Anomaly Detection and Root Cause Analysis for LTE Radio Base Stations

SERGIO LÓPEZ ÁLVAREZ
Anomaly Detection and Root Cause Analysis for LTE Radio Base Stations

SERGIO LÓPEZ
Abstract

This project aims to detect possible anomalies in the resource consumption of radio base stations within the 4G LTE Radio architecture. This has been done by analyzing the statistical data that each node generates every 15 minutes, in the form of "performance maintenance counters".

In this thesis, we introduce methods that allow resources to be automatically monitored after software updates, in order to detect any anomalies in the consumption patterns of the different resources compared to the reference period before the update.

Additionally, we also attempt to narrow down the origin of anomalies by pointing out parameters potentially linked to the issue.
Sammanfattning

Detta projekt syftar till att upptäcka möjliga anomalier i resursförbrukningen hos radiobasstationer inom 4G LTE Radio-arkitekturen. Detta har gjorts genom att analysera de statistiska data som varje nod genererar var 15:e minut, i form av PM-räknare (PM = Performance Maintenance).

I denna avhandling introducerar vi metoder som låter resurser övervakas automatiskt efter programuppdateringar, för att upptäcka eventuella avvikelser i resursförbrukningen jämfört med referensperioden före uppdateringen.

Dessutom försöker vi också avgränsa ursprunget till anomalier genom att peka ut parametrar som är potentiellt kopplade till problemet.
Contents

1 Introduction 3
  1.1 Background ........................................... 3
  1.2 Different Types of Resources ...................... 6
  1.3 Types of Anomalies ................................. 6
  1.4 Anomaly Detection Techniques ..................... 8
  1.5 Thesis Objective .................................. 8
  1.6 Delimitation ....................................... 8
  1.7 Economical, Social and Ethical Impact ............ 9

2 Related work 10
  2.1 Classification Based Anomaly Detection .......... 10
  2.2 Density and Clustering Based Anomaly Detection . 11
  2.3 Statistical Anomaly Detection ..................... 12
  2.4 Autoencoder Based Anomaly Detection ............. 13

3 Method 15
  3.1 Anomaly Detection Pipeline ....................... 15
    3.1.1 Raw Data ....................................... 15
    3.1.2 Data parsing and Cleaning .................... 17
    3.1.3 Feature Selection for Dimensionality Reduction:
          Random Forest .................................. 18
    3.1.4 Resource Prediction: Quantile Regression Forest
          .................................................. 19
    3.1.5 Anomaly detection ............................ 20
    3.1.6 Feature selection for Root Cause Analysis:
          Autoencoder ..................................... 22
  3.2 Software ........................................... 26
    3.2.1 Python: Anaconda distribution ................. 26
    3.2.2 Keras & Tensorflow ............................ 27
Glossary

**LTE** Long Term Evolution. Standard for high-speed wireless communication for mobile devices and data terminals.


**eNodeB** Evolved Node B. Radio base stations in the E-UTRAN section of LTE. It is connected to the mobile phone core network and communicates wirelessly with mobile handsets (UEs).

**ROP** Result Output Period. Time interval between data aggregation in the radio base stations. It corresponds to a 15 minutes interval.

**PM Counter** Performance Management Counter. Statistical measurements that describe the behavior of eNodeBs regarding their consumption, traffic load and other specific features.

**PDF PM Counter** Probability Density Function Performance Management Counter. **PM Counters** that contain several scalar values in the form of a vector of variable length. This vector may represent the bins in a histogram, or simply different readings grouped following some criteria.

**KPI** Key Performance Indicator. Statistical measurements from **PM Counters**, used to measure the contribution to subscriber perceived quality and system performance.

**UE** User Equipment that communicates with the cellular network.

**ISP** Internet service provider.
PRB Physical Resource Block. Minimum unit of bandwidth allocation for the LTE air interface.

PUCCH Physical Uplink Control Channel. Communication channel that carries control information between eNodeBs and UEs.

PCA Principal component analysis. A broadly used dimensionality reduction technique.

RRC Radio resource control.

CSV Comma separated values. File format that uses a comma to separate values.

XML eXtensible Markup Language. A markup language for encoding documents in a format that is both human-readable and machine-readable.
Chapter 1

Introduction

In data mining, anomaly detection is the identification of items, events or observations which do not conform to an expected pattern or other items in a dataset.


This degree project is carried out at Ericsson AB, a Swedish telecommunications company with world wide presence. The project focuses on anomaly detection for the radio base stations within the LTE technology, a global standard for wireless telecommunication networks based on the older GSM/EDGE and UMTS/HSPA technologies and commonly referred to as 4G. The first LTE network was launched in December 2009 by Telia and Ericsson.

1.1 Background

The high-level network architecture of LTE is comprised of three main components. The User Equipment (UE), corresponds to the mobile phones, computers, tablets, etc, that connect to the Internet using LTE. The Evolved Packet Core (EPC), known as core network, controls the flow of data from the radio access network to the provider of services located in the Internet. It is composed of multiple modules that are responsible for session management, packet routing, packet inspection or quality of service (QoS), among other tasks. Finally, The Terrestrial Radio Access Network (E-UTRAN) is made up entirely by eNodeBs. It
is the air interface that acts as a bridge between the users (UE) and the core network, and its structure is depicted in Figure 1.1.

[14] provides an extensive overview of the LTE and SAE technologies for third generation cellular networks. The document gives a thorough explanation of the evolution of cellular networks during the first two generations, that lead to the development of LTE. It also provides a very exhaustive look at LTE itself, specifically its radio interface, the part of most interest for this project, including all of its components and their functions, used protocols, interfaces and other technical details.

Figure 1.1: LTE E-UTRAN architecture

Anomaly detection in these radio base stations is a big concern for service and telecommunication equipment providers. The performance of eNodeBs is degraded slowly in the presence of faults in the node, decreasing but still providing service to the client. On top of this, problems related to the resource consumption of the node are very likely to go unnoticed until the resources are scarce, since they may not affect enough the quality of the service to trigger costumer
complaints. In a typical situation, nodes receive software updates regularly, in order to introduce or remove features or fix software issues. These updates, however, are also susceptible of introducing anomalies in their behavior.

In this thesis, we introduce methods that allow resources to be automatically monitored after these software updates, in order to detect any anomalies in the consumption patterns of the different resources compared to the reference period before the update. This setup is depicted in Figure 1.2. The reference data prior to the anomaly is assumed to be anomaly free, and represents an scenario in which the node is operating without any issues. The data from the measurement period is not known to be either anomalous or anomaly free, and our task consists on studying its relationship to the reference period in order to decide if there are any anomalies in it.
1.2 Different Types of Resources

eNodeBs are complex systems that consume multiple types of resources while in operation. They are made up by several processing modules, with limitations in their memory and CPU usage, but also consume other resources, such the ones associated to the utilization of the radio spectrum by antennas. Although the list of resources that this project’s work could be applied to is very long, here we briefly introduce only the ones that are mentioned in this report or used as examples of the system’s performance in chapter 4.

- **CPU Load**: The average CPU load on the whole system (all the cores) during the last 15 minutes.
- **PRB**: The X percentile of the utilization for downlink PRB capacity during the last 15 minutes.
- **RRC**: Sum of all sample values recorded for "number of UEs in "RRC_CONNECTED mode", divided by the number samples collected during the last 15 minutes.
- **PDCCH**: Average level of Physical Downlink Control Channel utilization during the last 15 minutes.
- **PUCCH**: Aggregated number of attempts to allocate PUCCH resources in the last 15 minutes.

1.3 Types of Anomalies

Anomaly detection is a wide field, depending on what characteristics of the data are considered to be anomalous [9, p. 2.2]. Sometimes the very value of the data, with respect to typical ones, will be enough to consider a data point anomalous, while other times a perfectly regular value may be considered an anomaly based on its context. In general, anomalies can be classified in three broad categories, that are represented with graphical examples in Figure 1.3:

- **point anomaly**: an individual data instance that is considered as anomalous with respect to the values of rest of the data.
• **contextual anomaly**: an individual data instance that is considered as anomalous only in a specific context, but not otherwise.

• **collective anomaly**: if a group of data instances are anomalous with respect to the entire data set, they are considered as a collective anomaly. The individual data instances in collective anomaly may not be anomalies by themselves, but their occurrence together as a collection is anomalous.

In the case of this project, we focus in collective anomalies, because we are interested in scenarios in which the system misbehaves continuously during an anomalous time period, and not single point anomalies. Even though we are dealing with temporal series, we are not interested in contextual anomalies either, as the load that nodes experience tends to be regular only up to a certain extent. For example, a node that punctually experiences an abnormally high load for some hours due to a business convection in the area it covers, should not be considered to behave anomalously. We define anomalies only as a change in the relationship between the consumption of the studied resource and the values of the rest of **PM Counters** that represent the traffic load in the node. This means that even if the resource consumption rises punctually, the system is not to detect the scenario as an anomaly as long as the rest of the data reflects this increase.
1.4 Anomaly Detection Techniques

Depending on the characteristics of the training data and the techniques utilized, anomaly detection techniques may be classified in three general categories [9, p. 2.3] [20]:

- **Supervised anomaly detection**: Includes techniques that assume the availability of a training data set that contains data instances labeled both as normal and anomalous, with the proportion between the two classes being reasonably balanced.

- **Semi-supervised anomaly detection**: Includes techniques that assume the availability of a labeled training data set that contains only anomaly free data instances.

- **Unsupervised anomaly detection**: Includes techniques that do not assume the availability of a labeled training data set, and are therefore more widely applicable.

In the case of this project, we will work with semi supervised learning techniques, as we have access to the value that the studied resource takes over time, but we assume that all data points in the reference period correspond to an anomaly free working node. In the second part of this thesis, in the process of finding PM Counters potentially related to anomalies, an unsupervised technique is used.

1.5 Thesis Objective

The objective of this degree project is to build an anomaly detection system that is able to detect anomalies in the resource consumption of the node during a measurement period, suggest features as potentially related to the anomaly, in case any was detected. It should be able to achieve this using only the data from a reference period as input, in which the system is assumed to be working normally.

1.6 Delimitation

Due to practical limitations in the amount of training data available for the reference period, we only assess techniques and algorithms that
perform well using small amounts of training data, and evaluate their performance on several scenarios.

The software tools developed in this project are design only as proof of concept of the feasibility of these approaches, and do not constitute deployable software tools.

1.7 Economical, Social and Ethical Impact

The implications of this work can be analysed from different perspectives.

From an economic point of view, detecting anomalies automatically can potentially reduce the maintenance and cost of supervision of the telecommunications infrastructure, as well as save costs derived from any damage, breakdown or failure due to a non detected anomalies affecting the system.

The social impact of this work is linked to the impact that mobile telecommunication networks do already have in society. Of course the specific impact is difficultly measurable by itself, since there are dozens of other mechanisms that could potentially improve the quality of service that users experience in a day to day basis.

From an ethical point of view, this project can be seen as a first step in the direction of fully automatized maintenance for telecommunications networks, which may impact negatively the employability of engineers in the sector. However, The electronic equipment that makes up telecommunication networks has a very high energy consumption, and it is generally not the number one factor of interest for the companies, that will typically deal first with the user satisfaction and experience rather than with a slight increase in energy consumption of the network. Being able to detect anomalies affecting the resources of eNodeBs means that we could potentially reduce the energy drawn by this electronic equipment, which would represent great savings in energy utilization if we consider the thousands of these nodes that provide service across the globe.
Chapter 2

Related work

In this chapter we offer an overview of the most popular work in anomaly detection, explaining the strategies that the different methods follow, as well as their advantages and disadvantages.

2.1 Classification Based Anomaly Detection

Classification based anomaly detection techniques work by training a model based on labelled data instances. The model learns what characterizes each class, and then new test samples can be classified, for example, as normal or anomalous.

Among the most popular and powerful methods we can find Deep Learning. This technique originates from traditional shallow Neural Networks [2], but presents deeper architectures with many hidden layers, that allow the network to find patterns in very complex forms of data. Since 2012, when the so called Deep Learning Revolution introduced many new high performance deep learning networks [28] [11], the capabilities of these models for numerous tasks have skyrocketed. Some variants of them, such as Convolutional Neural Networks (CNN), have been utilized with great success recently to detect anomalies in complex data structures, like images [8]. Another example are Recurrent Neural Networks, with many variants, such as the ones based on the LSTM architecture, also achieving great results in anomaly detection for temporal series [38].

The advantage of deep learning models is that they automatize the process of finding useful features in the data, a process that takes place on their hidden layers, before finally making a classification decision.
based on them. This is useful when the data structures are complex and not easily characterized by simple scalar features. When the data is representable in a simple way using hand crafted features, however, other older techniques, such as Support Vector Machines [12], can potentially be effective as well. This technique learns from the training data the boundaries that separate normal and anomalous points, and detects anomalies as data instances that fall outside of those boundaries. Kernels, such as the Radial Basis Function (RBF) one, can be used to learn complex, non linear boundaries [44].

Random Forest [7], an ensemble method based on decision trees, learn from the training data a series of rules that allow them to separate normal data points from anomalous ones. New test samples are classified by comparing the class of training data points that had similar responses to the same rules [37]. These methods can be considered an evolution of more traditional rule based models, with manually engineered rules, that would require a lot of field knowledge to be constructed.

All of these models have the advantage of having a relatively fast computational time, since they can be precomputed and at test time only the output of the model for the new data instance has to be calculated. However, they also rely heavily on the availability of labelled training data for normal and anomalous data points (or more classes if other outcomes are considered). This, as well as the availability of a balanced training set with similar representation of all classes, cannot be always taken from granted.

## 2.2 Density and Clustering Based Anomaly Detection

Density based methods rely on the definition of the concept of distance between two data points. If the available features are continuous, this can simply be the euclidean distance, but other measures are also popular. The anomaly score of test data samples is calculated then as the inverse of the computed density of its neighbourhood, using techniques such as the distance to the $k^{th}$ neighbour [26].

Clustering based methods, on the other hand, try to group similar data instances into clusters. Several variations of this concept exist, depending on the assumptions they do with respect to the training
data. For example, it is possible to assume normal data instances are the ones that belong to a cluster in the data while anomalies do not belong to any cluster, granted the clustering algorithm contemplates the latter as an outcome [18]. A different criteria could be for example assuming that normal data points are, by a sufficient statistical margin, closer than anomalies to the nearest cluster’s centroid [40].

These methods are some of the most widely used in the industry for anomaly detection. Contrary to classification based ones, they can handle imbalanced datasets with ease, as they can operate in a fully unsupervised way. This is also beneficial because obtaining non labelled data is in practise a much simpler process than labelled. However, they also present challenges and shortcomings. Computationally, density based methods have a relatively high testing phase complexity, since it involves computing the distance of each test instance with all other instances.

The performance of clustering based methods, on the other hand, is highly dependent on the effectiveness of the clustering algorithm in capturing the structure of normal data instances. On top of this, it is not a trivial task to determine the number of clusters the data should be split in, since the number of anomalies potentially affecting the system is unknown.

### 2.3 Statistical Anomaly Detection

Statistical anomaly detection models assume that normal data instances display a higher probability of being generated by a certain stochastic model. These model \( f \) learns the behaviour of normal data points from the training data, and stores it in the form of the model parameters \( \theta \). The anomaly score for a new test data sample \( x \) is then calculated as the inverse of the probability density function \( f(x, \theta) \), that is, the probability that the sample was generated by that model.

A popular example of these techniques are Gaussian based models, that assume the data is generated from a Gaussian distribution. The model’s parameters are estimated using for example Maximum Likelihood Estimation (MLE), and then the anomalies are identified following some criteria, for example as the ones that display a distance to the distribution mean \( \mu \) of more than \( 3\sigma \), where \( \sigma \) is the standard deviation for the distribution.
Another popular approach are regression based models [29]. The idea here is to fit a regression model to the training data, that represents normal data points. This model may be just a simple linear regression model, include normalization techniques, or otherwise be based on other techniques such as SVMs, regression trees, or neural networks. Afterwards, the residual for each test instance is used to determine the anomaly score.

A popular alternative to using the residual is to make use of regression models that can provide a measure of the uncertainty in the prediction, in the form of confidence or prediction intervals [15] [32]. Finally, anomalies can be detected as samples for which the target is not within the confidence interval boundaries.

The main disadvantage of statistical models is that they assume the data is generated from a particular distribution, which is in many cases not true for complex, real world data. They have however several advantages compared to classification, clustering and density based methods. They can use precomputed models, which means a fast computation at test time, can also handle imbalanced datasets, and provide additional information in the form of confidence interval for their predictions, which is very useful in order to measure the uncertainty of the model for every prediction.

2.4 Autoencoder Based Anomaly Detection

Autoencoders are neural networks that compress and reconstruct the data that goes through them. They are trained in an unsupervised way, using the input data also as target, so that the network learns how to reconstruct it efficiently [3].

These models have become popular recently for their ability to model very complex relationships in the data, benefiting from the same factors that have made of neural networks the go-to method for image classification [27].

The basic idea that enables autoencoders to be used for anomaly detection is similar to the one that regression based anomaly detection models use. While in regression models we try to predict a target based on certain inputs, in the case of autoencoders the input and output are the same, and the difficulty relies on the fact that the data has to be represented at some point in a low dimensional space, so recon-
structing it in an error free way is not trivial. The anomaly score is computed in a similar fashion, increasing with the reconstruction error the network achieves.

This concept has been expanded and modified in recent times, and applied extensively to fields such as preventive maintenance for industrial equipment [43], analysis of medical imagery [4] or anomaly detection in surveillance videos [33].
Chapter 3

Method

In this chapter we describe the architecture of the anomaly detection system designed in this project, providing a detailed explanation of the structure and functions of each of the modules that make up its pipeline, depicted in Figure 3.1.

The choice of this specific statistical anomaly detection strategy is motivated mostly by the limitations in the amounts of training data that are available for the experiments in this project, and expectedly for most scenarios it may be used in. Consequently, we have had to opt for a pipeline design that splits the three main parts of the task, prediction, anomaly detection and root cause analysis, and performs each of them using individual, separated techniques. Although there are state of the art methods that bring together these steps into a single model, such as GANs [38], they require the availability of big amounts of training data, and are therefore not suitable for this project due to data scarcity constrains.

3.1 Anomaly Detection Pipeline

3.1.1 Raw Data

The raw data consists on individual PM Counter readings. Each of them represents the value of a PM Counter for a single ROP. They are typically stored in .csv files, with one row corresponding to a single reading. Apart from the value of the reading, that can be scalar or vectorial (vectorial PM Counters are known as PDF PM Counters), the raw data also contains additional information such as the network and
Figure 3.1: Data flow through the Anomaly Detection Pipeline
node of origin or the subclass of the *PM Counter*.

### 3.1.2 Data parsing and Cleaning

The first state of the anomaly detection pipeline is the cleaning and parsing of the raw data. We start off by parsing the data in .csv format, containing individual readings of counters. After aggregating them temporally per ROP, *PDF PM Counters* are expanded to conform individual counters. Missing data is swapped by −1 to allow its identification (*PM Counters* do not take negative values) and *PM Counters* that always remain constant, as they do not encode any information and may pose problems when normalizing the data, are removed. The resulting data structure contains in each row all the readings for a single ROP.

With the readings, that represent in most cases the average value of each *PM Counter* for the last 15 minutes, already aggregated and arranged temporally, we apply normalization to them. *min-max scaling*, which we have found empirically to perform better than *standardization* for our problem, is used. It scales all data values for each *PM Counter* to the interval 0-1, helping machine learning algorithms find relationships between the features by discarding absolute scales, that may differ hugely between *PM Counters*, and are not meaningful for the task.

After this, the studied resource is separated from the rest of the data, and all *PM Counters* that may contain the same information or that can be combined to obtain the exact same readings, are removed. The intention is to keep *PM Counters* that, by describing the processes and activity in the node, are correlated to the resource consumption, but not the ones that contain, or can be combined to form, the exact same information as the one that we want to predict.

For example, in the case of CPU Load, all *PM Counters* describing the traffic load in the node should be preserved, but the ones depicting the load per CPU core should be removed. If we do not remove them, the algorithm will most likely come up with a simple combination of them that allows it to predict the average CPU Load perfectly, ignoring all other factors and always failing to detect any anomalies, as the prediction would always be perfectly accurate.
3.1.3 Feature Selection for Dimensionality Reduction: Random Forest

In this phase we reduce the dimensionality of the feature space, that may contain thousands of PM Counters, and choose a subset of them that, at least for the reference period, is enough to predict accurately the resource consumption. If the prediction for the measurement period samples is not accurate, we will assume that some other factor is influencing the resource consumption in a way that was not happening during the reference period, which may indicate an anomaly.

We select a subset of the most important PM Counters. In our experiments we use 16 of them, but this number is of course susceptible to change, and can be modified in the software implementation. The performance of the system will only increase, at the cost of more computing time, until it reaches a limit, that we have identified in to be close to the chosen number for our experiments. In order to rank the PM Counters by importance (so as to predict the specific studied resource) we use a random forest technique known as Gini importance or Mean decrease impurity. The Sklearn library [35] implements this process based on the description originally proposed in [6].

While in random forests for classification the idea is to measure the feature importance based on the total decrease in node impurity (using the Gini index) achieved when splitting the tree by the studied feature, when the model is being trained for regression we fit a regression model to the target using each of the available features, and compute the Sum of Squared Error (SSE) for each feature and several split subsets of the data. Then the feature that displays the lowest SSE is selected as the most important.

This is an optional phase. The Quantile Regression Forest algorithm can operate on the full set of PM Counters, but the size of this set may vary between different nodes and networks. In order to keep up with time constraints for the analysis, as well as avoid overfitting due to the curse of dimensionality and other related problems, feature selection techniques such as this one based on random forests, can be used to reduce the number of PM Counters used for prediction. On top of this, being able to undercover a subset of PM Counters that encode the relationship between traffic and a certain resource’s consumption is useful because it reduces the need for domain knowledge.
3.1.4 Resource Prediction: Quantile Regression Forest

In this phase, the system attempts to predict the resource consumption at each ROP based on either the full subset of PM Counters, or a subset of them chosen as the most impactful for the studied resource in the previous phase. The reference period data for this subset is used to train a Quantile Regression Forest for this task.

Quantile Regression Forest [32] is a modification of the Random Forest algorithm trained for regression, and so the feature importance to decide by which feature to split the tree at each level is computed using the Gini importance method. This process is then repeated recursively for the remaining features, or until the count of training samples in the split is less than some minimum (that must be enough to allow an accurate quantile estimation), when the splitting stops and the node is called a leaf.

The model’s main unique feature, and what distinguishes it from ordinary random forests, is that the former attempt only at estimating the conditional mean of the response variable, because they do not keep track of all samples in a particular leaf, but only their average. It can be shown that the conditional mean minimizes the expected squared error loss.

\[
E(Y | X = x) = \arg \min_z E \{ (Y - z)^2 | X = x \} \quad (3.1)
\]

Quantile regression forests, on the other hand, do not dismiss this information, keeping track of all samples in each leaf and computing an approximation of the conditional distribution of the response variable given by the probability that for \( X = x \), \( Y \) is smaller than \( y \in \mathbb{R} \).

\[
F(y | X = x) = P(Y \leq y | X = x) \quad (3.2)
\]

The Quantile regression forest algorithm approximates the computation of this conditional distribution as:

\[
F(y | X = x) = P(Y \leq y | X = x) = E(1_{\{Y \leq y\}} | X = x) \quad (3.3)
\]

For a continuous distribution function, this allows us to compute its quantiles, with \( Q_\alpha(x) \) defined so that the probability of \( Y \) being
smaller than $\alpha$ is $Q_\alpha(x)$.

$$Q_\alpha(x) = \inf\{y : P(Y \leq y | X = x) \geq \alpha\} \tag{3.4}$$

This ultimately allows us to compute a prediction interval $I_\alpha(x)$ that, for a given input $x$ and probability $\alpha$, will contain the corresponding observation of $Y$ with a probability $\alpha$.

$$I_\alpha(x) = [Q_{\{1-\alpha\}/2}(x), Q_{\{1-(1-\alpha)/2\}}(x)] \tag{3.5}$$

In this project, we use a prediction interval with $\alpha = 0.95$ for all experiments.

### 3.1.5 Anomaly detection

The purpose of this phase is to compare the estimated resource consumption for the measurement period with the real one and, based in their relationship, determine whether if there are any anomalies on the data.

#### The uncertainty problem

The most important problematic when it comes to detecting anomalies based on the predicted and real resource consumption are high uncertainty situations. If the model’s prediction is confident, meaning that the width of the prediction interval is relatively small, and the real resource consumption is within its margins, we can safely predict that there are no anomalies affecting the system. On the other hand, in case the interval misses the real consumption consistently for a certain time, we can also safely declare that some external factor is affecting the resource consumption, and call it an anomaly.

There are situations, however, when the model displays abnormally wide prediction intervals. An example of this are medium load data points, that are very scarce due to the day-night pattern of the load for most nodes, or pronounced anomalous situations in which the data points differ greatly from the ones the system was trained on. This can end up in a situation in which the real and predicted resource consumption do not match or even diverge from each other, but still sit within the prediction interval margins due to its extreme width, and therefore the system fails to detect any anomalies.
In order to deal with these scenarios we compute a threshold for the maximum acceptable prediction interval width. We train a quantile regression forest on a random subset of 70% of the training data and test on the remaining 30%. The threshold $t$ from which a certain prediction interval is considered anomalous is computed as the Tukey’s threshold for outliers detection over the widths of the obtained prediction intervals, introduced in [19].

Tukey’s criteria divides the prediction intervals into four quarters, with their boundaries defined by $Q_1$, or lower quartile, $Q_2$, or median and $Q_3$, or upper quartile, respectively. The difference between $Q_3$ and $Q_1$, $|Q_3 - Q_1|$, is called the inter-quartile range (IQR). Finally, the threshold for the width of the prediction interval is defined as:

$$t = Q_3 + 1.5|Q_3 - Q_1|$$  \hspace{1cm} (3.6)

Any prediction interval whose width is higher than $t$, will be considered anomalous, although in practice the system will only trigger warnings if the problem persists for a certain period of time.

![Figure 3.2: Example scenario containing all three anomaly detection classes.](image-url)
Outcomes

Formally, the three different possible outcomes that can be observed together in a single scenario in figure 3.2 are defined as:

1. **no issues**: The prediction $p$ is accurate and confident. There are no anomalies in the resource consumption. The real resource consumption $r$ is within the prediction interval $I_\alpha$ margins and the width of this one not exceed the threshold for acceptable prediction interval width $t$ for more than 4 $ROP$s, indicating a confident prediction.

2. **warning**: The real resource consumption $r$ is outside the prediction interval $I_\alpha$ margins or the prediction displays a high uncertainty, with the width of $I_\alpha$ being greater than the threshold for acceptable prediction interval width $t$ for at least 4 $ROP$s.

3. **anomaly**: The real resource consumption $r$ is outside the prediction interval $I_\alpha$ margins consistently for more than 4 $ROP$s. The model underestimates or overestimating the real resource consumption $r$, missing the prediction interval by the same margin for the whole time.

3.1.6 Feature selection for Root Cause Analysis: Autoencoder

Autoencoders are a special type of feed forward neural networks. Their most important characteristic is that they do not need any labeled training data to function. The very same data used as input is used also as the target.
Autoencoders: Structure

Autoencoders are composed of two main parts, an Encoder and a Decoder. The Encoder $\phi$, that comprises all layers from the input to a lower-dimensionality bottle-neck known as the code, shrinks the input data $X$ from its original space to this lower dimensional one, represented as $F$. Then the Decoder $\psi$, that comprises all layers from the code to the output layer, upscales the data back its original dimensionality space. This structure is represented graphically in Figure 3.3.

The fact that the data samples have to go through a bottle-neck means that it is impossible to keep their structure intact. Since in their architecture they are just ordinary feed-forward neural networks, autoencoders are also trained using the back propagation technique, and try to minimize the distance between the label (in this case the same as the input) and the output of the model.

In practice, this means that these networks specialize in finding low dimensionality representations of the data, that allow them to later on reconstruct the data samples back to their original space with a minimal reconstruction error.
Formally, the structure of an autoencoder can be defined by using the Encoder and Decoder definitions such that:

$$\phi : \mathcal{X} \rightarrow \mathcal{F}$$  \hspace{1cm} (3.7)

$$\psi : \mathcal{F} \rightarrow \mathcal{X}$$  \hspace{1cm} (3.8)

$$\phi, \psi = \arg\min_{\psi, \phi} ||\mathcal{X} - (\psi \circ \phi)||^2$$  \hspace{1cm} (3.9)

Where $\mathcal{X}$ is the space the input lives and $\mathcal{F}$ is the space where the low dimensionality representation of the data, the code, lives. Finally, $\psi \circ \phi$ stands for the composed function formed by combining sequentially the encoder and the decoder to form the autoencoder function.

**Autoencoders: Usage for Anomaly Detection**

In the case of anomaly detection, autoencoders can be used to detect outliers in the data. First we train them using the reference period data, that is assumed to be anomaly free, so that the networks learns what defines the structure of normal data, and how to compress it optimally, reducing the reconstruction error between input and output. At test time, new data samples from the measurement period are pushed through the network, and the reconstruction error is computed. The idea is that, since the network has been trained to efficiently compress and reconstruct normal data points, the outliers will display a higher reconstruction error, allowing them to be identified. Contrary to supervised techniques, one of the main advantages of these method is that it does not depend on the availability of labeled anomalies that are difficult and costly to obtain, identify and label, specially in large numbers.

**Autoencoders: Feature Selection for Root Cause Analysis**

In most cases, autoencoders are used for anomaly detection to detect outliers among new data samples. The strategy is to detect samples that, across all their features, display an abnormally high reconstruction error. This can be done for example by averaging the error across all features.
In this thesis project, however, since we are interested in collective anomalies maintained over a period of time instead of single outliers, and because the intent of this specific phase is to identify *PM Counters* potentially correlated to anomalies, we do not focus on the average reconstruction error for each data sample across all its features, but on the reconstruction error for each feature, each *PM Counter*, across all available samples in the test, or measurement period. This distinction is represented graphically in Figure 3.4.

This process is carried out twice. The first time, training on a subset of 70% of the reference data and testing on the rest. The second, training on the complete reference period data and testing on the measurement period data. The reconstruction error for both analysis is compared for each *PM Counter*. If their reconstruction error has varied significantly, it may be an indication that they are related to an anomaly.

**Autoencoders: The network architecture**

We used a simple autoencoder architecture, with only three layers, input, code, and output. The *PM Counter* data is compressed from its original dimensionality, to only eight dimensions, and afterwards reconstructed. The relative simplicity in the correlation between *PM Counters* describing the load in the node and the resource consumption allows a very simple network like this one to already minimize the reconstruction error more than enough to spot inconsistencies in the measurement period with respect to the reference. We use a linear activation function after each layer.

The network is trained for 5 epochs using a batch size of 20 samples and using the Adam optimizer introduced in [25]. This design decisions have been empirically observed to perform slightly better than their counterparts, although in general the improvements over sensible alternatives, such as using ReLu as the activation function for each layer or varying the number of layers of their depth are not very significant.

This very simple architecture is just effectively looking for linear combinations of *PM Counters* that allow it to compress the data. This works because the relationship between CPU Load and many other *PM Counters* that represent the traffic load in the node is simple. However, it may be the case that when trying to apply this method to other
resources or add more input data to the estimation other than *PM Counters*, the network would fail to carry out its task properly because, of non linear dependencies. Therefore, using an autoencoder and not a simpler technique such as vector correlation, is useful because it can be generalized to work in the same way when resources that display more complex relationships with the traffic load are being studied.

![Graphical representation of the reconstruction error evaluation by feature, compared to the more traditional one by sample.](image)

Figure 3.4: graphical representation of the reconstruction error evaluation by feature, compared to the more traditional one by sample.

### 3.2 Software

In order to implement the Anomaly Detection Pipeline, several different software tools have been utilized. All of them are available through open source licenses for educational purposes.

#### 3.2.1 Python: Anaconda distribution

Anaconda is a custom Python distribution. It provides access to a great variety of Python packages that facilitate data science tasks. These are some of the ones that have been used in this project

- *NumPy* [24]: The fundamental package for scientific computing in Python. It is a Python library that provides multidimensional array support, as well as support for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, basic linear algebra, statistical operations, random simulation, etc.

This project makes use of this library throughout the whole implementation, as the data and methods developed in this project
make use of multidimensional tabular data stored in arrays.

- Scikit-learn [35]: It is a Python library that provides tools for data analysis and Machine Learning. The Random Forest, Bayesian Ridge regression and Gaussian Process algorithms used in this project use all this library’s implementation of the method.

- Pandas [24]: It is a Python library providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It allows manipulation and processing of tabular data in an efficient way, and it is used in this project during the parsing and cleaning phases of the Anomaly Detection Pipeline.

### 3.2.2 Keras & Tensorflow

Keras [10] is a high-level neural networks API, written in Python and capable of running on top of several lower level neural networks APIs, such as TensorFlow [1], CNTK, or Theano. It was developed with a focus on enabling fast experimentation. In this project, it is used coupled with Tensorflow, to build the Feed Forward Neural Network method for resource prediction, as well as the Autoencoder Neural Network used inside the Anomaly Correlated Feature Selection module of the Anomaly Detection Pipeline.
Chapter 4

Experiments

In this chapter the performance of the designed system is evaluated on several different scenarios, including anomalous ones, anomaly free ones, and also scenarios that may be of interest due to additional reasons. The results from the anomaly detection phase are always analysed, whereas the ones from the feature selection for root cause analysis phase are only of interest, and therefore presented, in case an anomaly has been detected.

4.1 Data

eNodeBs generate Performance Management data periodically, and store it in XML files. The data is transferred to external databases, from where it can be fetched to perform data analysis on it. This project experiments have been carried out exclusively on data coming from one of these databases. The advantage of working on data coming from this source is the ability to fetch the data already preparsed, and stored in .csv files, which facilities further parsing and data cleaning.

The fetched data presents a tabular structure, with every row corresponding to a single PM Counter reading event. At each row, multiple columns store information regarding time stamp, node network and subnetwork of origin, data type, class of PM Counter, scalar or vectorial value, etc. For the most part, the only columns that contain useful information for anomaly detection are the ones storing the value of the reading and the name of the PM Counter, as well as the time stamp.

A great percentage of the PM Counters display a cyclic, almost binary daily behavior, going up during the mornings and staying in
high, relatively stable load levels (where the amount of people working is higher, known as "busy hour") before decreasing very fast every evening to very low levels (when most people rest). This behavior will turn out to be problematic, as it results in data scarcity for data points outside of these two categories. Specifically, the very fast transitions periods from low to high load that occur every morning and evening, taking no more than an hour, are the only source of data points that represent a medium load for the node. However, it is importance to notice that not all PM Counters display this cyclic behavior. An example of this are PM Counters that are cumulative, storing for example the total amount of packets sent through a certain link, and therefore present strictly increasing patterns. Once again, this behavior will also prove to be problematic because their values are not correlated to the general load that the node experiments at each time step.

4.2 Experiment I: Anomaly Free Scenarios

4.2.1 Resource prediction and Anomaly Detection

This section showcases the system’s resource prediction for several scenarios from different radio base stations and networks. Its intention is to show how the system is able to accurately predict the resource consumption of different resources that follow different patterns, as we can see in image 4.1

These examples illustrate scenarios in which the system works flawlessly, finding no anomalies nor situations susceptible of being considered as so. Even though there are of course variations in both the accuracy of the resource estimation and the confidence of those predictions, they both remain in very low levels for the whole duration of the measurement period. It is important to notice that there could always exist puntual temporal anomalies in the consumption patterns of all resources. However, in order to filter out this unavoidable temporal outliers, we must remember that we consider anomalies only as a continuous mismatch between prediction and real consumption.

Even if, for an anomaly free scenario, we would face a very pronounced puntual temporal anomaly, the existence of the intermediate warning class would allow us to point it out without having to trigger an anomaly. This flexibility of the system is very important because some resources can suffer from temporal anomalies due to real world
(a) RRC connected users

(b) CPU Load
Figure 4.1: Resource consumption prediction for several different nodes and resources.
changes in the load or software processes running in the background, possibly related to maintenance or small software updates. If these seemingly anomalous scenarios, that in reality do not represent any kind of anomaly in the global performance of the node, are maintained over time, the system will be obliged to trigger an anomaly. This situations, however, only tend to affect the system only for short periods of time.

4.2.2 Feature Selection for Root Cause Analysis

This last phase of the anomaly detection process is not performed for any of the examples displayed in this experiment. This is due to the fact that there are no identified anomalies at any point. In the case of warnings, however, the system could be reconfigured to launch this root cause detection process also in their presence, although that is not the case for the configuration utilized for the experiments in this work.

4.3 Experiment II: The Main Anomalous Scenario

4.3.1 Resource Prediction and Anomaly Detection

The Main Anomalous Scenario is a situation in which the CPU load of several eNodeBS within a certain network went up significantly during a whole week, immediately after a software update. The data available for analysis consisted on the .csv files from the an internal company database, covering all ROPs from the week prior to the anomaly, the anomalous week and the one after the anomaly, for a total of three weeks of data. Several thousands of PM Counters remained after parsing and cleaning the data.

The real cause of the anomaly were the attempts of the system to clear its disk space getting rid of temporal information. This process would fail due to a misconfiguration issue, causing the system to retry the same procedure in cycles, with the consequent rise in CPU utilization.

Here we present several experiments carried out in order to test the success of the proposed system in identifying the anomaly and finding PM Counters potentially correlated with it. All scenarios present
Figure 4.2: CPU Load prediction for two different nodes in the network during the Main Anomalous Scenario.
(a) High uncertainty for medium CPU Load

(b) Seemingly temporally anomalous scenario

Figure 4.3
resource consumptions recorded for the same node of the network, with the exception of Figure 4.2b, that corresponds to a different node in the same network.

Figure 4.2 shows a reference-measurement period split for two different nodes in the network, with the measurement period starting when the software update takes place, at the beginning of Main Anomalous Scenario. The CPU load goes up shortly after, and even though it keeps fluctuating daily for the next week, it appears to be affected by an extra, almost constant factor, that increases it at all times. The system, using the same PM Counters that were effective at predicting the resource during the reference period, achieves very high error rates during the measurement period, therefore confirming an anomaly is affecting the CPU Load.

Figure 4.3, on the other hand, shows a couple of special cases from before the start of the anomaly, that do not contain any real problems, but that for different reasons could be suspected to do so by a human because they contain temporal anomalies of some kind. In 4.3a, we train the system based on several days of regular, cyclic data, without anomalies. For the reference period, we choose the next day, in which the load of the node falls during the morning transition period, hovers at medium loads for some time and finally reaches standard levels at the middle of the day. The issue here is the data scarcity of data points representing medium load in the node. This produces a high uncertainty in the prediction of the model, that is however able to predict rather accurately the actual consumption during that period. This scenario did not contain an anomaly, but the high uncertainty in the prediction is a characteristic of anomalous situations, so the system triggers a warning for the correspondent time window.

4.3b shows a temporal, collective anomaly that affected the CPU load in the last hours of November, also prior to the anomaly. Contrary to the observable daily pattern that is display in previous days, the resource consumption of the node stayed high for a very limited number of hours before decreasing rapidly into an almost nightly pattern. Even though this behavior clearly constitutes a temporal anomaly with respect to the ordinary behavior of the node for a weekday (it was a Thursday), the system’s analysis revealed that no anomalies existed, as the CPU load had gone down correctly as a consequence of the decrease in traffic. Moreover, the levels of uncertainty, although higher than normal, were not constantly high, with only intercalated data
Figure 4.4: Prediction of non anomalous resources during the Main Anomalous Scenario.
points displaying high levels, so a warning was not triggered. We do not know what caused the decrease, although it is plausible that it was purposely retired from service temporarily in preparation for an incoming software update. In any case, this example illustrates the difficulties that a model based purely on forecasting based on historical resource data would experience to avoid false positives.

Finally, Figure 4.4 shows resource prediction after the software update for different resource other than CPU load. The intention is to show how no anomalies affect them, with their values being predicted accurately by the system.

### 4.3.2 Feature selection for root cause analysis

Once the anomaly has been identified, the anomalous feature selection phase is triggered. Here we present some of the *PM Counters* suggested by the autoencoder technique as potentially related to the anomaly. The network is trained using back-propagation and all available data prior to the start of the anomaly, including all *PM Counters*. Afterwards, the data within the measurement period that has been labeled as anomalous is pushed through the network, and the average reconstruction error for each *PM Counter* is compared to the reconstruction error it achieved when the same process was carried out training on a 70% split of the training data and testing on the remaining 30%.

Figure 4.5a represents the absolute normalized error for each of the approximately 3000 *PM Counters* available during the Main Anomalous Scenario. Most *PM Counters* that displayed a high reconstruction error belonged to the same class of *PM Counters*, related to the flow of a certain type of data packets from the eNodeBs to the core network.

Figure 4.5b represents the temporal evolution of one of the *PM Counters* pointed out by the autoencoder, in comparison with the evolution of the CPU in the node, for the total duration of the available data. This *PM Counter* registers the number of control data packets sent to the core network at each ROP, and shows how, during the whole duration of the anomaly, the node failed to communicate with the core network completely. This information, in the hands of an expert in the architecture of the LTE network, could facilitate a lot the process of identifying and solving whatever problem is affecting the node.
(a) reconstruction error distribution over available PM Counters

(b) Temporal distribution of anomalous subclass of data packets sent

Figure 4.5
Chapter 5

Conclusions

5.1 Discussion and Limitations

The objective of this thesis project was to build a system that could detect anomalies in the resource consumption of eNodeBs. We designed a prediction-based architecture to detect anomalies as a divergence between real and estimated resource consumption, and additionally defined a threshold for the amount of acceptable uncertainty in the model’s prediction. Moreover, we also designed a feature selection procedure for root cause analysis, that given a detected anomaly, could suggest PM Counters potentially correlated with it, based on an autoencoder analysis.

Our experiments show that the predictive model is able to accurately predict any proposed resource, so that the system detects anomalies as a divergence between the real consumption and the prediction. The erratic behavior of the consumption pattern of certain resources, potentially due to world events or network phenomena, will in many occasions provoke temporal anomalies, but the strategy of identifying these only based on the relationship between the resource and the rest of the data depicting the traffic load filters out in most cases these false positives. Figure 4.3b is an example of this.

Still, data scarcity can impact severely the performance of the system. The cyclic, almost binary behavior of the resource consumption pattern produces a scarcity of data points for intermediate loads, scenarios that typically appear during transition periods between low and high loads. The model will be uncertain about its predictions for these data points, possibly ending up in false positives. High uncer-
tainty is definitely the biggest concern observed in the performance of the system. Even though we deal with this problem by focusing in maintained anomalies that affect the resource consumption continuously for a certain time longer than this transition period, this is still an issue to be targeted, ideally, with the inclusion of more training data. However, as long as this binary pattern remains intact, there will be a huge imbalance between the proportion of medium load data points and high or low ones, a situation than in general is problematic for machine learning algorithms.
Chapter 6

Future work

The potential improvements and future work for this project can be divided broadly into two categories. On the one hand, the ones that consider maintaining the general architecture of the system, but including bigger amounts of training data or modifying it to accept different types of data as input. On the other hand, the ones that include total or partial modifications of the system’s algorithms, in order to potentially increase its performance.

Including more data into the current system could definitely increase its performance a lot. Most of the problems that we face right now have to do with data scarcity, a problem that could be dealt with by increasing the length of the reference period, or even better, by using data that instead of representing the average value of PM Counters for the past 15 minutes, reduces this as much as possible, providing a richer training set for the same time period. Employing other types of data for the estimation, and not only PM Counter readings, could also be very beneficial. Some of these new classes of data could be hardware information of the eNodeBs or records depicting which software modules suffered changes during the update (that separates the reference and measurement periods).

In the second category, modifying the actual methods of the system, there are a lot of potential improvements. The first one could be substituting the manually compiled rules that right now decide what constitutes an anomaly by some automatic algorithm, potentially based in neural networks or a different family of algorithms, that would learn which kind of relationship between prediction and ground truth constitutes an anomaly for this specific kind of problem.
This would be specially helpful in the decision making that involves distinguishing between anomaly free and warning scenarios, a non-trivial task since the amount of uncertainty that is normal varies between networks or even nodes.

Regarding the root cause analysis phase, it would be interesting to develop a filtering system that would deal with the high reconstruction error that some PM Counters display systematically, due to their monotonically increasing nature. Several ideas have been considered for this task, such as derivative based ones.

Finally, it is worth mentioning some potential alternatives to the prediction-based approach taken by this work, that were studied as alternatives at the beginning of the project, and discarded eventually in most cases due to training data limitations. Some of them, such as Generative adversarial Networks, constitute a very interesting approach to anomaly detection, because they also do not need anomalous labeled data to function. They have already been used in other fields for the task of anomaly detection, even for problems that required a root cause analysis, such as in [38], for medical images. Others, such as forecasting models, can still work as prediction-based approaches, but take into account the temporal dimension of the data, that could also be included into the system through recurrent neural networks, allowing us to maintain the usage of PM Counter data for prediction while incorporating the historical evolution of the resource consumption into the estimation.
Bibliography


