} \pm 1.900 \times 10^{-2}$
CNN-BiLSTM	$[\ 2.01 \times 10^{-2}, 3.391 \times 10^{-2}]$	$2.471 \times 10^{-2} \pm 5.453 \times 10^{-3}$
SCB-LSTM	$[4.05 \times 10^{-2}, 6.683 \times 10^{-2}]$	$5.302 \times 10^{-2} \pm 1.038 \times 10^{-2}$
Stacked BiLSTM	$[\ 2.221 \times 10^{-3}, 4.953 \times 10^{-2}]$	$2.132 \times 10^{-2} \pm 1.813 \times 10^{-2}$
Stacked LSTM	$[0, 5.300 \times 10^{-3}]$	$1.905 \times 10^{-3} \pm 2.086 \times 10^{-3}$

Table 5.4: Confidence interval for recall

Results	Range	Mean ± Standard Error
CNN-LSTM	[0.810,0.830]	0.00440 ±0.8265
CNN-BiLSTM	[0.818,0.826]	$0.8265 \pm 3.168 \times 10^{-3}$
SCB-LSTM	[0.821, 0.828]	$0.824 \pm 2.640 \times 10^{-3}$
Stacked BiLSTM	[0.820, 0.825]	$0.822 \pm 2.160 \times 10^{-3}$
Stacked LSTM	[0.8226, 0.82574]	$0.8244 \pm 1.187 \times 10^{-3}$

Table 5.5: Confidence interval for accuracy

specificity at the cost of a low recall (Table 5.5). To remark that, even though stacked architectures have the lowest loss they have the worse results either in precision or recall metric.

For the second experiment, only events close to sensor 2 were considered. Each validation sequence has approximately 10%-17% of leaks. The results were mediocre, the left confidence limit is 0. This is due that the sensor is not only detecting events close to it but leaks simulated in other parts of the network. This indicates that it may be difficult to locate the leaks through the sensors.

Figures 5.6, 5.7, 5.8, 5.9 and 5.10 depict the reconstruction of a normal validation set and the test set (normal and abnormal sequence).

Note that red and blue lines are respectively test and test target data as ground truth.

Validation error is smaller than training error, having a considerable generalization gap on Figure 5.11, Figure 5.11 and Figure 5.15. This is an issue on Keras, because usually dropout is activated when training but deactivated when evaluating on the validation set. Stacked architectures clearly converge 5.13 5.14.

Overall, the results are not good. The ROC curve is very close to the diagonal, being very naive. This can have several causes. There are leaks simulated near consumption points, i.e. far apart from the selected sensors. The size of leaks also have an influence, bigger leaks are easily detected. Comparing the first and second experiments, precision is worst when events are filtered. This is because events not considered close to the sensor are detected by the sensor. The sensor is detecting events not close to him. For the first experiment, the CNN-BiLstm has the best AUC. The stacked biLSTM has the best AUC for the second experiment. Since stacked biLSTM has the smallest loss, it is pickier to detect an anomaly. Those anomalies are generally closest to him. In the first experiment, there were events near sensor 6 being detected (RSV R5). Which makes sense due to the direction of flow shown in Figure 4.1. Small leaks are easily confused with normal behavior as can be seen in Figure 5.7. For the second experiment, is the stacked-BiLstm. For the first experiment, CNN-BiLSTM can detect losses as small as 0.08 m^3/h to medium losses 1.91 m^3/h. Stacked BiLSTM detects from 0.011 m^3/h to 1.49 m^3/h . There was not possible to differentiate between unusual consumption and leaks. For the third experiment, time series of sensors 2 were preprocessed. The water flow detected in other sensors was subtracted from sensor 2, according to the flow direction. The results of these experiments were mediocre.

5.4 Experiments in real data

Real data is a much more dynamic scenario, where is not possible to know exactly where or when a leak occurred or even if an event of another type occurs. Only burst events reported in the database by workers in the WME database were considered. These reported events are only leak related or some other maintenance operation that required open and close valves. Only flow sensors were considered. . To this analysis, three partitions were considered. Percentages for validation data were 16%, 14% and

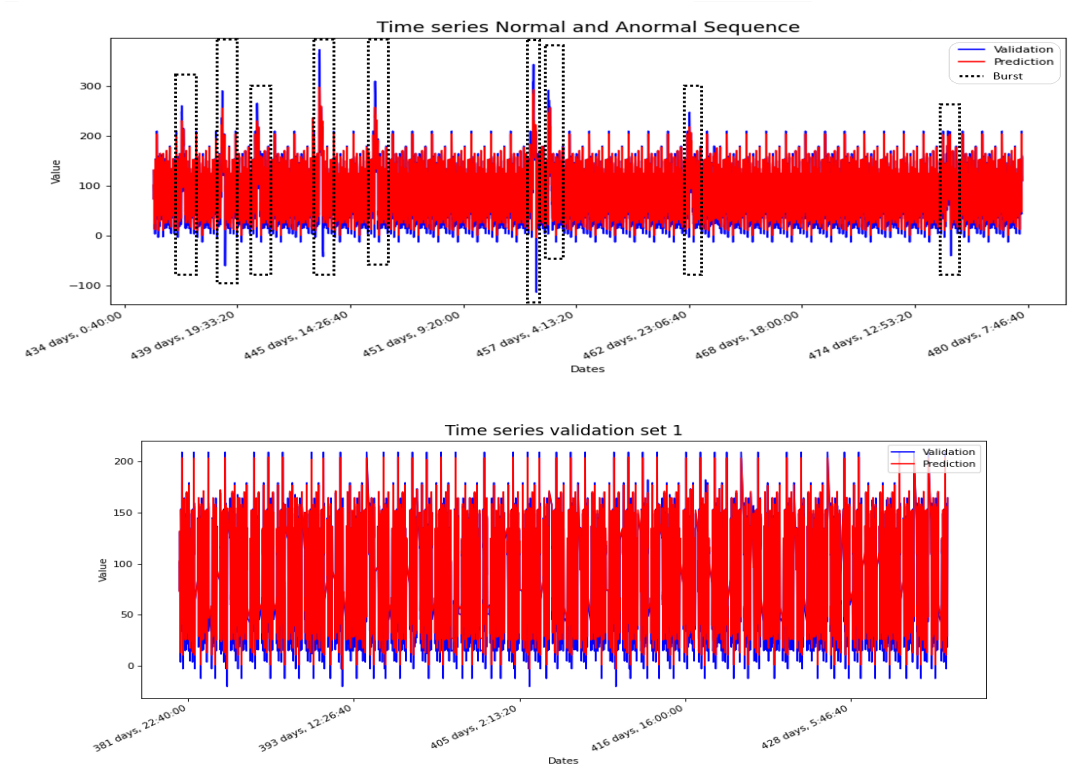

Figure 5.6: Comparisons between the original and reconstructed sensor readings on validation sets predicted by a CNN-LSTM trained with simulated data.

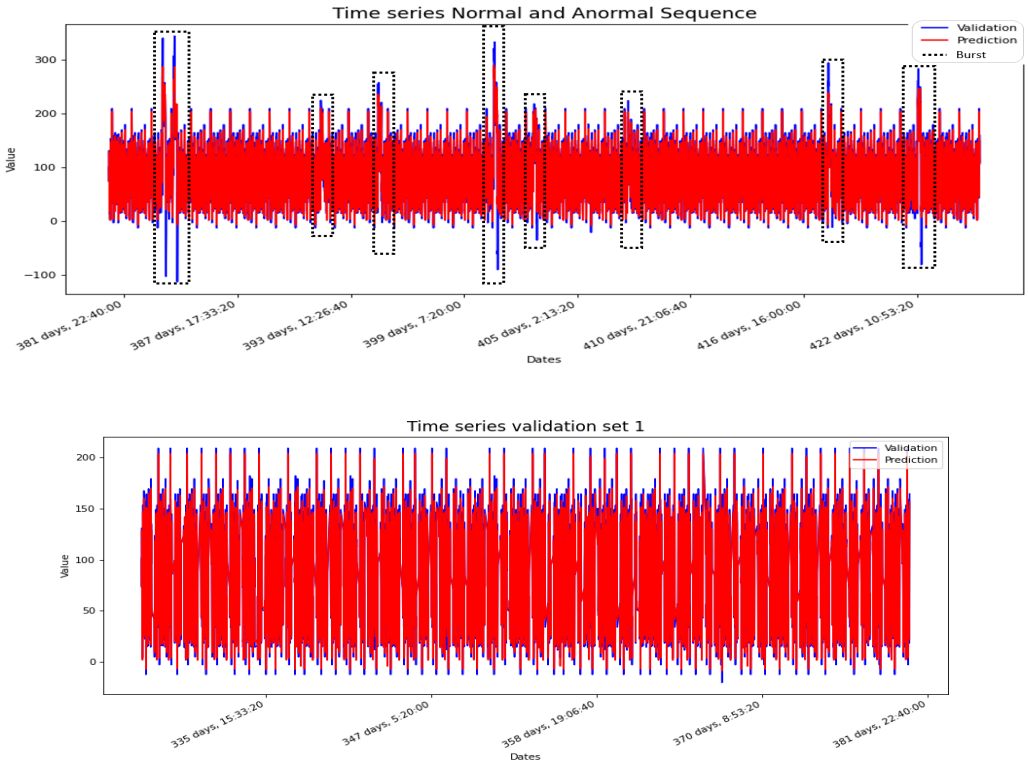

Figure 5.7: Comparisons between the original and reconstructed sensor readings on validation sets predicted by a CNN-BiLSTM trained with simulated data

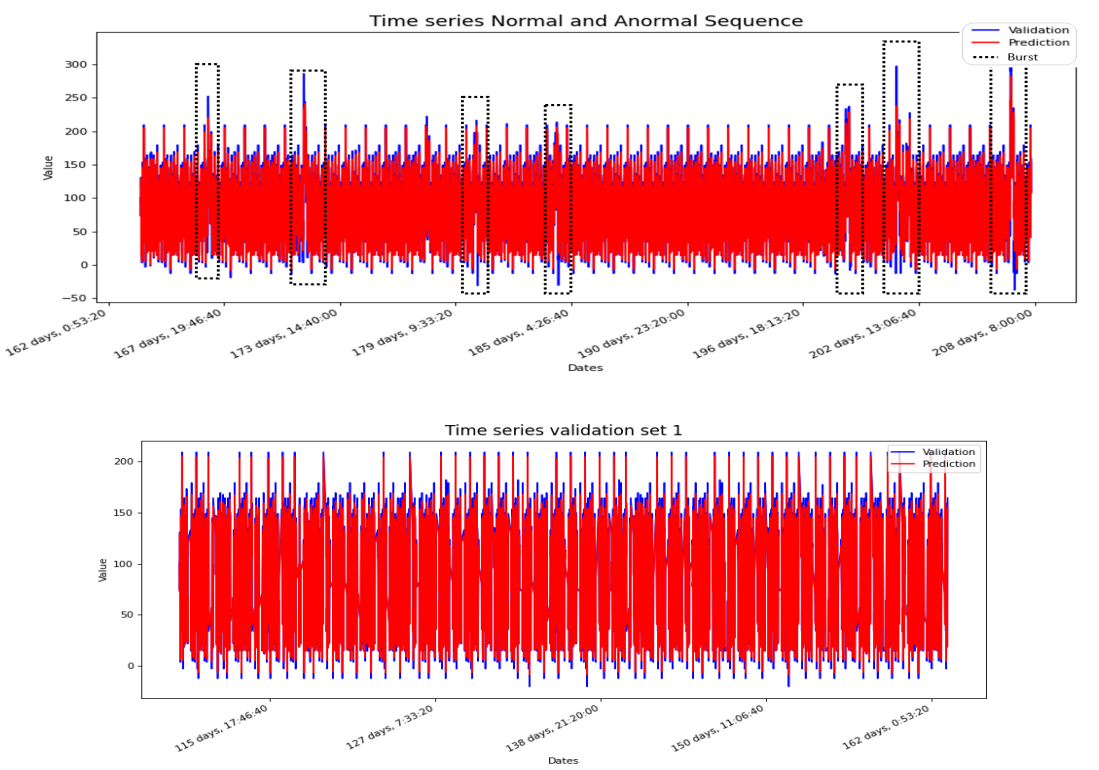

Figure 5.8: Comparisons between the original and reconstructed sensor readings on validation sets predicted by a SCB-LSTM trained with simulated data

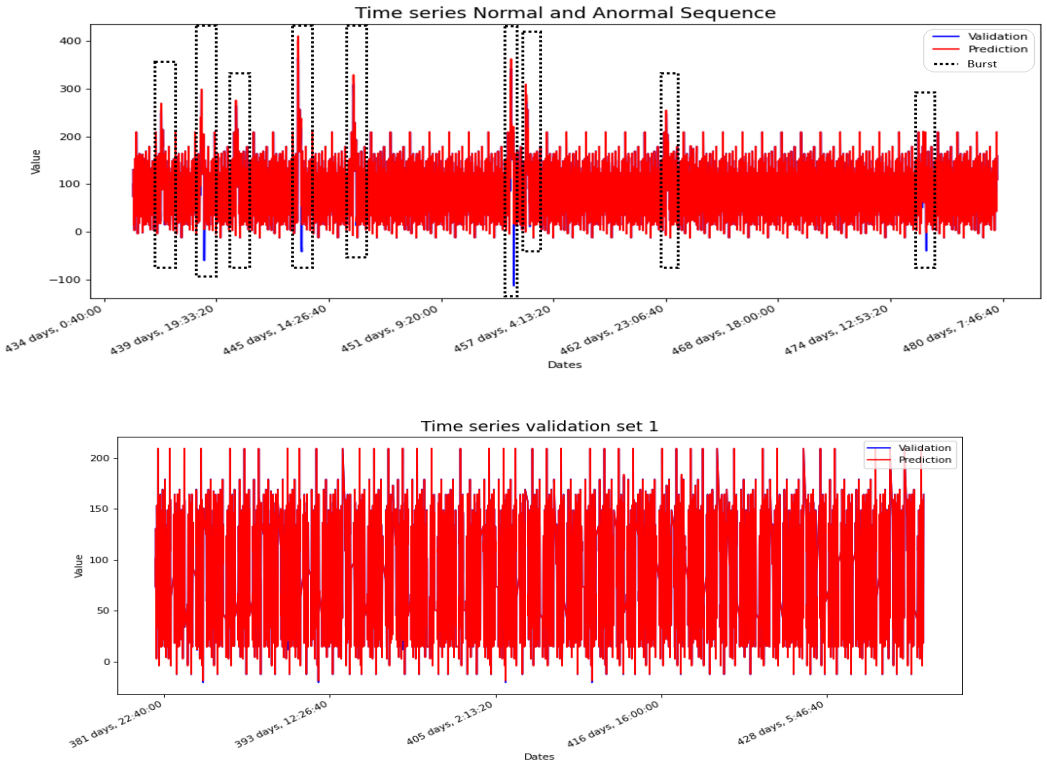

Figure 5.9: Comparisons between the original and reconstructed sensor readings on validation sets predicted by a stacked LSTM trained with simulated data

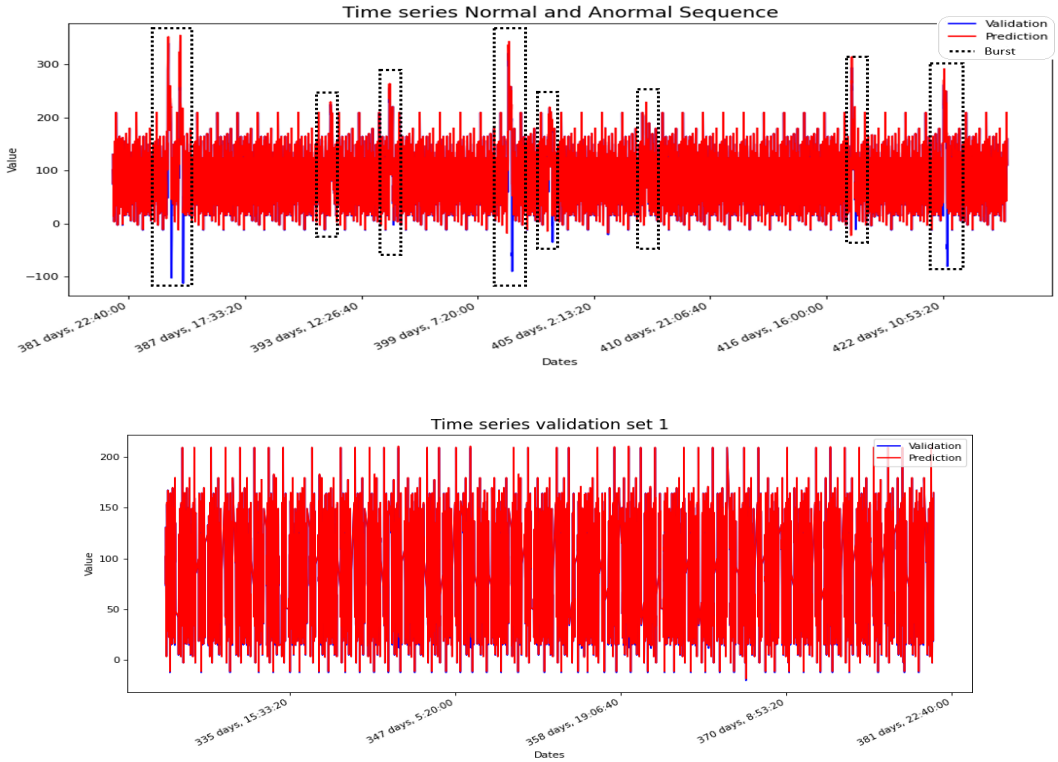

Figure 5.10: Comparisons between the original and reconstructed sensor readings on validation sets by a stacked BiLSTM trained with simulated data

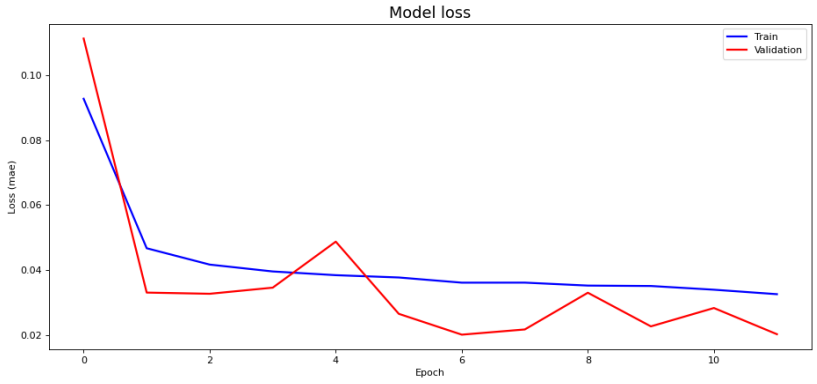

Figure 5.11: Train and validation learning curves for CNN-BiLSTM on synthetic data

Results	Range	Mean ± Standard Error
CNN-LSTM	[0, 0.12]	0.02857 ±0.05714
CNN-BiLSTM	[0, 0.202]	0.1062 ±0.0883
SCB-LSTM	[0, 0.162]	0.0691 $\pm 7.474 \times 10^{-2}$
Stacked BiLSTM	[0.011, 0.30]	0.17 ±0.10
Stacked LSTM	[0, 0.15]	0.074 ±0.0641

Table 5.6: Confidence interval for precision using only close events

Results	Range	Mean ± Standard Error
CNN-LSTM	[0, 0.0021]	0.0004 ±0.00094
CNN-BiLSTM	[0, 0.01322]	0.0061 ±0.00536
SCB-LSTM	[0, 1.092×10^{-2}]	0.003915 ±0.0042756
Stacked BiLSTM	[**0.002**, **0.0623**]	**0.03 ±0.022**
Stacked LSTM	[0, 0.0089]	0.00492 ±0.00409

Table 5.7: Confidence interval for recall using only close events

Results	Range	Mean ± Standard Error
CNN-LSTM	[0.810,0.830]	0.00440 ±0.8265
CNN-BiLSTM	[0.818,0.826]	0.8265 $\pm 3.168 \times 10^{-3}$
SCB-LSTM	[**0.862**, **0.995**]	**0.910 ±5.766** $\times 10^{-3}$
Stacked BiLSTM	[0.820, 0.825]	0.822 $\pm 2.160 \times 10^{-3}$
Stacked LSTM	[0.8226, 0.82574]	0.8244 ±0.001187

Table 5.8: Confidence interval for accuracy using close events

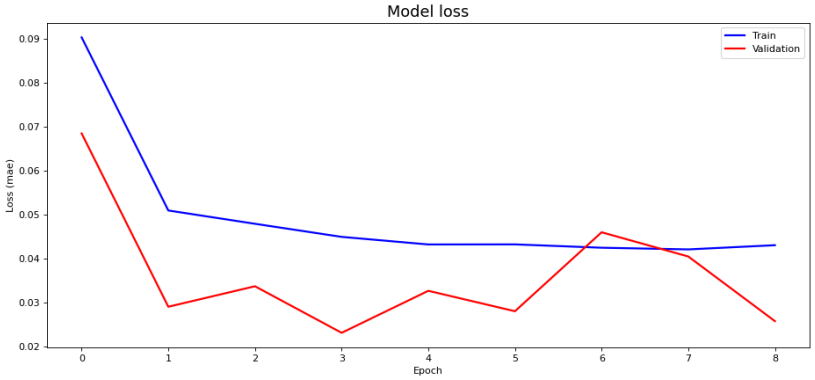

Figure 5.12: Train and validation learning curves for CNN-LSTM on synthetic data

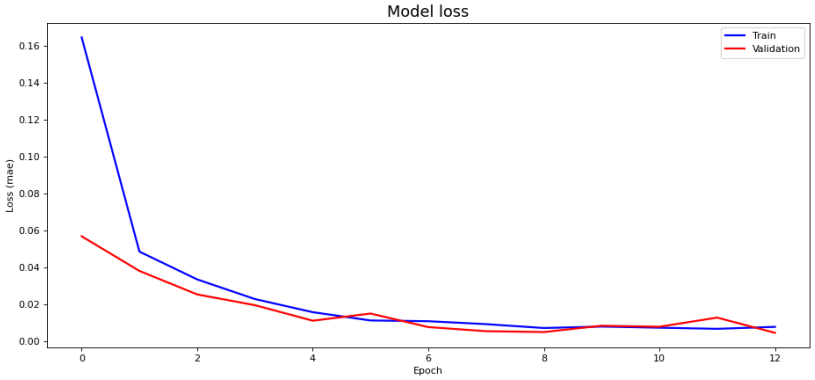

Figure 5.13: Train and validation learning curves for Stacked BiLSTM on synthetic data

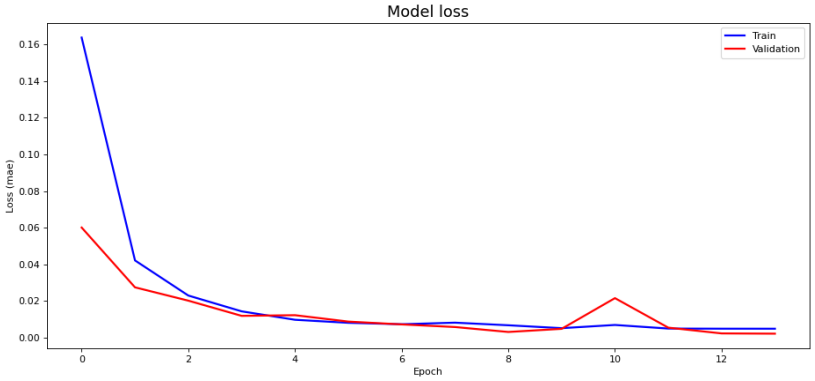

Figure 5.14: Train and validation learning curves for Stacked LSTM on synthetic data

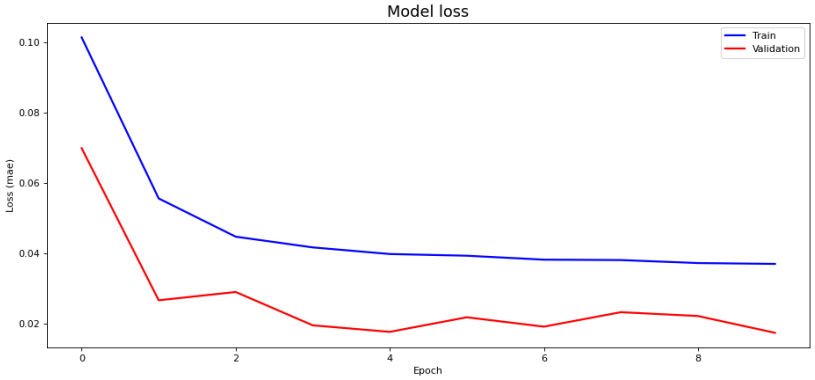

Figure 5.15: Train and validation learning curves for SCB-LSTM on synthetic data

12%. Each anomalous sequence has a percentage of anomalies of 19-10%. The sensor 6 was chosen for analysis because it has more events close to it. For the first experiment, all events were considered.

For multi-head CNN-LSTM, 2 sensors were considered: sensor 6 (flow) and 7 (pressure). ROC curve for Multi-head CNN-LSTM had a very bad performance so it was not considered for further experiments.

Stacked BiLSTM had the lowest training and validation error. For that, it trades training time. Stacked architectures are the ones with the longest training time. Convolutional layers as encoders are very fast to converge, but its SCB is the fastest.

For the second experiment, events were filtered to that they were close to sensor 6. Results were mediocre, as expected by the simulated data because it did not detect close events probably because of the flow direction.

Removing seasonality made the train time to be shorter. Loss was overall smaller too.

The ROC curve improved using the corrective model 5.25 compared to the ROC curve without it 5.24.

Results for the test sequence are worst removing seasonality. This is due to the loss being lower. In these types of models, too much generalization can also generalize anomalies.

Same for the test sequence using the corrective model. The small train size is a cause.

Analysis up close of the events is done from 10 December (Sunday) to 17 December(Sunday). A burst was detected on 12 December. The event was considered to start at 2017-12-10 12h26 (2 days before perception) and end at 2017-12-12 16h50 (open valves). That week also had abnormal periods - most probably unusual demands. The abnormal classification covers situations where the model detected an anomalous period, indicating that an event of some type did occur analyzing the time series, but for which there was no correlation with any further information from the WME database. Assuming these demands for this week, the first experiment using CNN-LSTM revealed 4440 anomalous points, from which 22% were abnormal events, 5% related to the burst, and 73% are ghosts. For the first experiment testing for CNN-BiLstm for 2471 anomalous, from which 4% burst related, 15% correspond unusual demand, and 81% are ghosts. Testing for SCB-LSTM detected 2379 anomalous points, from which 7% burst related, 24% correspond to unusual demand and 69% are ghosts. Testing for stacked BiLSTM revealed 5729 anomalous points, 5% burst related, 24% unusual demand and 71% are ghosts. Testing for stacked LSTM revealed 5946 anomalous points, 5% burst related, 22% unusual demand and 73% are ghosts.

Results	Validation error	Training error	Training time (s)	Epochs
CNN-LSTM	4.708×10^{-2}	4.268×10^{-2}	224	17
CNN-BiLSTM	3.310×10^{-2}	4.005×10^{-2}	215	**11**
SCB-LSTM	6.566×10^{-2}	6.523×10^{-2}	**136**	19
Stacked BiLSTM	$\mathbf{5.279}\times10^{-3}$	$\mathbf{5.207}\times10^{-3}$	3314	21
Stacked LSTM	2.150×10^{-2}	1.460×10^{-2}	3548	26
Multi-head CNN-LSTM	2.800×10^{-2}	3.685×10^{-2}	438	18

Table 5.9: Analysis of convergence for all architectures using real data

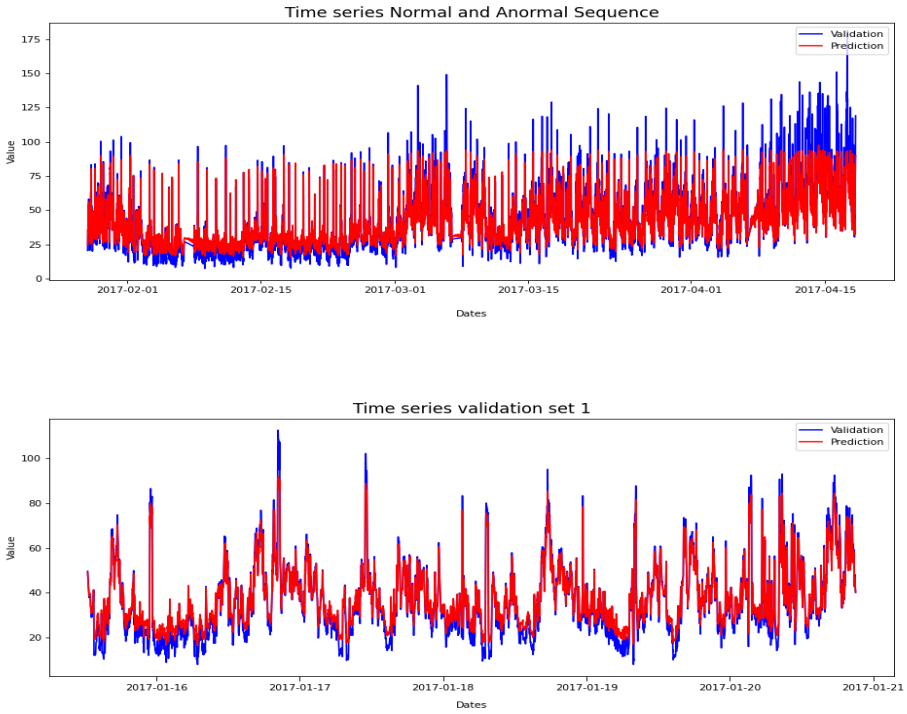

Figure 5.16: Comparisons between the original and reconstructed sensor readings on validation sets predicted by a CNN-LSTM trained with real data.

Results	Accuracy	Precision	Recall	True positive rate	False positive rate
CNN-LSTM	0.91	0.39	0.014	0.74	0.98
CNN-BiLSTM	0.91	0.32	0.023	0.54	0.98
SCB-LSTM	0.91	0.42	0.006	0.85	0.99
Stacked BiLstm	0.91	0.51	0.13	1.0	0.87
Stacked LSTM	0.91	0.36	0.09	0.64	0.92

Table 5.10: Results for real data for 1 September to 30 December of 2017 for 6 bursts

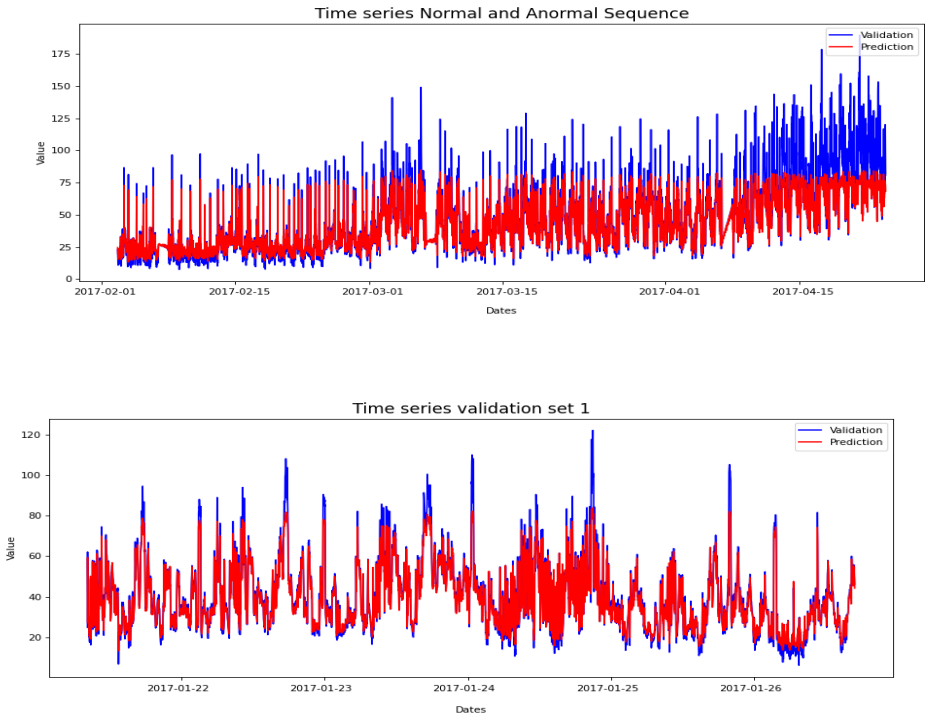

Figure 5.17: Comparisons between the original and reconstructed sensor readings on validation sets predicted by a CNN-BiLSTM trained with real data.

Results	Validation error	Training error	Training time (s)	Epochs
CNN-LSTM	3.946×10^{-2}	3.886×10^{-2}	152	**8**
CNN-BiLSTM	3.310×10^{-2}	3.367×10^{-2}	138	17
SCB-LSTM	3.523×10^{-2}	4.448×10^{-2}	**123**	17
Stacked BiLstm	$\mathbf{5.094}\times10^{-3}$	$\mathbf{4.734}\times10^{-3}$	2408	21
Stacked LSTM	1.268×10^{-2}	1.113×10^{-2}	2886	21

Table 5.11: Analysis of convergence for all architectures for real data removing seasonality

Results	Accuracy	Precision	Recall	True positive rate	False positive rate
CNN-LSTM	0.83668	8.067×10^{-2}	6.920×10^{-2}	0.836	1.014
CNN-BiLSTM	0.7439	8.8565×10^{-2}	0.182	0.925	1.018
SCB-LSTM	0.749	8.390×10^{-2}	0.164	0.872	1.030
Stacked BiLstm	0.820	9.40×10^{-2}	0.106	0.988	1.0013
Stacked LSTM	0.390	9.60×10^{-2}	0.64	1.01	0.97

Table 5.12: Results for real data removing seasonality for 1 September to 30 December of 2017 for 6 bursts

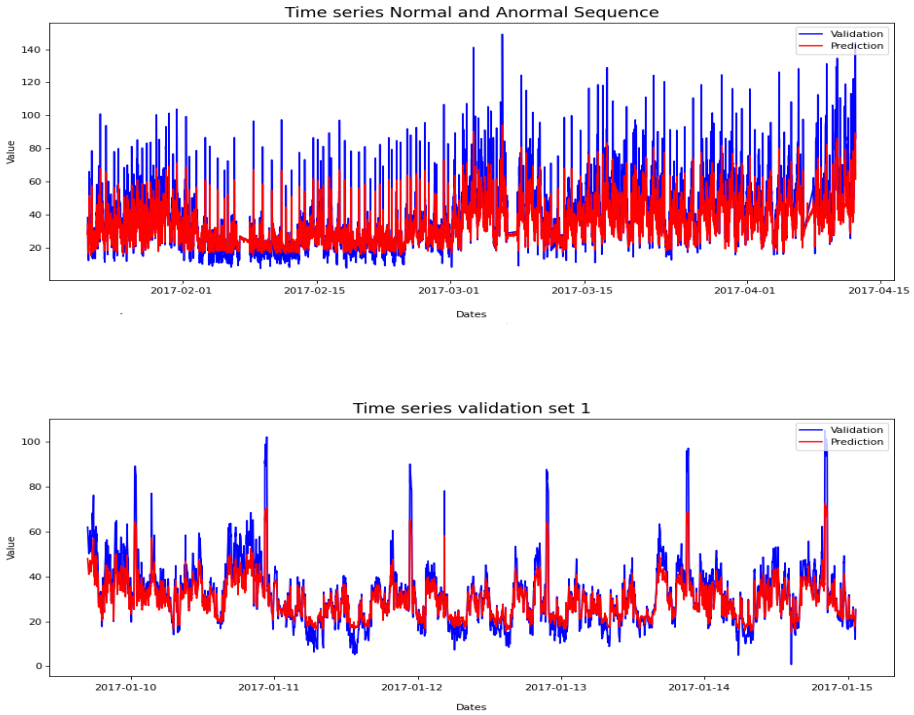

Figure 5.18: Comparisons between the original and reconstructed sensor readings on validation sets predicted by a SCB-LSTM trained with real data.

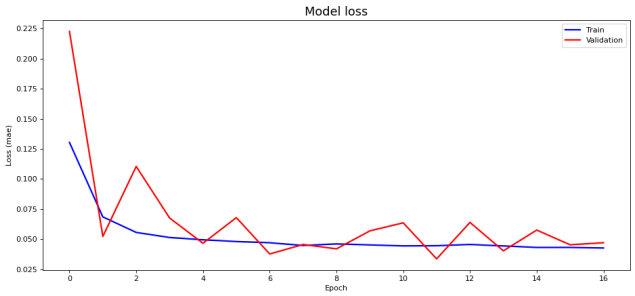

Figure 5.19: Train and validation learning curves for CNN-LSTM on real data

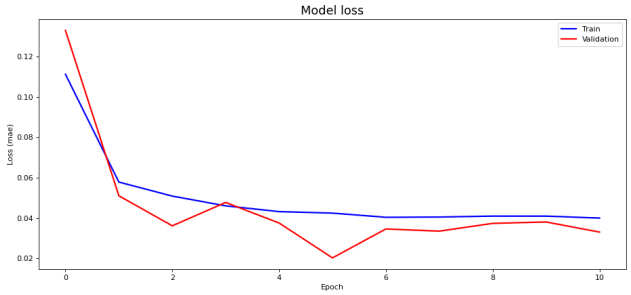

Figure 5.20: Train and validation learning curves for CNN-BiLSTM on real data

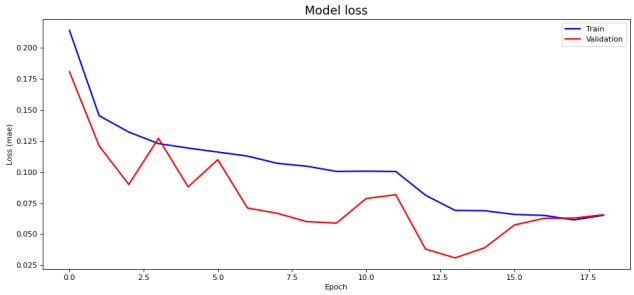

Figure 5.21: Train and validation learning curves for SCB-LSTM on real data

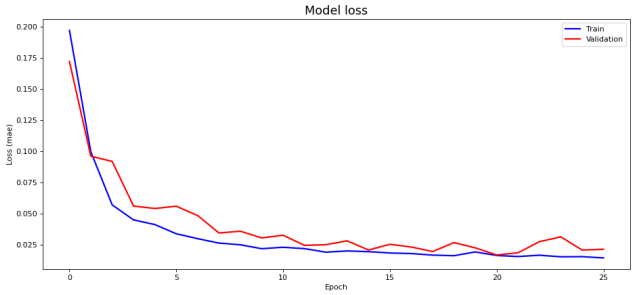

Figure 5.22: Train and validation learning curves for stacked LSTM on real data

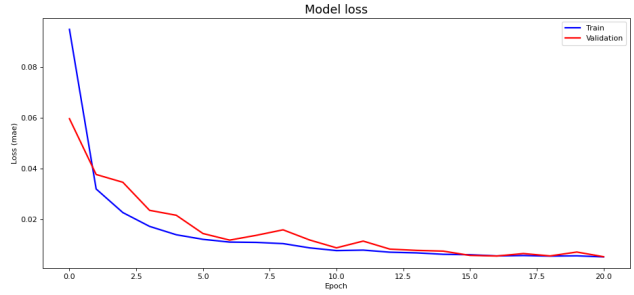

Figure 5.23: Train and validation learning curves for stacked BiLSTM on real data

Results	Accuracy	Precision	Recall	True positive rate	False positive rate
CNN-LSTM	0.657	7.546×10^{-2}	0.233	0.781	1.093
CNN-BiLSTM	0.784125	9.566×10^{-2}	0.152	1.012	0.998
SCB-LSTM	0.480	9.175×10^{-2}	0.505	0.967	1.0364
Stacked BiLstm	0.880	4.274×10^{-2}	1.252×10^{-2}	0.42727	1.0172
Stacked LSTM	0.8528	9.531×10^{-2}	6.540×10^{-2}	1.008	0.999

Table 5.13: Results for real data using corrective model for 1 September to 30 December of 2017 for 6 bursts

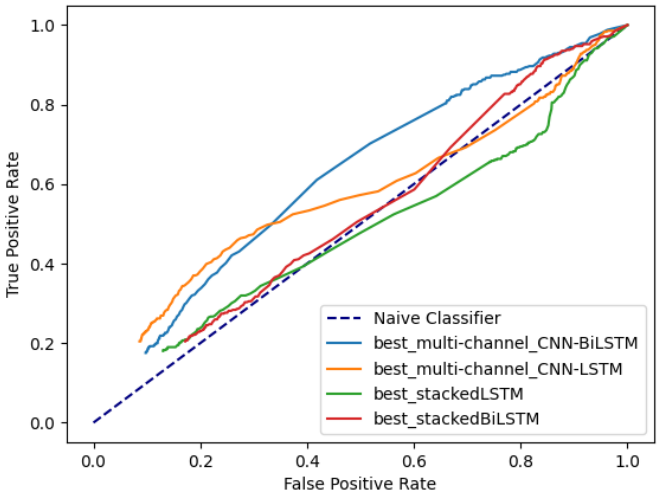

Figure 5.24: ROC curve for real data

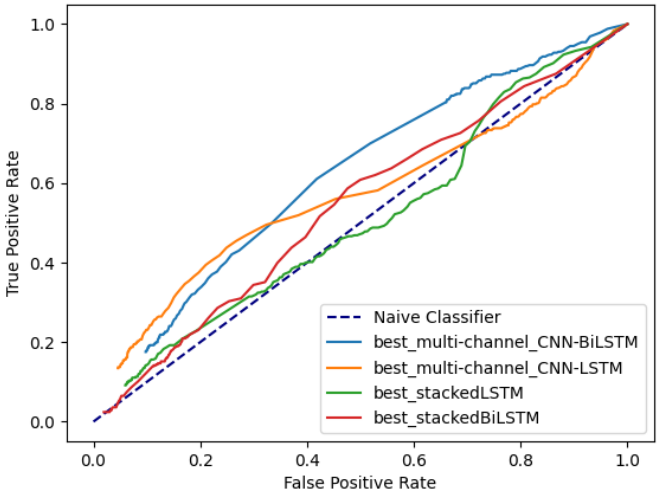

Figure 5.25: ROC curve for real data using correction model

6

Conclusion

Contents

6.1 Discussion

In this section, the obtained results are discussed in a general way, mainly identifying some of the factors that affected the performances obtained.

This work compared multiple deep learning architectures, in a complex scenario such as the targeted anomaly detection in a WDS. The majority of works that use these types of network models consider very simple datasets [17, 50]. One of the objectives of this work was to ascertain if there was a real need to make a database of interesting patterns or leak labeling [13, 56], since this is a costly task. Since deep learning architectures usually learn well in raw sequences [17], and these types of neural networks were only used in water demand forecast [55], this is the gap that it is to be filled. Models were applied as an unsupervised problem. The common practice in the unsupervised anomaly detection is to assume all the training data as normal. However, this assumption does not always hold in practice since the outliers occur within the typical process in many systems. Thus, such methods suffer in performance when the training data is polluted with the anomalies. Notice that just data points related to reported leaks were removed. It could be polluted with anomalies Generally, autoencoders are prone to overfitting. Besides, for a deep learning data problem, our data is small being even more prone to overfitting. They can learn the noise of train data, leading to overfitting and small reconstruction errors of anomalies. To prevent this problem dropout was used. There are more false positives than false negatives, which can indicate the possibility that the model is underfitting because regularization is too strong. For overfitting, there are more false negatives. In certain graphs, there is a generalization gap. For synthetic data, this generalization graph is bigger than for real data. However, graphs do not seem to be overfitting. The signature of overfitting is when validation loss starts increasing and starting loss starts decreasing. Due to these facts, small reconstruction errors are not due to overfitting but due to a very good generalization, which in this task is not ideal. Generalization can make the model generalize anomalies.

These architectures can learn good representations of the input data, having a low loss. Stacked Bi-LSTM is the one that stands out in this matter, due to the characteristics of Bi-LSTM to learn both past and future dependencies. All architectures assume a reconstruction error, assuming that an anomalous data point would have a high reconstruction. However, this assumption is not always true. Since a deviation outside the confidence interval for a particular singular time step does not always indicate an anomaly. These architectures can generalize so well that can also generalize anomalies [65]. This can be seen especially in the analysis of simulated data, where the scenario is ideal because all data points are labeled. Even though normal sequences are learned almost perfectly by the stacked LSTM or stacked Bi-LSTM, some leaks are considered as normal. Thus, models with lower loss perform worse in classifying leaks.

Loss obtained in architectures CNN-LSTM and CNN-BiLSTM was very similar to the work of urban water prediction [55]. In [55] Data train size was 3 years and this research used only 7 months, achieving

similar results in reconstruction loss. The loss obtained in architectures SCB-LSTM, stacked LSTM, and stacked BiLSTM was very similar to loss obtained from work detection anomaly for sensory data [17]. This work had better results in the test set for stacked BiLstm but used very simple scenarios. Again, this scenario is not fair since is very simple. Hu et al. also state that the temperature correction model improved accuracy prediction 1% 3% [55]. In this work, the application of this model worsens the results. This is because they used 3 years for the train set, and we just had 7 months. It is to remark that much existent work has highly controlled trials and thus makes a bad comparison. They also have very little time of test or very few case tests.

It was not possible to differentiate a specific deviation for bursts and other types of events. This is because different events in the WDS have similar behavior. For example, high flow can be caused by water loss resulting from a pipe burst, but other activities can also cause this, such as unusual demands, unknown activity, maintenance activities, or significant short-term increases due to some unknown consumption. However, the detection of these activities can also be interesting. Their detection provides an additional check for the entity. No other work differentiated deviations. In [9] Mounce used SVR. SVR only needs 3 months of data to be trained. The testing data was 6 months. In these 6 months, it only produced 46 detections. The window was 12 hours. For our system, its only needed 90 minutes. However, cite Mounce a larger window makes less false alarms for the case of SVR [9]. Thresholds found by SVR were not able to differentiate between events too. Low recall are very similar to work [50]. However, precisions are not comparable (0.92 and 1). Again scenarios were much simpler. Malhotra's work of anomaly detection was the one that used similar evaluation metrics as this research (precision, recall, accuracy). Overall, the loss is comparable to other research using recurrent neural networks. Even though we had similar convergence in our work considering less data, the gather performance is still not comparable with the referenced performance in alternative domains. We also found factors that contribute to anomaly detection such as the leak size, the sensor position, flow direction, other events not reported at the WME database. There is a trade-off between the generalization because too much generalization can also generalize anomalies, especially if they are of short duration.

6.2 Concluding remarks

In this research, architectures could effectively extract the characteristics of water flow. The best model to detect leaks considered was CNN-BiLSTM, even though stacked Bi-LSTM was the one with the lowest loss. This research was able to analyze differences between techniques and it also identified the problems at this specific scenario. The identified advantages of these models are:

- The idea of AEs is straightforward and generic, even for complex data as in the target scenario.
- Different types of AE variants can be built to perform anomaly detection, yielding different prop-

erties.

- There is no need for labeled data.
- It has reasonable results for a small dataset comparing with the sizes of the datasets of other works.
- Good performance without the need to preprocess data.

The disadvantages of reconstructions models:

- The learned feature representations can be biased by infrequent regularities, the presence of anomalies, or noise in the training data.
- Thresholds are difficult to set, being very sensitive and exhaustive to find.
- Assumption that the training data is a normal sequence.
- Amount of training data necessary is still bigger than traditional machine learning methods.
- Setting the ideal regularization. It can easily underfit or overfit.
- If it has a good fit, it is possible to also generalize anomalies.

Notice that for real data, data was polluted with non-burst anomalies. It was only removed anomalies related to bursts, which were reported by the workers of the WME. However, some bursts may have not been reported or had incomplete information about dates. There are specific problems associated with WDS anomaly detection. Leaks are easily confused with other events, such as unusual demands or maintenance activities. Besides that, it is hard to verify events detected by the models because sometimes there is no data about them. In synthetic data, it was verified that depends on the leak size. Bigger leaks and close to sensors are easily detected. The flow direction has also influence. Leaks can also be confused with demand if very close to the consumption point. For the spacial location of leaks as many constraints in this research. This is due to the location of the sensors and the scarce number of pressure sensors. Pressure sensors would have given some other type of information. It is possible to detect leaks close to a sensor with greater intensity, but it also can detect other events occurring in other areas of the network. There is a necessity for other principles to overcome the identified problems.

6.3 Future Work

These architectures were identified has generalizing anomalies. To solve this drawback, a negative learning technique could be applied. This approach controls the compressing capacity of an autoencoder by optimizing the objectives of normal and abnormal data. But this would only work if data was labeled or partially labeled [65]. Variational Autoencoders (VAE) would work without the need of labels [66]. They

have the advantage of not needing to set a threshold. The reconstruction probability is more objective than the reconstruction error. The reconstruction probability is a probabilistic measure that considers the variability of the distribution of variables.

It is possible to create different types of neural network layers and architectures under the autoencoder framework. It is an area that still requires research. To solve the problem of the normal sequence being polluted with anomalies or having noise, the idea of RPCA may be used in AEs to train more robust detection models [67]. This method learns a nonlinear subspace that captures the majority of data points while allowing for some data to have arbitrary corruption. Another alternative is, instead of exclusively using loss based on the distances between series for training, sequences with labeled leaks could be used, and loss would be penalized if the model would reconstruct leaks wells. The worse the leak reconstructions, the lower would be the loss.

Since finding a good regularization is difficult a solution is not to regularize at all. Data is randomly partitioned into many equally sized parts, overfit each part with its autoencoder, and to use the maximum overall autoencoder reconstruction errors as the anomaly score [68].

In the area of this scenario, more studies should be performed. What could be concluded is that to be sure in a real dataset, a data expert must verify the results. Tests should be also taken on sight so information about events is more reliable. A dataset of patterns should be build so the real scenarios could be more controlled. It is important to define clear leak zones, to help to locate the leaks.

With the insight obtained, these architectures can help to detect bursts. This could help to relax the already existent alarm on the WDS, reducing the number of false alarms. Also, the results obtained can help a domain expert, a civil engineer, to make decisions or to find interesting patterns. These models can select certain sequences that, under expert validation, can be annotated and later used for developing supervised methods.

Bibliography

[1] K. Greff, R. K. Srivastava, J. Koutník, B. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 2222–2232, 2017.

[2] P. Gleick, "The world's water, vol. 7, the biennial report on freshwater resources," *SERBIULA (sistema Librum 2.0)*, 01 2011.

[3] J.-M. Faures, J. Burke, and H. Turral, "Climate Change, Water and Food Security," eSocialSciences, Working Papers id:7821, Dec. 2015. [Online]. Available: https://ideas.repec.org/p/ess/wpaper/id7821.html

[4] D. Hawkins, *Identification of outliers*, ser. Monographs on applied probability and statistics. London [u.a.]: Chapman and Hall, 1980. [Online]. Available: http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+02435757X&sourceid=fbw_bibsonomy

[5] D. Loureiro, C. Amado, A. Martins, D. Vitorino, A. Mamade, and S. T. Coelho, "Water distribution systems flow monitoring and anomalous event detection: A practical approach," *Urban Water Journal*, vol. 13, no. 3, pp. 242–252, 2016. [Online]. Available: https://doi.org/10.1080/1573062X.2014.988733

[6] K. Adedeji, Y. Hamam, B. Abe, and A. Abu-Mahfouz, "Leakage detection and estimation algorithm for loss reduction in water piping networks," *Water*, vol. 9, pp. 1–21, 10 2017.

[7] M. Zhou, Z. Pan, Y. Liu, Q. Zhang, Y. Cai, and H. Pan, "Leak detection and location based on islmd and cnn in a pipeline," *IEEE Access*, vol. 7, pp. 30 457–30 464, 2019.

[8] P. Cheung, G. Girol, N. Abe, and M. Propato, "Night flow analysis and modeling for leakage estimation in a water distribution system," 01 2010.

[9] S. Mounce, R. Mounce, and J. Boxall, "Novelty detection for time series data analysis in water distribution systems using support vector machines," *Journal of Hydroinformatics*, vol. 13, no. 4, pp. 672–686, 11 2011.

[10] J. Mashford, D. Silva, D. Marney, and S. Burn, "An approach to leak detection in pipe networks using analysis of monitored pressure values by support vector machine," 01 2009, pp. 534–539.

[11] J. Gertler, J. Romera, V. Puig, and J. Quevedo, "Leak detection and isolation in water distribution networks using principal component analysis and structured residuals," in *2010 Conference on Control and Fault-Tolerant Systems (SysTol)*, Oct 2010, pp. 191–196.

[12] S. R. Mounce and J. Machell, "Burst detection using hydraulic data from water distribution systems with artificial neural networks," *Urban Water Journal*, vol. 3, no. 1, pp. 21–31, 2006. [Online]. Available: https://doi.org/10.1080/15730620600578538

[13] S. Mounce, J. Boxall, and J. Machell, "An artificial neural network/fuzzy logic system for dma flow meter data analysis providing burst identification and size estimation," 01 2007, pp. 313–320.

[14] T. J. J. A. J. B. S. R. Mounce, R. B. Mounce, "Pattern matching and associative artificial neural networks for water distribution system time series data analysis," 2014.

[15] M. Cañizo, I. Triguero, A. Conde, and E. Onieva, "Multi-head cnn-rnn for multi-time series anomaly detection: An industrial case study," *Neurocomputing*, 07 2019.

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[17] K. Lee, J.-K. Kim, J. Kim, K. Hur, and H. Kim, "Stacked convolutional bidirectional lstm recurrent neural network for bearing anomaly detection in rotating machinery diagnostics," 07 2018.

[18] J. Chen, S. Sathe, C. C. Aggarwal, and D. S. Turaga, "Outlier detection with autoencoder ensembles," in *SDM*, 2017.

[19] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," 04 2015.

[20] R. Puust, Z. Kapelan, D. Savic, and T. Koppel, "A review of methods for leakage management in pipe networks," *URBAN WATER JOURNAL*, vol. 7, pp. 25–45, 02 2010.

[21] S. Boyer, *SCADA: Supervisory Control and Data Acquisition*, ser. An Independent learning module from the Instrument Society of America. International Society of Automation, 2010. [Online]. Available: https://books.google.pt/books?id=5q-qQAAACAAJ

[22] R. Cobacho, F. Arregui, J. Soriano, and E. Cabrera, "Including leakage in network models: An application to calibrate leak valves in epanet," *Aqua*, vol. 64, p. 130, 03 2015.

[23] R. A. D. Peter J. Brockwell, *Introduction to Time Series and Forecasting*. Springer-Verlag, New York, 1996.

[24] J. Han, M. Kamber, and J. Pei, "Data mining concepts and techniques third edition," *The Morgan Kaufmann Series in Data Management Systems*, pp. 83–124, 2011.

[25] H. Abdi, H. Edelman, D. Valentin, B. Edelman, SAGE., and i. Sage Publications, *Neural Networks*, ser. Neural Networks. SAGE Publications, 1999, no. n.º 124. [Online]. Available: https://books.google.pt/books?id=ZEID3w9STOUC

[26] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[27] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Outlier detection for time series with recurrent autoencoder ensembles," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 2725–2732. [Online]. Available: https://doi.org/10.24963/ijcai.2019/378

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[29] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[30] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, March 1994.

[31] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 103–111.

[32] I. Danihelka, G. Wayne, B. Uria, N. Kalchbrenner, and A. Graves, "Associative long short-term memory," 02 2016.

[33] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[34] K.-L. Du and M. Swamy, *Associative Memory Networks*, 12 2014, pp. 187–214.

[35] T. Masters, *Practical Neural Network Recipes in C++*. Academic Press, 1993. [Online]. Available: https://books.google.pt/books?id=7Ez_Pq0sp2EC

[36] J. C. W. Y. Fuchun Sun, Jianwei Zhang, *Advances in Neural Networks - ISNN 2008*, 2008.

[37] B. Kosko, "Bidirectional associative memories," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 1, pp. 49–60, Jan 1988.

[38] D. Gil, A. Ferrández, H. Mora, and J. Peral, "Internet of things: A review of surveys based on context aware intelligent services," *Sensors*, vol. 16, p. 1069, 07 2016.

[39] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," vol. 12, 01 1999, pp. 582–588.

[40] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander, "Lof: Identifying density-based local outliers." vol. 29, 06 2000, pp. 93–104.

[41] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, Dec 2008, pp. 413–422.

[42] C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, Dec 2016, pp. 1317–1322.

[43] J. Serras, "Outlier detection for multivariate time series," 2018.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[45] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *CoRR*, vol. abs/1611.03530, 2016. [Online]. Available: http://arxiv.org/abs/1611.03530

[46] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006. [Online]. Available: http://dx.doi.org/10.1162/neco.2006.18.7.1527

[47] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 338–342, 01 2014.

[48] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb 1989.

[49] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," *CoRR*, vol. abs/1303.5778, 2013. [Online]. Available: http://arxiv.org/abs/1303.5778

[50] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *CoRR*, vol. abs/1607.00148, 2016. [Online]. Available: http://arxiv.org/abs/1607.00148

[51] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional lstm networks for improved phoneme classification and recognition." 01 2005, pp. 799–804.

[52] S. Oehmcke, O. Zielinski, and O. Kramer, "Input quality aware convolutional lstm networks for virtual marine sensors," *Neurocomputing*, 11 2017.

[53] T.-Y. Kim and S.-B. Cho, *Predicting the Household Power Consumption Using CNN-LSTM Hybrid Networks: 19th International Conference, Madrid, Spain, November 21–23, 2018, Proceedings, Part I*, 11 2018, pp. 481–490.

[54] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, June 2018, pp. 125–134.

[55] P. Hu, J. Tong, J. Wang, Y. Yang, and L. d. Oliveira Turci, "A hybrid model based on cnn and bi-lstm for urban water demand prediction," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, June 2019, pp. 1088–1094.

[56] S. Mounce, J. Machell, and J. Boxall, "Online application of ann and fuzzy logic system for burst detection," 08 2008.

[57] S. Mounce, A. Khan, A. Wood, A. Day, P. Widdop, and J. Machell, "Sensor-fusion of hydraulic data for burst detection and location in a treated water distribution system," *Information Fusion*, vol. 4, pp. 217 – 229, 09 2003.

[58] A. Jalalkamali and N. Jalalkamali, "Application of hybrid neural modeling and radial basis function neural network to estimate leakage rate in water distribution network," *World Applied Sciences Journal*, vol. 15, 01 2011.

[59] Z. Rao and F. Alvarruiz, "Use of an artificial neural network to capture the domain knowledge of a conventional hydraulic simulation model," *Journal of Hydroinformatics - J HYDROINFORM*, vol. 9, 01 2007.

[60] J. Austin, "Distributed associative memories for high-speed symbolic reasoning," *Fuzzy Sets and Systems*, vol. 82, no. 2, pp. 223 – 233, 1996, connectionist and Hybrid Connectionist Systems for Approximate Reasoning. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0165011495002588

[61] D. Falessi, J. Huang, L. Narayana, J. F. Thai, and B. Turhan, "On the need of preserving order of data when validating within-project defect classifiers," 2018.

[62] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[63] C. Bergmeir and J. Benítez, "On the use of cross-validation for time series predictor evaluation," *Information Sciences*, vol. 191, p. 192–213, 05 2012.

[64] M. Bakker, H. van Duist, K. van Schagen, J. Vreeburg, and L. Rietveld, "Improving the performance of water demand forecasting models by using weather input," *Procedia Engineering*, vol. 70, pp. 93 – 102, 2014, 12th International Conference on Computing and Control for the Water Industry, CCWI2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877705814000149

[65] M. Stubbemann, T. Hanika, and G. Stumme, "Orometric methods in bounded metric data," in *Advances in Intelligent Data Analysis XVIII - 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27-29, 2020, Proceedings*, ser. Lecture Notes in Computer Science, M. R. Berthold, A. Feelders, and G. Krempl, Eds., vol. 12080. Springer, 2020, pp. 496–508. [Online]. Available: https://doi.org/10.1007/978-3-030-44584-3_39

[66] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," 2015.

[67] R. Chalapathy, A. K. Menon, and S. Chawla, "Robust, deep and inductive anomaly detection," *CoRR*, vol. abs/1704.06743, 2017. [Online]. Available: http://arxiv.org/abs/1704.06743

[68] B. Lorbeer and M. Botler, "Anomaly detection with partitioning overfitting autoencoder ensembles," 2020.