POLITECNICO DI TORINO

Department of Electronics and Telecommunications

Master Thesis Course in

Communications and Computer Networks Engineering

Master thesis:

Energy Saving in Cellular Base Stations Under the Control of Supervised Neural Networks

Supervisor:
Prof. Marco Ajmone Marsan
.....

Co-supervisor:
Prof. Antonio Capone
.....

Candidate:
Igor Donevski
.....

December 2018

Acknowledgements

Two years ago, I have started from a point where the concept of Machine Learning was totally alien to me. Nonetheless, I have progressed to develop the concepts presented in this thesis. And for this, I want to express my utmost gratitude to my tutor, over and above my mentor, professor Marco Giuseppe Ajmone Marsan for his dedication, availability and support of my work. His guidance and support for my ideas have given me the leeway to experiment and develop as a researcher. I want to also thank professor Michela Meo for her advising. Likewise, I want to express my dearest thanks to the person that laid the basis for this work and was essential to its progress, Greta Vallero. Finally, thanks to professor Antonio Capone for partaking as my co-supervisor from Politecnico di Milano.

I want to greatly acknowledge the help of the Alta Scuola Politecnica program through these two years. It was immensely supportive in my scholarly progress during the two years as well as providing me with essential financial aid. And most importantly, this program introduced me to my dearest colleagues and friends which provided me with incredible support during my studies.

And finally, I want to dedicate a special thanks to my parents, sister and my grandma for always being the beacon of my success. I further thank my close friends that have provided me the smiles and strength to endure. And last but not least, I am grateful to Aleksandra for her daily support and endless love.

Table of Contents

List of Figures

List of Tables

Abstract

Energy consumption due to ICT is a consistent ever-increasing trend. One of the main culprits is the booming global dependence on cellular devices that obligate expansion of service providing stations. This trend compels an action as a responsibility towards the environment, while nevertheless providing compelling financial drive to investigate decreasing the energy expenditure of present and future cellular architectures. Base Station (BS) switching, where unused stations are switched off, is a viable, potentially a no compromise solution, for both current heterogeneous networks and future ones. The reason being, densification of small base stations servicing areas with high traffic demand is imminent for future increases in capacity. Therefore, if created in a flexible manner, the newer architectures are proven to sustain very high energy efficiency while maintaining next generation speeds. In support, linear load model derived from state of the art measurements for Macro and Micro cellular BS is taken for analysis. Upon which, a novel approach to efficiently calculating the most optimal combination of switching decisions was suggested.

However, as estimation is an integral part to providing a solution in a realistic scenario, Machine Learning (ML) techniques are considered essential. In this research, two approaches for the problem of online BS switching are developed and tested. In the first, referred as two-part solution, ML is used to perform short term load estimations that are later used to calculate the best combination of switching decisions. The other, is a unified solution where both the switching optimization and estimation are performed by one machine. For these two approaches, several potentially useful Artificial Neural Network (ANN) implementations are tested, two main paradigms being Dense Neural Networks (DNNs) and Recurrent Neural Networks (RNNs).

In the end, the unified solutions proved difficult to train requiring massively bigger dataset than the only estimation counterpart. However, many useful results were provided, and the unified solutions were shown to be flexible in controlling a tradeoff between guaranteed coverage and energy efficiency. When adequately adjusted, it is possible to guarantee less than 1e-3 probability of reduced quality of service, while maintaining good energy efficiency. Such solution is considered to have proved the applicability of ANN solutions for the problem of online BS switching in modern cellular architectures and the planning of future ones.

1 – Introduction

1.1 The Energy of Global Communications

A consistent trend in the field of communications is the rise in global traffic demand. According to the traffic measurement analysis performed by Cisco Systems Inc. the amount of traffic per smartphone per month has risen from 1169 MB in 2015 to 1614 [1]. This jump of 38% is held accountable to the increase in connection speeds in cellular networks for more than three times for the same years [1]. As a matter of fact, such rise was well expected as the analysis of the five-year difference in traffic demand is eighteen times bigger for the span of 2011-2016 [1]. As for the near future of 2021, [1] estimates that the average connection speeds of cellular networks will increase threefold which will increase the share of cellular networks in total IP traffic to one fifth.

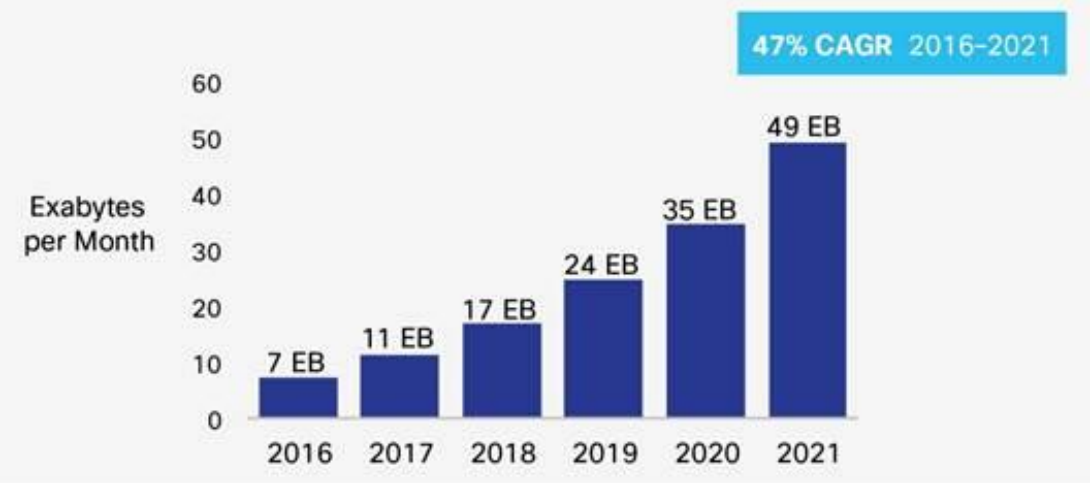


Figure 1 Estimations for per month data demands, Source: Cisco VNI Mobile, 2017 [1]

As impressive as those numbers are, they translate directly into increased energy consumptions. Analyzing past behavior in [2] it is estimated that the total electricity consumption between the years of 2007 and 2012, in all types of communication networks increased by 10.4 percent. In numbers, it jumped from 200 TWh to 330 TWh. And from this pie of energy consumption, telecom operator networks consist of a piece sizing up to 77%, making the 2012 consumption due to telecom to amount to 254 TWh

[2]. In accord, an estimate performed by the Digital Power Group [3], ICT systems consist of 10% spending of the world electric energy or in total number, an estimated 1500TWh annually.

Although those numbers are already big by themselves, the energy demands are expected to further increase in the field of cellular communications. It is in the wireless nature of cellular communications that radio interface technologies are not a very energy efficient method for information transfer. Furthermore, with every step in the evolution of cellular networks the throughput of the covered areas and the quality of service (QoS) per customer is improved. Every generational step brought higher capacity connections to customers by improving the technology and increasing the infrastructure. This is relevant as the newer generation of 5G systems is expected to be brought to market in the following years. In accord, it is expected that in 2021 finally 5G connections will come to consumers and appear as 0.2 percent of total connections [1]. Accounting that an average 5G connection will have the potency to generate 4.7 time more traffic than that of a 4G connection, it shows that as 5G penetrates even more of the market, cellular communications will be an even bigger player in total IP traffic.

However, no matter the efficiency of a newer paradigm, as demand for data rates increases, cellular infrastructure expands and with that the energetic demands of the network. Electricity expenditure in the field of Information and Communication Technologies (ICT) is important as the looming 5G expects to increase data speeds thousand-fold by relying on denser Base Station (BS) placement. In a longer run worst-case scenario estimation for up to 2030 [4], it is expected for communication technologies take a share of 51% in the total global energy demand. Though this estimation is noted to be a worst-case scenario, it must be taken as a motivation to move forward and avoid such scenario by implementing many technologies that would drive the communications to a more sustainable state.

The aforementioned information gives a rational incentive to act upon improving the energy demands of the network. Reducing such power demands can result mainly in a reduction in Operative Expenditure (OPEX) of the company, as well as contribute to reducing the carbon footprint. This advocates towards improving the energy expenditure not only from a profit standpoint for the company, but as a responsibility towards the environment. In accordance, there has been an increase in the interest of the scientific community into trying to increase the energy efficiency of cellular networks. Therefore, in 2010 the Energy Aware Radio and Network Technologies (EARTH), major European research project was initiated with an objective to decrease

the consumption of the communications sector by 50% [5]. The EARTH project has therefore shed a lot of attention on the problem and gathered a lot of manpower towards solving the issue.

1.2 Towards Energy Efficient Communications

One approach from the many possible for increasing the efficiency of the network is Base Station Switching (ON/OFF) which is very promising but difficult to implement. The difficulty comes from the fact that a switched off base stations result in a total death of the services in an area, that is if single tier coverage is assumed. This hindrance may be removed by the introduction of 5G networks which are expected to be implemented through increasing the radio interface infrastructure in many different sizes and hierarchies and form heterogeneous networks (HetNets). This HetNet paradigm is expected to create many overlapping networks of bigger Macro Base Stations (MBSs) and that cover underlying Small Base Stations (SBSs). A matter of fact, a much more exaggerated paradigm may be used in dense urban areas that will provide 1Gbps through extreme BS densification [6].

Organizing heterogeneous networks in this way levels the playing field for base station switching algorithms by increasing the total network power requirements which makes the technology necessary whilst enabling it by the overlapping coverage. Before such change in the paradigm, for a base station to be turned off, it had to possess a measure with which it will monitor requests for service in order to not lose coverage. In comparison, in HetNets, shutting off an idle SBS results only in loss of potential capacity which if well planned can contribute to energy savings without interrupting normal operation of the network. As massive implementation of HetNets ensues, the lasting hindrance of avoiding total service outage in an area is solved and we can safely assume that BS switching is only possible in areas of overlapping coverage. For this reason, many different models have been proposed for implementing Base Station (BS) switching in 5G heterogeneous networks.

Optimizing base station switching has some challenges that still need to be addressed. In response to this there is a plethora of scholar activity to address the many different possible implementations of it, as shown in [7]. Many of the implementations recognize a different culprit for the complexity. But mostly the main issue with switching base stations is that the current equipment is not engineered to sustain very often changes in mode of operation and require special attention when switching. This deters the

service providers to exploit the approach often. By not having the liberty to switch off the BSs on demand requires a precise estimation on the future service requirements of the area. This is a difficult problem that requires advanced understanding of the area's service history and a good estimation tool. In addition, comes the normal combinatorial optimization problem which solves the most efficient combination of BSs to be turned on at one time. A problem which is easy to solve while the number of combinations is still small.

To address these two problems at once requires a robust algorithm that may solve both problems simultaneously or in a two-step manner. By having a lot of success in many diverse fields, machine learning algorithms are deemed very appealing for this issue. More specifically, a subgroup of machine learning methods, Artificial Neural Networks (ANNs), are especially gaining the attention not only of the scientific community but also of the public as very powerful tools for solving complex problems. ANN implementations were able to solve many problems that were deemed impossible for computers. Though the use of ANN for a such a specific task as in solving BS switching optimization is a very novel approach, there has been some attention to the problem. Usually approaches have taken that no estimation is needed or that QoS is guaranteed by other sources. A very simple case is to try and optimize having full state awareness and bare the ANN with the problem of optimization [8]. A more complex example of a solution is [9] where data caching at BS is assumed. The model [9] guarantees some data accessibility and is mainly concerned with delay constraints. Though many different approaches for modelling the problem can be made, this entire research was based on simplifying the structure of the model in favor of showing eligibility of such solution on a very basic scale. In this way, it can be properly analyzed if ANNs poses the power to solve energy optimization in an Online fashion without having any compromise on the quality of service.

Finally, the recent attention to Neural Networks makes the technology much more comprehensible and widely accessible. In example, there are many available Application Programming Interfaces (API) that operate on different layers and simplify working with neural networks. The most widely known is Google's developed TensorFlow, developed by the Google Brain team and released to the public in 2015. There are many other APIs that enable even trivial use of neural networks or a more specialized one. The whole programming work was done with the Python programming language by using the TensorFlow libraries sometimes helped by the Keras API. The decision was done on the basis that TensorFlow is a very well optimized framework for doing Machine Learning research as it is fast, supports hardware acceleration

through Graphical Processing Units (GPUs) on many platforms, and is very intuitive. Furthermore, python is very comprehensible, there is an exhaustive source of guidelines for using the TensorFlow framework as well as very good community support. The possibilities with the APIs that are present are exhaustive and provide very easy learning curve for implementation of neural networks as well. This thesis would not be possible without the community behind Machine Learning that makes the field approachable for anyone.

This thesis is presented in the following manner:

Chapter 2. covers all the important aspects of the underlying network. It will start by explaining the design decisions that were taken. In the terms of the scenario, it justifies using the overlapping coverage, the number of stations and the applicability. It further explains the main issues behind the possible solution for the problem. It introduces the dataset that was available for use, the dataset contents, and the possible usefulness of it. It further carries on to the energy model which guides the analysis. In the end, taking this energy problem, the optimization scenario is analyzed. A successful reduction in the solution search space for the scenario is also achieved by analytic analysis of the problem functions.

Chapter 3. covers all the technologies essential for the development of a solution and the specific implementations that were conducted. It is named Machine Learning and Artificial Neural Networks as it obviously analyses the possibility of using them as solutions to BS switching. It starts off with presenting the evolution of regression models and the evolution of the algorithms that are needed to use learning mechanisms. It justifies the ANN technologies used, and their potential as a solution. Finally, the proposed ML solutions are presented in detail, where their internal structure and their usage is explained.

Chapter 4. displays the results achieved by the proposed. It first introduces the important parameters of the analysis, how they can be achieved and the significance of each into the final analysis. The main parameters are then converted into percentages for easier performance classification. Afterwards the results are shown for each implementation as an average and simplified results. In consequence to this, the best performing algorithms are taken and compared in closer detail to discover the performance structure of the implementations. A summary of all the aforementioned performance analysis is written to compare the many possibilities of implementation and the eligibility of the approach is justified.

This is than followed by a conclusion section, the bibliography and the appendix which contains the non-essential math derivations and codes.

2 – Problem Model and Setting

2.1 Problem Setting

The design decisions behind the creation of this model is to be potentially useful for both existing LTE architectures that have overlapping coverage as well as the expected HetNet paradigm for 5G networks. This is due to that the whole model is approximated to having one big MBS and many SBSs fully under its coverage. There may be a more complex hierarchy in the network such as more than one tier of overlapping coverages. But, to consider that the model needs to analyze different combinations of MBS and SBSs, complicates the optimization problem due to the near infinitely many combinations of expenditures and traffic loads for all hierarchies. Therefore, it is assumed that the possible system to be optimized consists of one MBS and few underlying SBSs that are only one hierarchical tier smaller than the MBS, as shown in Figure 2.

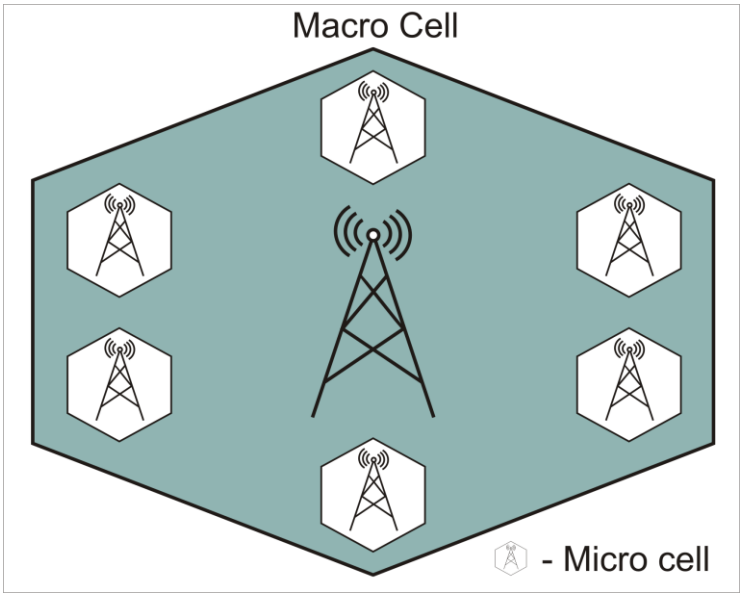


Figure 2 The assumed cellular architecture

The difficulty in implementing the switching paradigm is concentrated around the inflexibility of the commonly used equipment which may get damaged if switched too

often [11]. For this reason, the Quality of Service measurements must be done by calculating coverage for a fixed period, timestep, that is relatively long and support slow activation and deactivation. This requires the system to estimate the usage of the network for that period to avoid bad QoS. If such an event occurs, it is concluded that the QoS for that timeslot/period has been compromised, with an evaluated impact. The evaluation calculates the excess traffic demand that may have been possible if switching decisions have not been made. More details about the decisions for how the QoS measurements are performed, will be provided win the Chapter 4 before presenting the results of the scenarios.

Finally, different approaches in the literature try to balance many different factors in the operation of a BS such as: caching, energy harvesting, battery, handover prediction etc. [7][8][9]. Even though many of such scenarios are possible implementations in the system, the most robust use case is correlating the future load based on past load with a goal to minimize the total energy spending without compromising QoS.

2.2 Traffic Characteristics

To be able to provide adequate analysis, the data used is real traffic that is provided in confidentiality by an Italian mobile service provider. The traffic is fragmented in upload and download traffic over the services of Voice, SMS and Internet data. However, as Internet data overwhelms the other two services in terms of BS load, it is safely approximated as being the only traffic that impacts the analysis. Furthermore, the area is split in a way to hide the real underlying structure, for the purposes of confidentiality, and is represented in arbitrary square areas that aggregate the data of the underlying BSs. The dataset spans across a large area, over the general metropolitan area of the city of Milan summing up to 1419 different areas. These arbitrary areas vary in size to adjust to the underlying traffic demands, smaller squares represent higher traffic demands with more congested BSs while larger areas present smaller traffic demands underneath, as shown on Figure 3. From here on, one individual area will be taken as an approximation of a base station, and neighboring areas a part of a heterogeneous network. This approximation holds because the areas still contain geographical correlation considering the internal traffic to their neighbor-areas.

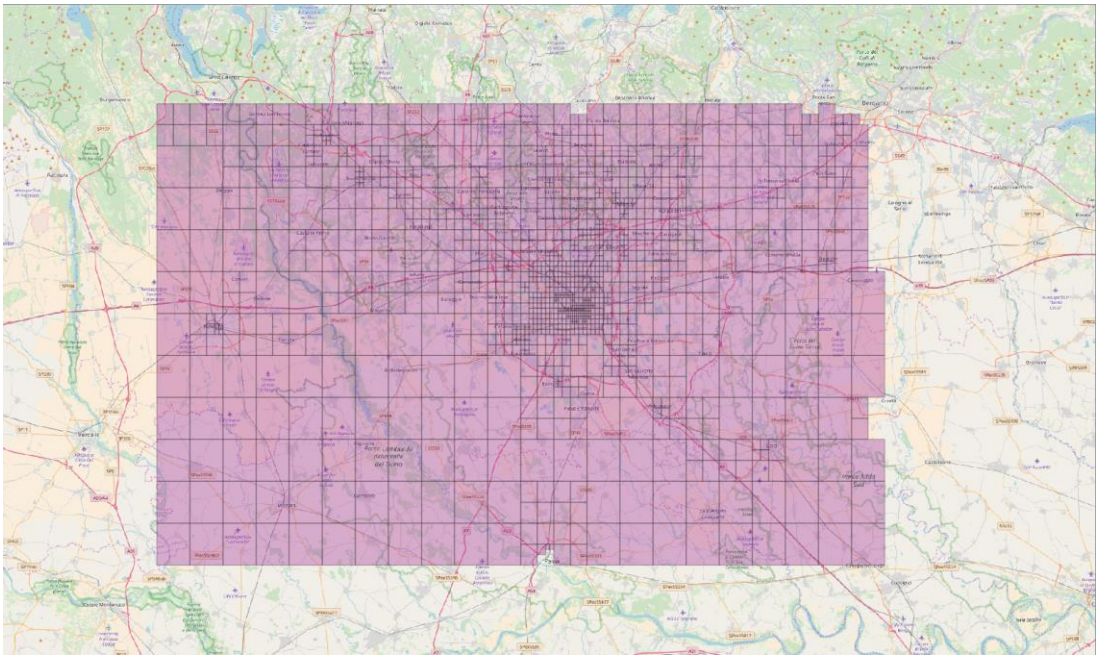


Figure 3 The areas of varying sizes that present the splits of traffic that are taken as Base Stations.

Finally, knowing the geographical area of our base station approximations allows for detailed analysis of the algorithm on top of different types of neighborhoods. Some categorization was done based on human residency such as: business, industrial and residential; while others were mainly based on significant local infrastructure such as: the popular football stadium of San Siro, a very traversed Highway section and the main tourist attraction areas such as the Duomo of Milan.

The timespan of the provided data spans exactly two months, from the first of March to the thirtieth of April in the year of 2015. The data is taken as sums of fifteen-minute-wide timeslots that belong to the underlying area. For the purpose of this research, in order to avoid switching base stations too often, the data is summed to a granularity of sixty minutes by summing four fifteen-minute samples to one hour. On Figure 4 it can be seen a representative case for a certain BS for the whole set of timeslots.

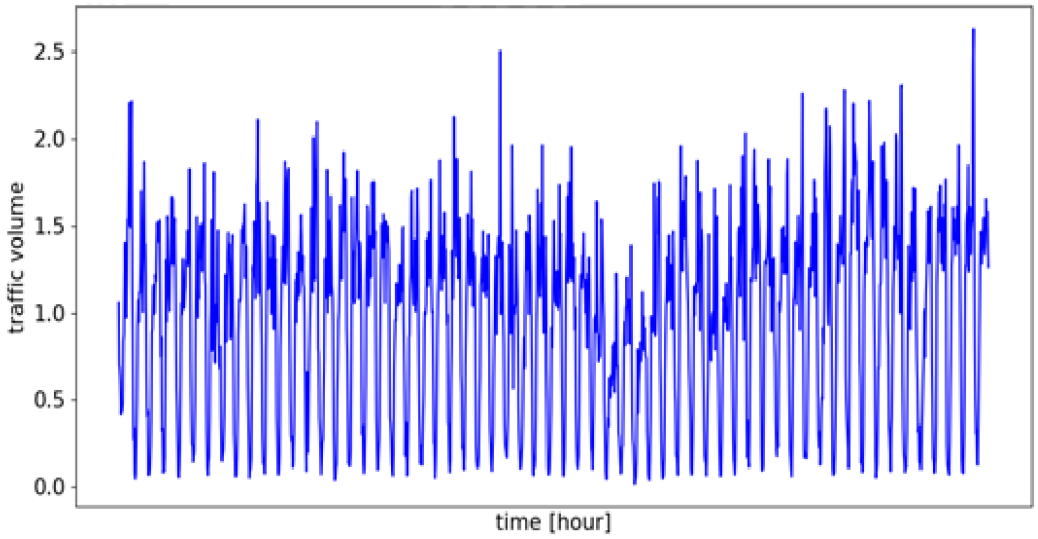


Figure 4 An example of the traffic showing all the 1463 timeslots

In relation to the time series, the data used is particularly interesting for analyzing as it includes two phenomena that are very irregular when considering the normal cycles of human behavior and habits. These phenomena are possibly disruptive for the training of an artificial neural network and may show the robustness of the implementation. The first phenomenon is the decrease in traffic due to the Catholic Easter on the 5th of April 2015 which is a holiday that every year occurs on a different Sunday between 22th March - 25th April not more than one week after a full moon. This makes it a concept that is unrepeatable between years that is hard to relate and a good example of trying to short term estimate a holiday with limited information about its occurrence. The other phenomenon is the Daylight Savings time that skips one hour. This is also a concept that may prove hard for the algorithm to adapt as it shifts the common human habits by one hour in that period. Both these phenomena are good reasons behind using neural networks that may solve the conceptuality behind them without being explicitly told. In the end, the data consisted of 24*61 - 1 =1463 timeslots, where one hour is lost in the daylight savings time shift for all 1419 base stations that consisted of uplink and downlink internet data.

Finally, from all 1463 timeslots, the last fourteen days were separated from the dataset. The resulting two datasets were split in order to create a consistently separated datasets for the purpose of training, the first 1127 hours, and testing the final 336 hours the models.

2.3 Energy Expenditure

To represent the total energy spent of the BSs network the model is assumed to be a simple linear model. Taking the linear model as a safe approximation was due to the contribution of [10] that analyses the many different power elements that influence the energy expenditure: power amplifier, RF transceiver, Baseband processor, DC-DC converter, cooling and AC/DC power supply. The work in [10] concludes that the load dependent power consumption is nearly linear in all scenarios and is a justified approximation. This contributes to simplicity and improves the robustness in applying the algorithm in many different existing and future 5G scenarios.

The model uses approximated costs of idle cost and a variable cost that depends on the type of the Base Station. The fixed energy cost P_0 of elements within one base station depends mainly upon the Baseband processor and other elements which are load independent. Variable cost of the base station $\Delta_p P_{out}$ depends mainly upon the power amplifier element of each transceiver. Finally, the consumption per one base station depends upon the number of transceivers resulting in total power seen at the input from the energy grid as presented in the Equations 2.1-2.2 [10].

$$P_{in} = \begin{cases} N_{TRX} \cdot (P_0 + \Delta_p P_{out}), & 0 < P_{out} \leq P_{max} \\ N_{TRX} \cdot P_{sleep}, & P_{out} = 0 \end{cases} \quad (2.1)$$

$$P_{out} = \rho \cdot P_{max} \ , \qquad 0 \leq \rho \leq 1 \quad (2.2)$$

Where:

- N_{TRX} – is the number of transceivers on the BS
- P_0 – idle power consumption per transceiver
- Δ_p – the slope of the load dependent elements
- P_{out} – the instantaneous transmitting power of the transceiver
- P_{max} – the maximum possible transmitting power of a transceiver
- ρ – the equivalent load normalized from 0-1 by the ratio $\frac{P_{out}}{P_{max}}$
- P_{sleep} = 0 – the amount of energy spent during the sleeping phase

MBSs and SBSs have different values for each parameter making the final linear function per type of BS behave differently. Throughout the whole thesis, the values used for each parameter for Equation-2.1 are shown on Table 1.

BS type:	N_{TRX}	P_{max}	P_0	Δ_p
Macro	*6*	*20*	*130*	*4.7*
Macro-RRH	*6*	*20*	*84*	*2.8*
Micro	*2*	*6.3*	*56*	*2.6*
Pico	*2*	*0.13*	*6.8*	*4*
Femto	*2*	*0.05*	*4.8*	*8*

Table 1 The values used for the Macro-RRH and Micro BS based on State-of-the-art estimations from 2010 [10]

Therefore, from here on, the cases analyzed will use the values from Table 1 as follows. For the Macro base stations, it is assumed that the Remote Radio Head (RRH) implementation is used rather than a boxed one as it offers many benefits and is already widely used when applicable, and for the several Small cells the values will be taken as the cell that is smaller by only one tier, the Micro Base station.

Furthermore, it is usual that P_{sleep} is taken as a value other than zero, that is for cases when fast deactivation of components is supported. Throughout the thesis, it is assumed that such mechanism is unsupported and the only way to implement switching is to turn off the whole BS for more than an hour. Therefore, P_{sleep} is assumed to be negligible or zero for each type of station.

2.4 The Offline Energy Optimization Problem

Taking the energy spending model as mentioned in Equation-2.1 the energy expenditure of the whole system becomes a combination problem. As presented in Equation-2.3 the job of an optimization mechanism is to minimize the total energy expenditure of all stations in the scenario while finding the best combination of micro base station switches τ_i for each Micro BS i.

MINIMIZE:

$$P_{tot} = N_{TRXM} \cdot \left[P_{0M} + \Delta_{\rho M} \cdot P_{MAXM} \cdot \left(\rho_M + \sum_{i=1}^{N_s} \rho_{si}(1 - \tau_i) \right) \right] + \sum_{i=1}^{N_s} \left[\tau_i \cdot N_{TRXS} \left(P_{0S} + \Delta_{\rho S} \cdot P_{MAXS} \cdot \rho_{si} \right) \right] \quad (2.3)$$

S.T:

$$\rho_M + \sum_{i=1}^{N_s} \rho_{si}(1 - \tau_i) \leq 1 \tag{2.4}$$

$$0 \leq \rho_M \leq 1 \tag{2.5}$$

$$0 \leq \rho_{si} \leq 1, \qquad i \in 1,2,3 \dots N_s \tag{2.6}$$

$$\tau_i = 0,1 \qquad i \in 1,2,3 \dots N_s \tag{2.7}$$

But first, to simplify the switching decision, we must understand the behavior in a case where we only have one Macro BS and one Micro BS, $N_s = 1$. In the scenario of only two BSs the problem dumbs down to just ether turning off the micro BS (τ = 0) or keeping it on (τ = 1). Both scenarios, are two-dimensional planes dependent on the slopes of the loads of the Macro BS and the Micro BS. However, since both planes share the same slope of the linearity for the value of the load of the Macro station ρ_M, results in the total power expenditure between the different combinations of τ_i depending only upon the amount of the load of the Micro Station. Assuming that all constraints are satisfied, the crossing value, threshold, between both functions can be found through:

$$P_{\tau=1} = P_{\tau=0} \tag{2.8}$$

$$\begin{aligned} &N_{TRXM}\left[P_{OM} + \Delta_{\rho M} \cdot P_{MAXM} \cdot (\rho_M + \rho_S \cdot O)\right] + 1 \cdot N_{TRXS}\left(P_{OS} + \Delta_{\rho S} \cdot P_{MAXS} \cdot \rho_S\right) \\ &= N_{TRXM}\left[P_{OM} + \Delta_{\rho M} \cdot P_{MAXM} \cdot (\rho_M + \rho_S \cdot 1)\right] + 0 \cdot N_{TRXS}\left(P_{OS} + \Delta_{\rho S} \cdot P_{MAXS} \cdot \rho_S\right) \end{aligned} \tag{2.9}$$

$$N_{TRXS} \cdot P_{OS} + N_{TRXS} \cdot \Delta_{\rho S} \cdot P_{MAXS} \cdot \rho_S = N_{TRXM} \cdot \Delta_{\rho M} \cdot P_{MAXM} \cdot \rho_S \tag{2.10}$$

$$t_{eff} = \rho_S = \frac{N_{TRXS} \cdot P_{OS}}{N_{TRXM} \cdot \Delta_{\rho M} \cdot P_{MAXM} - N_{TRXS} \cdot \Delta_{\rho S} \cdot P_{MAXS}} \tag{2.11}$$

The threshold t_{eff} is where the efficiency of keeping the micro BS ON is determined. Since it is our task to minimize the final power spending by switching micro BSs off we should analyze if such switch-off is feasible, or if $P_{\tau=0} < P_{\tau=1}$. Therefore, we take all the micro BS load values below the threshold ($\rho_S < t_{eff}$) to be more efficiently processed by the Macro station.

In the end, since the load dependence is linear, the optimization problem dumbs down to finding the largest set of micro BSs that have load below the threshold t_{eff} and best fill the unused capacity of the Macro base station. Finally, the value for the threshold, as calculated by Equation-2.11 when substituting with the appropriate values from Table 1 is $t_{eff} = 0.369$. The steps to find the Equations 2.12-2.16 are presented in the appendix.

MAXIMIZE:

$$\sum_{i=1}^{N_s} \tau_i [\rho_{si} - t_{eff}] \tag{2.12}$$

S.T.

$$\rho_M + \sum_{i=1}^{N_s} \rho_{si}(1 - \tau_i) \leq 1 \tag{2.13}$$

$$0 \leq \rho_M \leq 1 \tag{2.14}$$

$$0 \leq \rho_{si} \leq t_{eff}, \quad i \in 1,2,3 \dots N_s \tag{2.15}$$

$$\tau_i = 0,1 \qquad i \in 1,2,3 \dots N_s \tag{2.16}$$

In this model, as all ρ_{si}will be smaller than t_{eff} it is apparent that in order to maximize we would want to turn all micro BSs off ($\tau_i = 0$) that fall under the rule of $0 \leq \rho_{si} \leq t_{eff}$. Furthermore, it shows that if the capacity in the macro BS does not allow for such extreme step, the combination of the turned on micro base stations does not have such great impact as the sum of turned off base stations. In conclusion, this analysis

provides a way to accurately solve the optimization problem in a more efficient way where:

1. The micro BSs with load less than t_{eff} are separated.
2. At first the BSs need to be sorted for their expected load.
3. Starting from the smallest load, turn off BSs until the macro is saturated which results in the maximum number of possible BSs to turn off – N_{off}.
4. By knowing N_{off} the problem for fully accurate calculation reduces to a combination of N_s into N_{off} positions. Where N_{off} is expected to be a much smaller value

Finally, it is worth noting that this approach is not essential for calculating the most energy efficient combination of turned on and off BSs regarding this thesis. This is since this research considered values for N_s which are generally lower than ten, and therefore not requiring much computing power to analyze. Aside that fact, this analysis gives very good insight into future potential solutions to the problems that are much bigger, such as the 5G HetNet paradigm. And as such, it can be used into cellular systems with several tiers of overlapping coverage.

3 – Machine Learning and Artificial Neural Networks

3.1 The Basics of Machine Learning

Machine Learning (ML) is a broad term covering many different algorithmic implementations that improve their performance, overtime, by being provided adequate data. The term, coined by Arthur Lee Samuel [13], is very broad and covers many different techniques that have a machine, which is a mathematical equation consisting of adjustable values, weights. To such machine we are inputting some values, called features, and getting a desired output from the end, all being achieved without explicitly instructing it with the written code. The advantage of these algorithms is that the weights of their internal structure can be modified and improved iteratively through operational research methods such as Gradient Descent or Simulated Annealing. The phase in which we try to adjust the inner weights is commonly referred to as training.

Depending on the type of feeding the data during training, Machine Learning techniques can be divided in the three main categories of supervised, unsupervised and reinforced learning, where:

- Supervised learning is performed by feeding the machine a desired output and expecting it to adjust its internal weights to meet the demand of the output value, while being fed the proper data at input.
- Unsupervised learning is mostly used to segment and understand statistical structures and distributions without specifying the desired output.
- Reinforcement Learning is a type of constantly trained, online, algorithm where it plays within the rules of a game (in the sense of game theory), and automatically improve itself through each cycle by being reinforced or punished for its decisions.

Throughout the thesis, all the developed machines will be trained through supervised learning. Also, the used optimization mechanism was always some form of gradient

descent. Furthermore, when training, batching methods will be always used. Batching refers to splitting the whole available dataset into smaller groups, batches. This technique only helps accelerate the process of training, as group computations are done more efficiently than performing each training step separately. The size of the batch depends only upon the hardware used and memory available. Finally, when passing all training batches once it is considered that one training epoch has passed.

The first step to initiating the training phase of a supervised machine, we need a cost function. This function, evaluates the difference between the desired output of the machine against the one already being outputted. The most common and the only cost function used throughout this thesis is the Mean Square Error (MSE) function, where we only calculate the difference between the desired output and the real one and square it. Using the estimation notation where the value of the desired output is Y_i and the value estimated by the network is $\hat{Y}_i$ for each value in the array of outputs N, the function of MSE takes form:

$$\mathrm{MSE} = \frac{1}{\mathrm{n}} \sum_{i=1}^{\mathrm{N}} \left[Y_i - \hat{Y}_i \right]^2 \tag{3.1}$$

From here on the job of the training is to improve the internal weights of the machine by minimizing the cost function. Once the error is calculated, along with an assigned learning rate value which controls the intensity of the gradient, the algorithm tries to "descend" into the direction where it expects to find a minimal value in the feature-space. Finding the best direction in the variable-space is done by calculating the negative gradient for the function of the output of the machine F(x), which for our cases is the MSE equation. The variables in the MSE are part of the estimated value $\hat{Y}_i = \mathrm{y}(x, w)$ which depends on the internal weights of the machine w. After knowing the amount of correction that needs to be applied $\gamma \nabla F(w_t)$, the machine learning algorithm then needs to adjust its weights per every item within the underlying structure.

$$w_{t+1} = w_t - \gamma \nabla F(w_t) \tag{3.2}$$

Here, γ represents the learning rate that needs to be chosen per each individual implementation, it is usually an arbitrary set value that is smaller than one and bigger than zero. The internal weights adapt per each training step t and hopefully converge towards good results and reduce the error. This process is repeated until stagnating values for the cost are received and it can be noticed that the structure cannot improve itself anymore with the provided dataset. It is worth to note that an important topic when

regarding training called overfitting needs to be addressed and will be answered before presenting the results in Chapter – 4.

Furthermore, we need to explore the possible structures that can be improved through the aforementioned training procedure. Two elementary mathematical structures for machine learning algorithms are the linear regression and the logistic regression algorithms. The linear regression is a very simple case where each of the many input features are assumed to have linear dependency with the single output. Here, each input feature has a corresponding weight as shown on Equation-3.3 as well as one featureless weight that is referred to as bias.

$$y = \beta_0 \cdot 1 + \beta_1 \cdot x_1 + \cdots + \beta_n \cdot x_n + \varepsilon = X^T\beta + \varepsilon \tag{3.3}$$

The output of the sum algorithm is always a real number whose value is unrestricted. This is useful if unconstrained values are to be expected but may induce inefficiencies if otherwise. For some problems, such as classification, trying to train the linear regression to decide inside a very strict set of values may prove unfeasible. Therefore, if the output of the linear regression is passed through a sigmoid function, $f(x) = 1/(1 + e^{-x})$ as in Equation-3.4 the machine will be trained to better classify between the values of zero and one, which are commonly referred to as logits.

$$y = \frac{1}{1 + b^{\beta_0 \cdot 1 + \beta_1 \cdot x_1 + \cdots + \beta_n \cdot x_n}} \tag{3.4}$$

3.2 On Artificial Neural Networks

There are many other structures that are more complicated than the two mentioned. But the most interesting and most promising for the case of BSS switching deem the subset of Artificial Neural Networks (ANNs). ANNs are a try at replicating the functioning of the brain where a single computing node is called a neuron. The principle behind the architecture is that, like in a human brain, a neuron can transfer its information to another neuron through the artificial links as shown on Figure 5. This approach is very powerful in extracting information from seemingly uncorrelated features, an ability for which NNs have drawn a lot of attention in the scientific world [14].

Simply put, an artificial neuron in ANNs is very similar to a singular linear regression node which may be activated by some activation function that may not be necessarily a sigmoid.

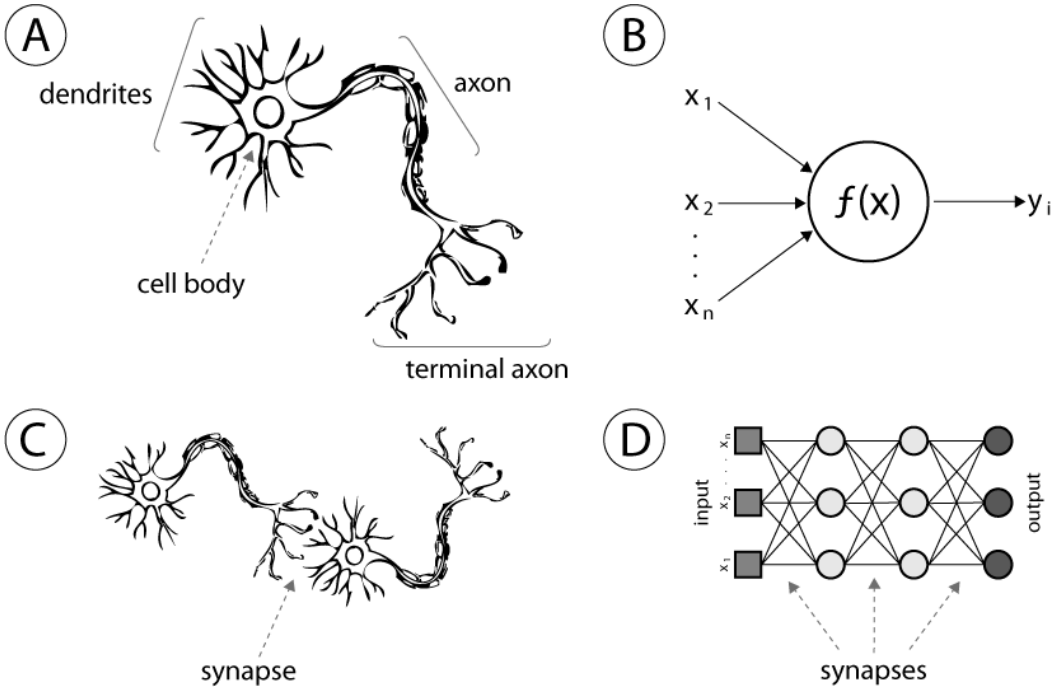


Figure 5 The resemblance of ANNs to biological neurons, image courtesy of [12]

The power of ANNs comes when combining the many neurons that form large abstractions which are often incomprehensible in their problem-solving abilities. This is done by putting several neurons in a layer in order to feed information from the previous layer towards the next one. When drawn, ANNs most resemble classical weighted and directed graph structures and the elements in them are therefore referred to nodes with the links between them edges. Each edge has a real number between zero and one assigned to it, giving weight to the connection between the neurons.

Furthermore, ANNs are directed graphs usually created in a feed forward manner. Which means, the information put on the input of the graph only propagates towards the output and does not revert. Opposed to this, there are networks whose inputs depend on the outputs of previously computed information and are called Recurrent Neural Networks (RNNs). However, even RNNs can only be trained when unwrapped until their starting state is reached, to finally resemble a feed forward network. Essentially, these graph structures create multidimensional matrices (tensors) which represent the weights and can be trained as such. Finally, as it is justified further down the thesis, the aforementioned topologies of Dense Neural Networks (DNNs) and Recurrent Neural Networks (RNNs) are to be analyzed in their performance for finding the best combination of switched base stations.

Finally, the most important feature of any machine learning algorithm are the training periods where the internals of the neural networks are adjusted. The adjustment of the internal weights is not as trivial as in the simple regression algorithms. Therefore, training ANNs is done through an algorithm called back-propagation. Here, the calculated cost at output, evaluated by a cost function fully dependent on the difference between the desired output and the output generated by the network, passes one layer at a time from the one closest to the output to meet the values at the input. Just as the name suggests, as we start from the output and propagate the difference that needs to be corrected at each step of the network, backwards. On a per neuron basis, the gradient descent behaves as in an isolated scenario, as shown on the following Figure.

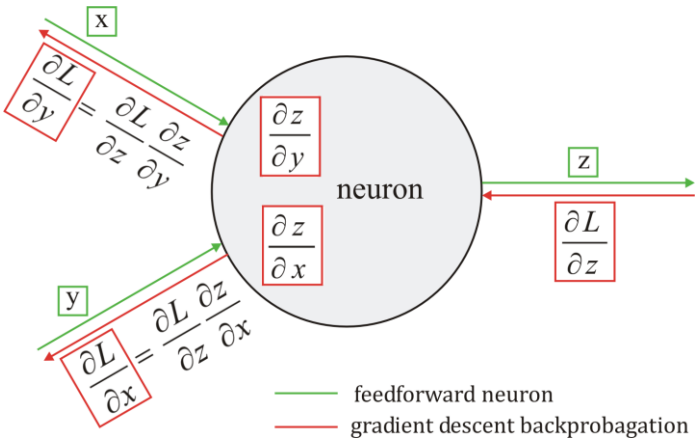


Figure 6 Illustrating gradient descent at each neuron as on an example of two inputs x, y and one output z

3.2.1 Dense Neural Networks (DNNs)

Dense Neural Networks is a common name for an ANNs that are strictly feedforward and have full interconnection between their layers. Contrary, ANNs that are not fully interconnected are therefore called sparse.

In DNNs the processing is split into many layers, usually having the same size, as shown on Figure 7. Here, each layer aside from the first, input layer and the last, output layer, every other layer in the network is dedicated to improving the learning capacity of the network and is called a hidden layer. This is the most common ANN architecture, is very robust and fits many different use cases if features are carefully fed and chosen.

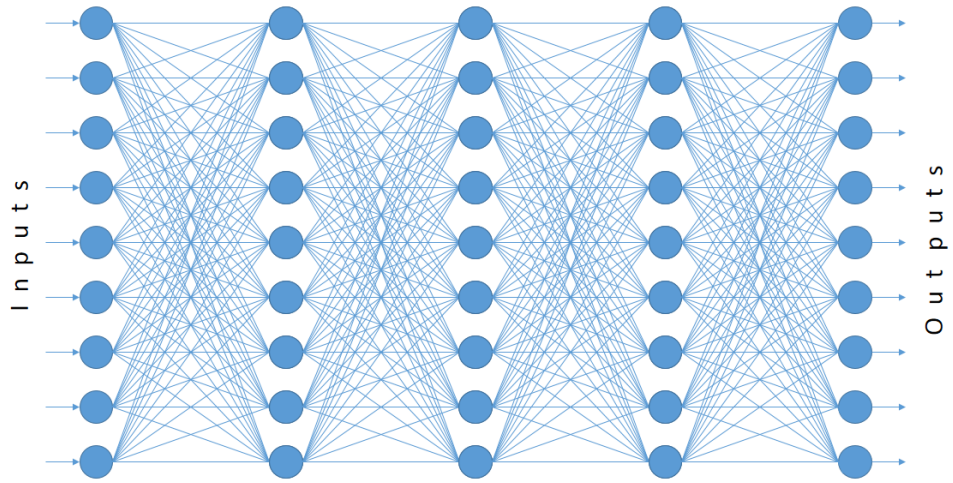


Figure 7 An example of a dense neural network

In more detail, the output of a neuron in the network is the sum of the weights and values of all neurons from the previous layer, which for the first layer is shown in:

$$z_a = g_{1,a}\left(\sum_{j=1}^{N_{in}} w_{a,j} \cdot i_j + b_a\right) \tag{3.5}$$

Where:

- N_{in} is the number of input neurons, features
- z_a is the output of neuron number a of the first layer
- $\sum_{j=1}^{N_{in}} w_{a,j} \cdot i_j$ is the sum of all the weights with the appropriate accompanying input values passing through the previous layer.
- b_a is a bias term that is independent on the network but only applied for each neuron individually.
- $g_{1,a}$ is an activation function used for each neuron in the first layer. It is used as a decider inside the neuron that bonds the output of the neuron to the operating band of values.

The activation function $g_{1,a}$ is important in applying before transmitting the information down the ANN. A node in a ANN must modify its output through the chosen activation

function in order to achieve better convergence of the values while training. Common activation functions are sigmoid, tangent, SoftMax and the rectifier linear unit (RELU).

To further expand a DNN and add another layer, the procedure continues as in the Equation-3.6 for the first layer. Adding a neuron in each additional layer is done by summing all the output values of the previous layer with their accompanying weights. This creates a nested sum. In example, in a DNN with 2 hidden layers, a neuron in the second layer is the following sum:

$$z_b = g_{2,b}\left(\sum_{a=1}^{N} w_{s,a} \cdot z_a + b_b\right) \tag{3.6}$$

This sum-nesting continues until the end of the final output layer which finally results with number of nested sums as layers leading to it, including the hidden layers plus the input features layer.

Finally, in order to train a DNN in a supervised manner, we need to create two parameters that need to be fed. The training input that is a multidimensional vector (tensor) which has values of [*batch_size*number_of_batches, feature_size*] where also the *feature_size* may contain more than one dimension. The other parameter is the training output with size of [*batch_size*number_of_batches, output_size*] again *output_size* may be multidimensional. This means is that per every batch of the *number_of_batches* we instantaneously insert into the back-propagation algorithm a size of *batch_size* instances of *feature_size* at input and *output_size* at output.

3.2.2 Recurrent Neural Networks (RNNs)

In common RNNs the structure of the graph appears to be similar to that in DNNs except for that every neuron depends on its state from the previous processing step. This type of Neural Networks is specialized for solving series problems most commonly used in: speech recognition, text recognition, time series estimation, etc.

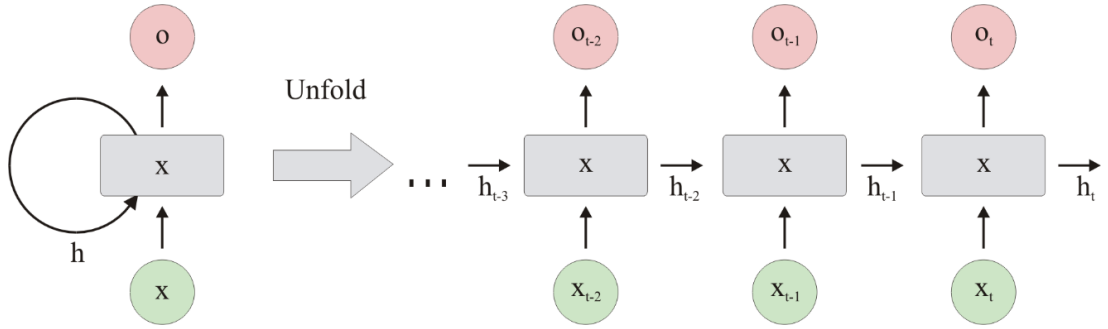


Figure 8 An illustration of unwrapping a Recurrent Neural Network

This makes the RNNs depend not only at the input features, but also at the output state of itself from one timestep earlier as it is shown on Figure 8. Here, the input values X_t are concatenated with the state output of the previous timestep H_{t-1}, resulting in $X = X_t | H_{t-1}$ where again each element of X has an appropriate weight w. Even though this recurrent behavior seems to complicate the situation of the RNNs it can be efficiently addressed.

In order to train the RNNs and make use of the state transitions, the RNNs need to be unwrapped as on Figure 8. Which means that when inputting features, the dimensions of the length of the time series needs to be defined while defining the whole RNN, and before starting the training phase. Therefore, when initializing the size of the RNN, we usually give three important numbers at the training input and training output parameters. The training input multidimensional vector has values of *[batch_size*number_of_batches, sequence_length, feature_size*] where as in DNN the *feature_size* may contain more than one dimension. The other parameter is the training output with size of [*batch_size*number_of_batches, output_sequence_length, output_size*] where the output sequence length must be smaller or equal to *sequence_length*, and again *output_size* may be multidimensional.

Therefore, per every batch of the *number_of_batches* we instantaneously insert into the back-propagation algorithm a size of *batch_size* instances of *sequence_length*, *feature_size* at input and *output_sequence_length*, *output_size* at output. Having these dimensions specified, allows the RNN to be unwrapped for the *sequence_length* number of times for an input of *feature_size*; same applies for the output. This results in a feed forward network that has a depth of *sequence_length* and inputs of *feature_size* at each of the layers within.

After initializing the RNN, each timeslot is inserted one by one through the following steps:

1. Define an initial zero state to be inserted at the last item in the sequence.
2. Insert the last value from the time series into the RNN cell and by computing it with the initial zero state, produce the first output O, and first state V.
3. Create a copy of the previous cell, insert the state value of the previous cell along with the value that is next in the time series.
4. Repeat the previous step until the last item in the time series that give out the last output O and the last state V.

RNNs are complex structures and using them impose many different difficulties. One of which is solving the initial zero state, that makes several initial values in the time series to be inaccurate and harder to train. For this, using a smaller time series is inferior to using a bigger one. However, this leads to the other big drawback to RNNs which is the computing cost to using long timeseries. Sequential items, inherently, are creating the network to look like a normal DNN where each feature appears in a sequential step. This creates long series of cells when unwrapping the network to perform back-propagation in the training phase. This creates very deep networks, in the sense that the ANN contains a lot of hidden layers, which in turn greatly increases the processing power needed. While developing the solutions for our problem, an illustration of a network that has dimensions of *[_ ,50,1]* is shown on Figure 10. In the end, if unlimited computing resources exist it would be immensely powerful to use the full timeseries of training data as one input, but such data must be segmented into smaller sequence sizes.

In addition, when creating RNNs simply adding a feedback edge on a classic DNN neuron, with an activation function would not provide good results as the network will suffer from vanishing gradients when trying to train the network. The problem of vanishing gradients appears due to using gradient-based method for learning combined with backpropagation. When using this approach, on each training step, each edge's weight is corrected to a partial derivative of the error function. Therefore, vanishing gradients appear in very deep networks as the value of the gradient may become smaller the further the output and stop the neural network from training.

To resolve this issue many implementations of RNN, cell architectures were created. Among the most used are Long short-term memory (LSTM) and Gated Recurrent Unit (GRU). These architectures are significant as they employ a forget gate which helps resolve vanishing gradients issue. The LSTM and the GRU both implement the same concept of using a forget gate while having some slightly different internals. However, it is generally accepted that GRU tends to converge to good results faster than an LSTM, while in some cases LSTM use may in the end find a better result. These

mathematical structures are very good substitutions for the regular ANN cell when used in an RNN manner.

The LSTM cell, introduced by Sepp Hochreiter and Jürgen Schmidhuber [15], first showed its promise in natural language text compression. The inner structure of a LSTM cell is as follows on the set of equations. It has three sets of inputs and outputs where:

$$f_t = \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{3.7}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{3.8}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{3.9}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{3.10}$$

$$h_t = o_t \cdot \sigma_h(c_t) \tag{3.11}$$

GRUs are very contemporary alternative to LSTM units that offers similar performance, introduced by Kyunghyun Cho et al. in [16]. It is easy to notice that GRU cells rely on much fewer calculations as in Equations 3.12-3.14.

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{3.12}$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{3.13}$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \sigma_h(W_h x_t + U_h(r_t \cdot h_{t-1}) + b_h) \tag{3.14}$$

In the end, both types of cells were tested to search for better estimation of the traffic load on the infrastructure, showing very similar performance to each other.

3.3 ANN Solutions for BS Switching

As said, the network providers are mostly deterred from using a BSS switching as a solution as it may lead to damages to the equipment. For this reason, when optimizing the overall consumption, it has been taken that a base station (BS) needs to be switched with at least one hour pause after the last switching decision. The developed

switching algorithm can become more appealing to a potential final implementation. Using data from the Metropolitan Milano Area, a dataset of the total traffic per hour per area was created aggregating the bidirectional traffic. Therefore, a proposed algorithm would operate strictly in per hour instants and decide the availability on each as previously desired. In the end, a reasonable size of a network of 1 Macro BS with full overlapping coverage of 6 Micro BSs was considered for consistent testing and result comparison between the different methodologies.

The goal of the solutions is to minimize the total energy expenditure of the network of base stations without compromising the Quality of Service (QoS). However, since it is required to disable the base stations for at least an hour a time it is essential that the solution performs estimation for the traffic of the following hour. It is also important to understand that this type of estimation, where there is no enough data to solve annual problems, is referred to as short term load forecasting. This type of forecasting must be the only one considered since the total available data spans only two months. Therefore, using more than a handful of weeks' history would result in a big shrink in the training data available.

Finally, it can be concluded that the task of the solution can be split into two parts: estimation of the traffic for the following hour and switching decision based on the estimation. ML solutions were tested as part of **only estimation**, **only switching decision** as well as **estimation and switching together as a unified solution**.

3.3.1 ANNs for Estimation

Since working with only 6 Micro BSs, the number of combinations for the case is small and can be very swiftly calculated as in the offline scenario presented in the second chapter. If we treat the system as a two part and leave the combination problem to be solved with a traditional measure from an estimated value, it is only the estimation that is left to be solved by the machine.

ANN load estimation is not only a problem of communications but many load dependent scientific areas, most of which are trying to solve energy demand. As our method uses one-hour timeslots to avoid the very fast switching of BSs, it coincides with the work on short term load estimation of (Lee, 1992). This paper was not only taken as an example for estimation with DNNs but also as a good practice of choosing features for all scenarios of short term load prediction.

3.3.1.1 DNN Estimation

In detail, five features are enough for a decent estimation of the value for the next hour without overbearing the network with information about all hours in between [17]. The 5 features are chosen upon their significance for estimating the load for hour t, and are the past loads at the same BS at times:

feature 1 $t-1$

feature 2 $t-24$

feature 3 $t-1-24$

feature 4 $t-2\cdot 24$

feature 5 $t-1-2\cdot 24$

It suggested to use a four-layer architecture, with the initial feature input layer, two hidden layers between them, and a single output neuron layer for the final estimation of the timeslot t, as shown on Figure 9. The activation functions used, excluding the final output neuron, consist only of sigmoid functions. As for the final output neuron, it is left without an activation function in order to not restrict the working range of the output. To train the model the classical back-propagation with a gradient descent optimizer is used through first calculating the mean square error between the estimation of the network and the desired output.

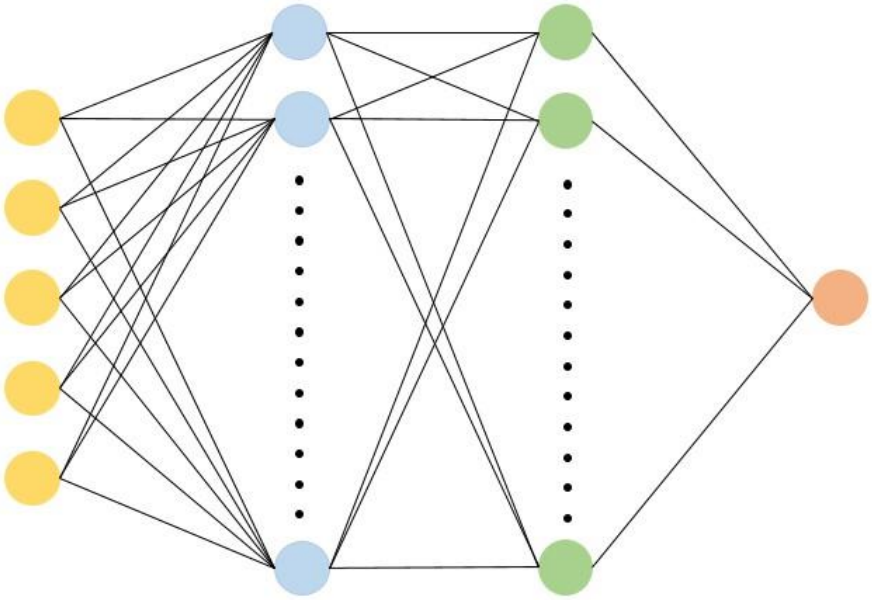


Figure 9 The DNN of [17] hence classic-ANN

Finally, although the model in [17] uses a distinction between the weekday, and weekend periods. This means that there are two different machines used to estimate for the two different groups of data. Though this is feasible for estimating energy demands which behave in a much more stable fashion. For the use of traffic data, such division does promise gains and was considered as one possible flavor of the algorithm in the testing. However, as it is shown in the following chapter that presents the results, it did not immediately result in such gains for our case.

To find out the best approach to the problem many different flavors of the algorithm were developed. The presented set of ANNs are created as a close approximates of the implementation in the work of (Lee,1992), where the main architecture us kept the same as in Figure 9, but only making a change in some aspect of the data on input:

1. have only one machine per Base Station to estimate the traffic of that BS only, naming it ***classic-ANN***. Meaning that every BS should have a machine to estimate its own traffic,
2. have two different machines for the weekday and weekend periods for estimating one BS's traffic. Here, each BS would have two machines swapped during weekday hours and weekends;
3. have only one machine for estimation of all 7 - BSs in the scenario, named ***ANN-new***. This means that there would be one machine per Macro BS that estimates the values for itself and each covered Micro BS with the same machine. The machine is of course trained with the traffic of each 7 base stations, where the data is inserted as 7 different batches of input data.

3.3.1.2 RNN Estimation

As mentioned, RNNs are very complex structures that need much more attention to work properly. To create a viable solution a good start was replicating the feature inputs as in the DNN scenario. It is usual that the size of the input data for entering an RNN is fragmented in three dimensions. These three dimensions are displaying the most important variables when working with RNNs: *[batch_size*number_of_batches*, *sequence_length, feature_size]*

When inputting this data, the first value of *batch_size*number_of_batches* is generally left to be of variable size *"_"*, while the sequence length vs. the feature size are the most important metrics that change the architecture of the RNN. As discussed, the depth of a RNN network depends only upon the size of the *sequence_length* value. Knowing this, as we are working with estimation only, there can be a trade-off between

including some values of past load as features or as a sequence in the time series as well. Therefore, without increasing the depth of the RNN at all, there are several viable flavors of RNNs to be analyzed, the two tested are:

1. A replica of the DNN input features, with the use of LSTM/GRU cells with an input array of *[_, 1, 5]* – ***Simple LSTM Cell***. This means that each BS has a machine that does the estimation for itself. Furthermore, here the recurrent passing of states is not used.
2. An increase in the feature size of the implementation above, here the input array has a size of *[_, 1, 50*], where each of the 50 features follow the rule of Equation-3.15; this implementation was named ***Extended LSTM Cell***. For this implementation, only one machine is trained to estimate the traffic for the whole Milan area, meaning that the data had 1419 different batches at input.

The exact value of 50 taking consequent timeslots was taken, as that results in taking all the values up to the hour that is two days and two hours before the hour up for estimation. This is done since the implementation in [17] analyses the correlation between past and future data to decide the significance of past values in the estimation of future ones. Using a size of features bigger than 50 will shrink the available training data, which with a total 1127 time sequences was a limited resource already.

$$feature\ i = t - i, for\ i = 1,2,3 \dots 50 \tag{3.15}$$

After testing the previous implementations, one last try at RNNs was tested where a more complex state passing is used. As mentioned earlier in this chapter, the initial state which is the input to the very first element in the time sequence is the biggest drawback of the implementations using RNNs. And, as for the implementations 1 and 2 mentioned above, that perform state renewing on each step means that the main benefit of RNNs is not used. As for this third implementation named ***stateful-RNN***, exploits the state passing properties of the RNN.

This implementation, has a size of *[_, 50, 1],* which means that in the input size the time slots were 50 with exactly one feature which is the load at the appropriate time. The RNN architecture was implemented as presented on Figure 10. It was created through two layered LSTM or GRU recurrent NN where the output of the second layer is tied to an output neuron. Note that on Figure 10, each block containing LSTM/GRU cell is of size 200, this is usually referred to as cell size rather than naming it separate 200 cells. Finally, from the first two proposed implementations above, it is apparent that this, third, solution would be a lot more computation heavy as it increases the

depth of the RNN is increased from 1 to 50 to which it is added the complexity of adding one additional layer of LSTM.

Regarding the evolution of the internal states of the neurons, the RNN has an input zero *state_0* and finishes with *state_1*. The zero state is set to flat zeros, while the output state is a non-zero matrix. This *state_1* matrix can be approximated to be a useful state as if there was another entry in the timeseries, needed for the next step of estimation. This implies that such state can be useful for the next-in-line estimation.

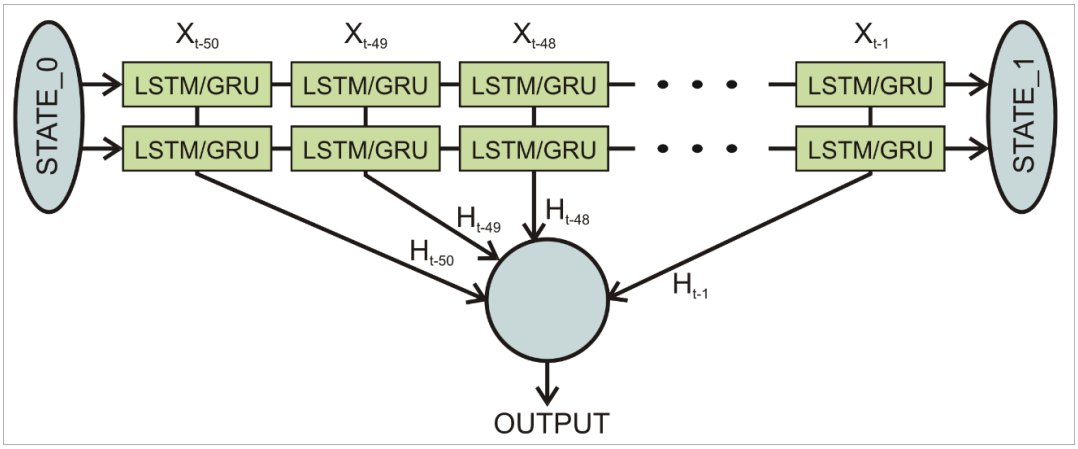


Figure 10 The concept behind a stateful RNN where the states are passed down in the future

However, if left like this, the zero state will be reset each time another value is up for estimation. To battle this drawback a fully tied state passing mechanism can be employed, commonly referred as stateful. Here, data inputting follows very strict rules and each new batch must start where the previous batch ended. Meaning that, at the end of training with batch with position *i*, we save the output state *state_i*. This *state_i* is then used as input for the zero state of the next batch *i+1*. This new batch *i+1* will receive the appropriate features for estimating the next value in line utilizing *state_i* leading to better estimations for the value i+1, and only improve with each passing of the size of the time series. This potentially makes the NN pass down information for an unlimited time window down the ongoing estimation slot. This method is very powerful in passing down information from timeslots that have already been in the network but are not part of the current 50 sequential items. However, the stateful approach cannot carry information about a significant phenomenon if it has not already been recognized as significant (trained) within the band of 50 sequential inputs that are given to the RNN at each time.

Finally, using the stateful approach implies that training and estimating can only be performed one estimation per time and very careful batching must be employed. In the end, the created solution was using the data of all base stations as *batch_size*, and the number of values up for estimation becomes the number of batches, leaving the size of the two other values to be 50,1. This implementation is referred to as the ***stateful-RNN***.

3.3.2 ANN for Offline Switching Optimization

Until now, we have proven that ANNs are capable of Short Term Load Estimation. However, to develop a unified solution, ANNs must be tested for their performance in solving problems that have strict rules such as the Offline Switching Optimization problem mentioned in the second chapter. This test relies to assuming fully correct values from the estimation part of the research. Therefore, to test this, a new DNN was created, and it was not tested with RNNs as no sequences are in the nature of the problem. The DNN would receive on input only the true values of the traffic as features and output the correct combination of micro base stations. The final form of the ANN consisted of: One input layer that has the instantaneous traffic of all 7 BSs, one Macro BS and 6 micro BSs, three fully interconnected hidden layers, and in the end 12 neurons as shown on Figure 11.

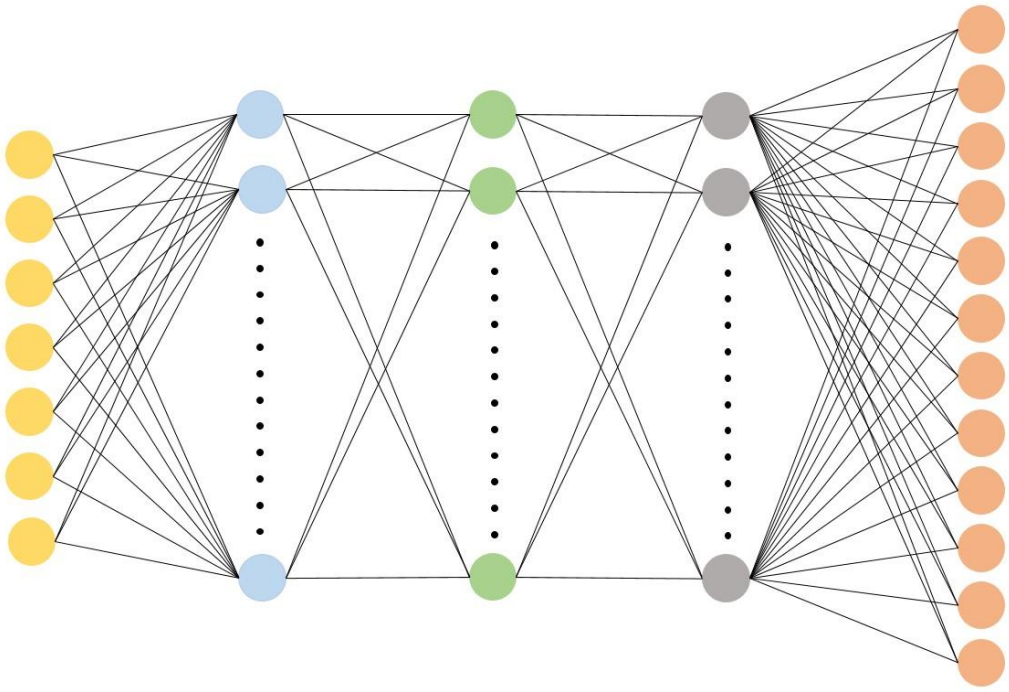


Figure 11 The NN used for solving the optimization problem

The 12 outputs represent the ANN's certainty in deciding as in:

- If the micro BS *i* needs to be kept ON, the i-th of the first six neurons O_i should equal to one and neuron in position *i*+6 O_{i+6} is equal to zero,
- or if the micro BS *i* needs to be turned OFF, the *i*-th neuron O_i needs to have a value zero and the *i*+6 neuron O_{i+6} of one.

This approach of presenting the output state of each micro BS in two neurons is in the interest of simplicity when understanding the networks certainty towards either choice. In more detail, the neural network is trained to output a real number between [0,1] on each output O. This means that the final decision is left to be performed on after the outputted real values. This approach is very useful if an activation function other than sigmoid is used in the final layer where the values in the output neurons may exceed the range of [0,1]. In this way, the decision can be taken by the comparison of the certainty for a switch. In detail, it is taken as the highest probability of the *i*-th and *i*+6-th output neurons. In this case, if the value of neuron *i* is bigger than the value of *i*+6 the micro station *i* is kept on, if otherwise, the micro BS is turned off; as shown in Equation-3.16. This solution does not require any more computing power than while it simplifies decisions and is therefore used across all the unitary solutions as well.

$$mBS_i = \begin{cases} 1, & O_i \geq O_{i+6} \\ 0, & O_i < O_{i+6} \end{cases} \tag{3.16}$$

Finally, an approach that is very significant for enabling this case, a new ***expanded dataset*** was created as a synthesis of the previous data. Since the training and testing of this network can be simply done by synthetically generated data, a more middle of the road solution was performed. This dataset consisted of 100 neighboring sets of 7 base stations from the downtown center of Milan area. Here we increase the dataset even more with the possibility of having each of the 7 BSs as a macro and creating a dataset of 700 scenarios. Since each scenario can be further expanded by swapping the input position of the micro base stations, 15 different combinations were added and resulted in a final number of 700*15 scenarios.

3.3.3 ANNs for Full Decision – Unified Solutions

As it is shown in the results, an ANN has some struggles but can solve the optimization problem nonetheless. This opened the way for creating the unitary solution that does estimation and optimization within its structure. As this is an inherently more complex problem than both previous solutions together, a more careful approach was taken towards both the architecture and data feeding to the Neural Network. For this part a lot of different architectures were tested with varying datasets, layer sizes and depths of the network. The main basis of all the testing was done behind DNN and RNN solutions.

3.3.3.1 The proposed DNN Solutions

At first, a structure similar to that of the estimation DNN network was considered. Where, there are two sets of five features per BS, excluding the Macro BS for which only one set of 5 features was inputted, summing to 65 total features on input. The first set of features per BS were chosen just as in the DNN estimation network, as the loads at timeslots:

feature 1 $t-1$
feature 2 $t-24$
feature 3 $t-1-24$
feature 4 $t-2\cdot 24$
feature 5 $t-1-2\cdot 24$

And the second set of five features included only per micro BS were chosen to be the accompanying best optimal switching decision for each of the timeslots for which we input the load (the times listed above). The size of the internal network consisted of 3 hidden layers with 65 neurons each making a balance between the power of the network and the time to its convergence towards usable results, this solution was named ***Unified-10***, as it relies on 10 features per Micro BS. The final size of the network is 65-65-65-65-12 where the output features are modelled in the same way as in the ANN for offline switching optimization.

A second type of DNN implementation was a simpler one but more computing heavy where on input there are fifty consecutive timeslots up to the one for estimation per each BS, as in Equation-3.15 listed before. This resulted in an input layer of size 350 features, called ***Unified-50*** as it had 50 features per BS. Since the first hidden layer would benefit of having the same number of neurons as the input layer, as the previous implementation shows, leading the NN to understand the abstraction behind predicting the best combination needs no network of such size which also needs a lot of time to

converge towards good results. Therefore, for this design it was decided that for each layer the number of neurons is halved resulting in a design of: 350-350-175-87-12. Again, here the output is in the same format as the ANN for offline switching optimization. This may be a NN much bigger than needed to achieve the required results but can be taken as a no compromise solution taken only for testing.

Both implementations would receive the expanded dataset scenario for training. Which means that the implementations are a solution for each possible scenario for the region of the Milan area. This simplifies the fact that the solution will only be present in a specialized computation area which will synchronously compute all the BS switching scenarios at once with one machine.

3.3.3.2 The proposed RNN Solution

The third type of a unified solution created, was a modified version of the stateful RNN estimation, called ***Unified-RNN***. It was implemented with dimensions of: 50 for the length of the time series as in Equation-3.15, and 7 for the feature size resulting in input size of [_, 50, 7]. Here for each element in the time sequence there are 7 elements because we want to only use the loads of all BSs at each timeslot up to the one for estimation because if we also include the best – past switching decision, the neural network will be overwhelmed with information. The final layout of the RNN is resembled in the following image 12.

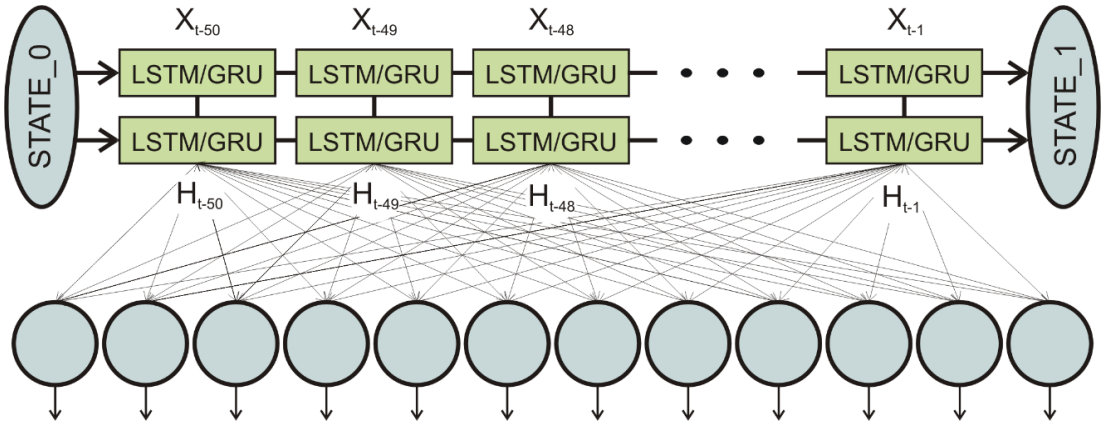


Figure 12 The RNN solution for the unitary RNN

As for all other unified implementations mentioned above, the outputs of this implementation are formatted in the same style as the Offline Switching Optimization neural network. They use 12 features to present the best switching optimization, like

in the optimization only scenario. Also, the same method for creating a big training dataset of 700*15 base stations was used for training a single machine capable of solving optimization for the whole region of Milan.

Finally, the main expected benefit of using unitary solutions is the possibility of scaling the output. Using the following model, if the value of neuron *i* is bigger than the value of *i*+6 the micro station *i* will be kept on, if otherwise, the micro BS will be turned off as shown on Equation-3.16. But here, arbitrary thresholds t for employing a tradeoff between energy savings and QoS can be used as in Equation-3.17. As the outputs of our neural networks are passed through a sigmoid, the output range is constrained to 0-1 and therefore thresholds are used from 0 to 1 with a step of 0.1. This technique shows its benefits in the final results as an optional implementation of unified solutions.

$$mBS_i = \begin{cases} 1, & O_i \geq O_{i+6} - t \\ 0, & O_i < O_{i+6} - t \end{cases} \tag{3.17}$$

4 – Results

Before testing the ANNs listed in chapter 3, some important concepts need to be understood. Firstly, a good portion from the limited dataset should be split for testing only, usually around one quarter of the whole set. This split is very important, and it is what offers reliable information about the performance of the algorithm. When performing estimation of timeseries, as all the employed algorithms do, this split needs to be done in a sequential manner such as the proposed in chapter 2: the first 1127 hours dedicated for training and testing only with the final 336 hours. These timeslots were never mixed and were always kept separate, even if the traffic of a specific BS is used for training the only the first 1127 hours of it are used.

Another thing imperative in understanding the results of the ANNs is knowing the drawbacks of the dataset and the point of approach to the problem. To do this, the first thing needed to understand is the definition of and how to address overfitting. This is further accompanied by investigating how it occurs and what measures can be taken to tackle the problem.

4.1 Overfitting Analysis on ANNs

Overfitting is “The production of an analysis which corresponds too closely or exactly to a particular set of data and may therefore fail to fit additional data or predict future observations reliably” [18]. A usual example for overfitting that is easy to portray is on the linear regression model with one feature as shown on Figure 13. In Figure 13 we can clearly see that the trained function strictly follows the targets which it was trained to estimate, without paying any attention to the abstraction behind it, which is a simple linear behavior. This phenomenon is also often referred in literature as noise fitting. In more complex structures, such as the neural networks, overfitting mainly occurs due to two reasons: too many neurons in the network or a small training dataset.

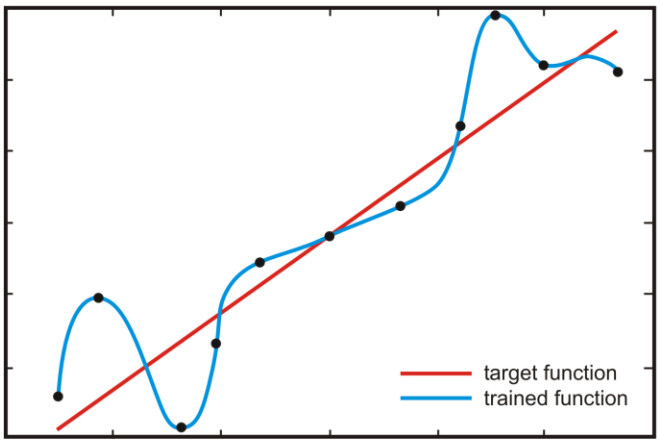


Figure 13 An illustrative example of a function that overfits the training dataset

However, to diagnose overfitting in more complex systems such as ANNs, another way of visualizing is required during the training phase of the machine. To be able to detect overfitting, it is very crucial to follow the evolvement of the network's mean square error with the test data as input. There are other performance metrics, other than the mean square error, that may be more important in some case studies. This encompasses metrics like accuracy of rounded logits, but for our implementation, the mean square error is sufficient. Ideally, while training the network, when feeding for both the training and testing dataset, the mean square error should decrease equally for both, that is, if normalized with the batch size. Therefore, undesired behavior is analyzed through following the mean square error for each passing epoch. Here, passing an epoch refers to passing once through the whole training dataset or all the available batches. Overfitting in this way becomes obvious as the calculations of the mean square error when fed with the training set versus the testing set start to diverge as presented on Figure 14.

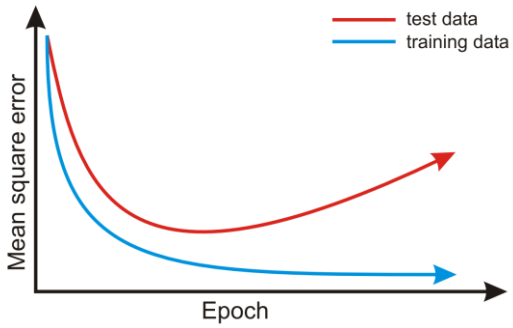


Figure 14 An example of calculating the mean square error over the training data (blue) and over the testing data (red) to spot overfitting.

If the ANN overfits due to the big number of neurons in its structure, it needs to be scaled down. However, sometimes implementations may have too many neurons but if fewer neurons are used the performance of the algorithm drops significantly. This would require a painstaking procedure of constantly retrying different layer and depth sizes until overfitting is eliminated. A tool to counter this issue, is a method called dropout [19]. Dropout reduces overfitting due to the number of neurons by “killing” a percentage of neurons chosen randomly per training step and reintroduces them after that step. This slows down the performance for some scenarios but is worth the versatility that offers to the implementations. Additionally, finding an adequate dropout percentage is essential. Choosing a very big value for the percentage of dropped neurons may lead to bad or no convergence at all. While a very small percentage, may not fill out the potential of the method for the case. This tool is also very crucial when solving very hard problems such as the unitary solutions that must be good at both estimation and optimization. Throughout this research, the dropout rate was set between 25% and 10% for the whole ANN.

4.2 ANNs for Optimization

As it is said in chapter 2, the dataset although does not span a long period of time, only two months, it contains a lot of BSs approximations, 1419 to be exact. And for simplicity in replicating the results in every analysis, the number of BSs under one macro is 6. As highly congested areas are of the biggest interest, such as downtown Milan, there are 11 different scenarios taken within the Milano urban area that are of the biggest interest concerning this research. In Figure 15 the location of each of the 11 areas are drawn on top of an “Open Street Map” (OSM) topological map to better illustrate those areas of interest. It can be noticed that for the analysis many different combinations of sizes for the cells were used each in different scenario and geographical importance. These are the main scenarios used for testing.

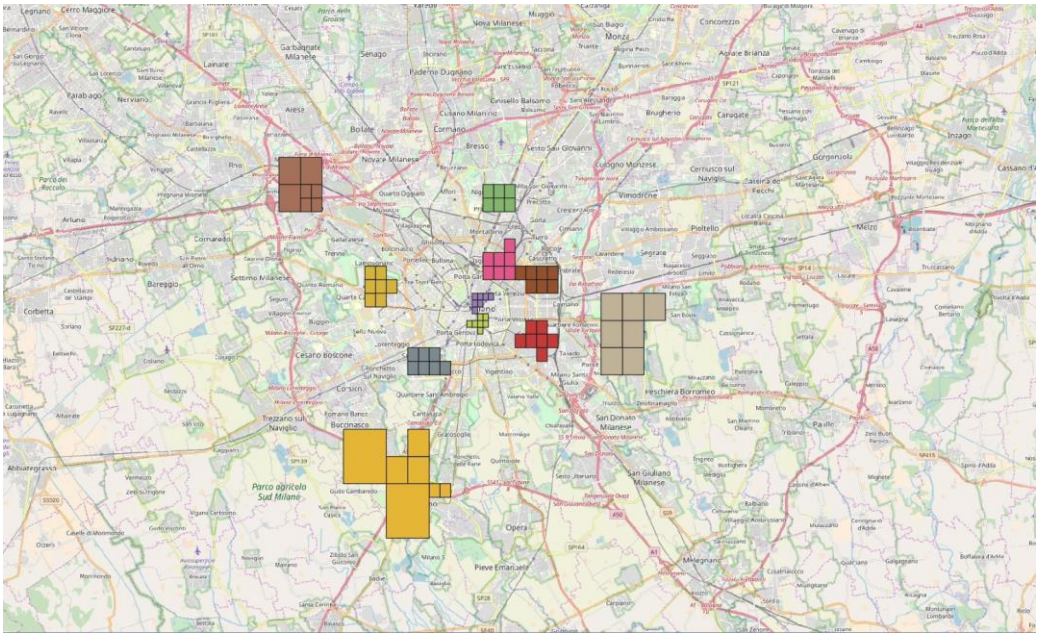


Figure 15 The locations of all 11 scenarios analyzed on top of the map of Milan

The 11 scenarios are taken as a good representation of possible situations that the algorithm needs to face regarding important landmarks and that differ in residential activity. The 11 scenarios are named***: Business*** on ***Figure*** 16***, Duomo*** on Figure 17***, FS*** on Figure 18***, Highway*** on Figure 19***, Industrial*** on Figure 20***, Linate*** on Figure 21***, Polimi*** on Figure 22***, Residential*** on Figure 23***, Residential – 2*** on Figure 24***, Rho*** on Figure 25 and ***SanSiro*** on Figure 26***.***

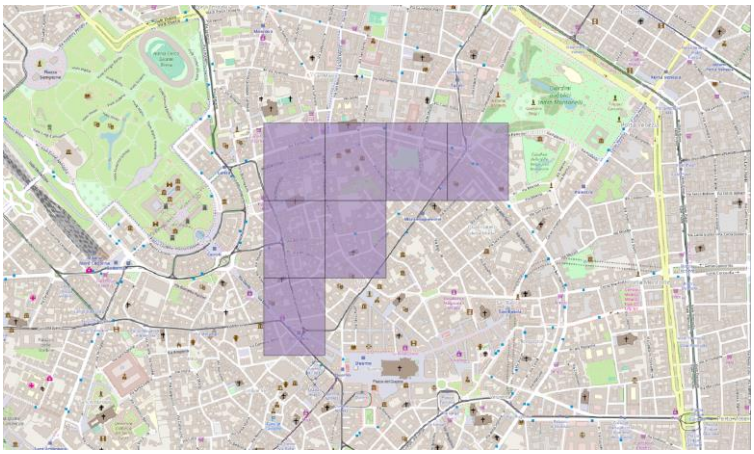


*Figure 16 The main **Business** area of Milan located in the city center*

*Figure 17 The city center of Milan including the **Duomo** and many touristic attractions*

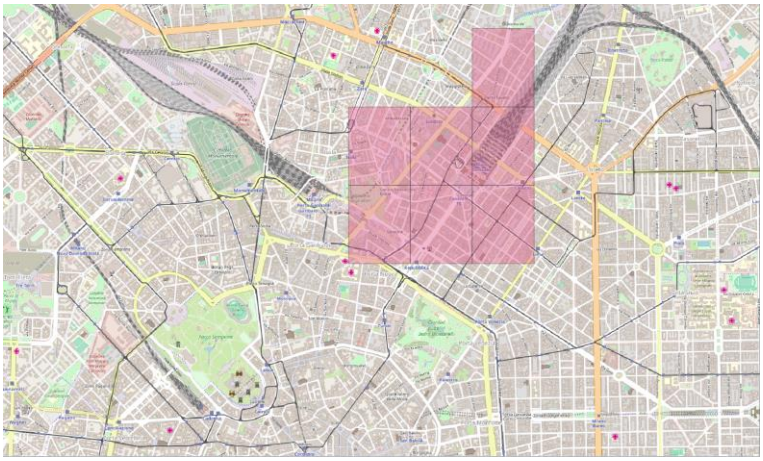


*Figure 18 The area covering the main train station in Milan "Milano Centrale" referred to as **FS***

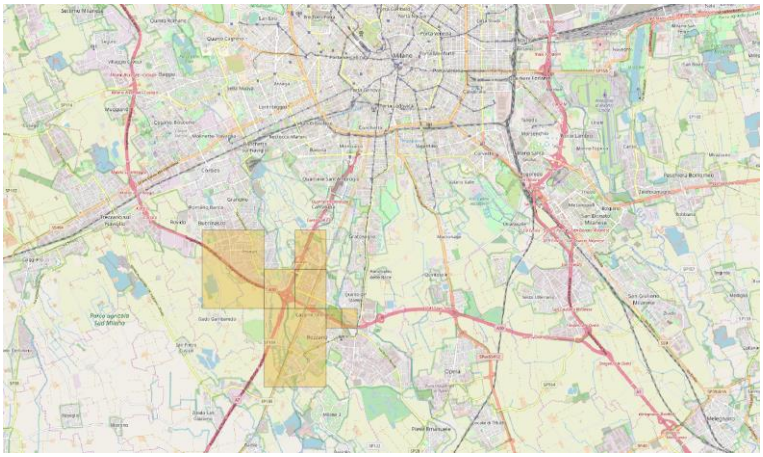


*Figure 19 The one of the busiest **Highway** sections*

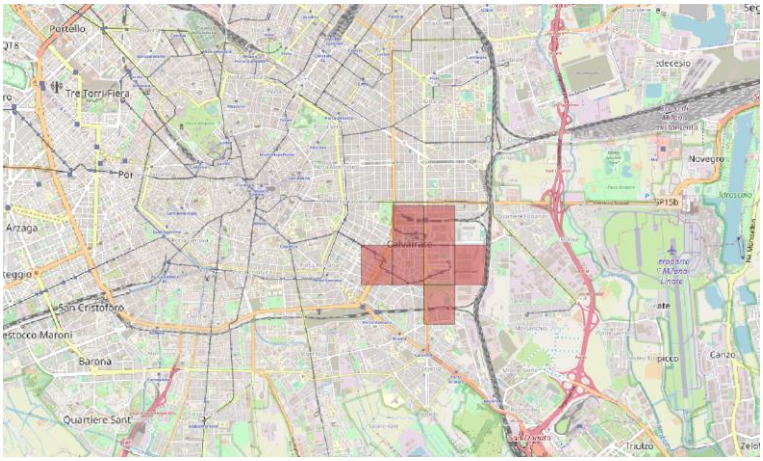


*Figure 20 The main **Industrial** region of Milan*

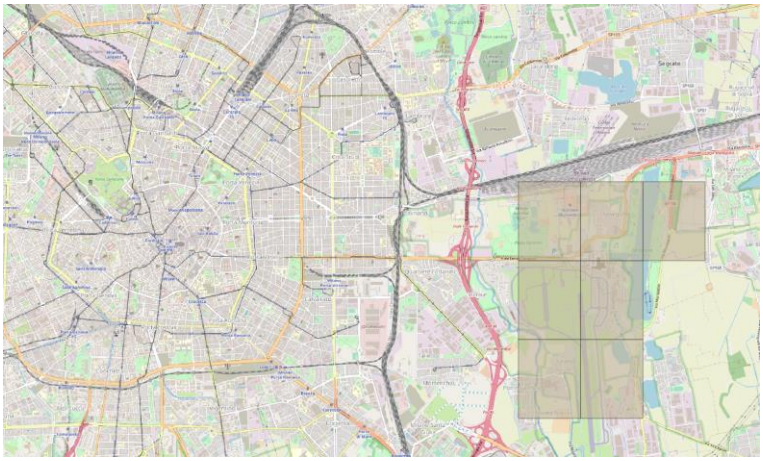


*Figure 21 The airport **"Linate"***

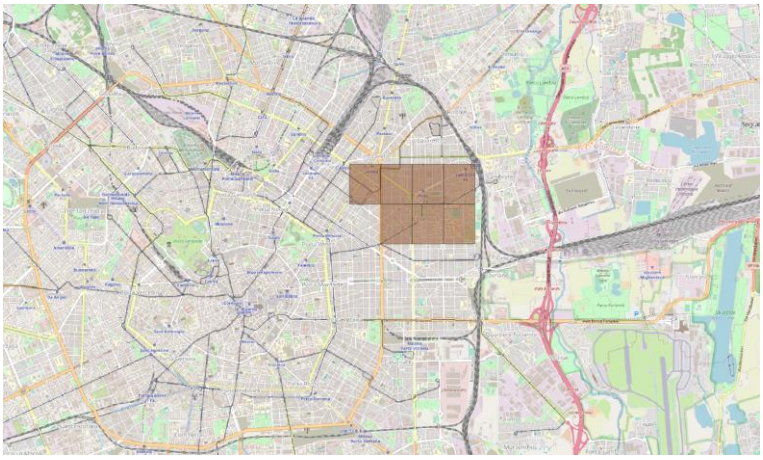


*Figure 22 The region around the "Politecnico di Milano" – **Polimi***

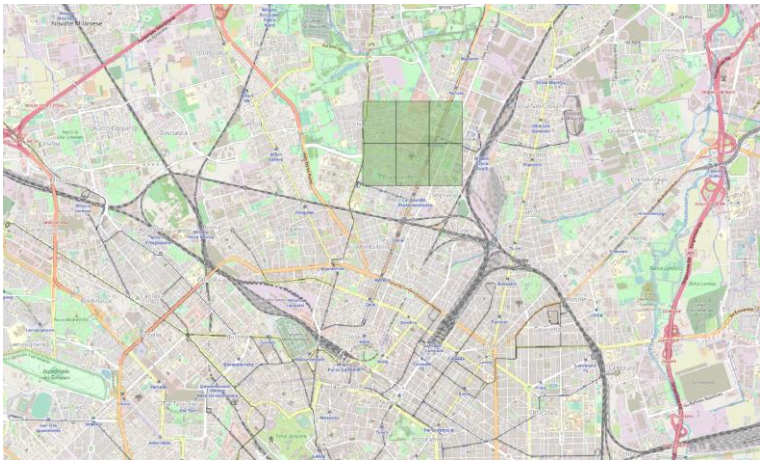


*Figure 23 A **Residential** area in the northern part of the Milan metropolitan area*

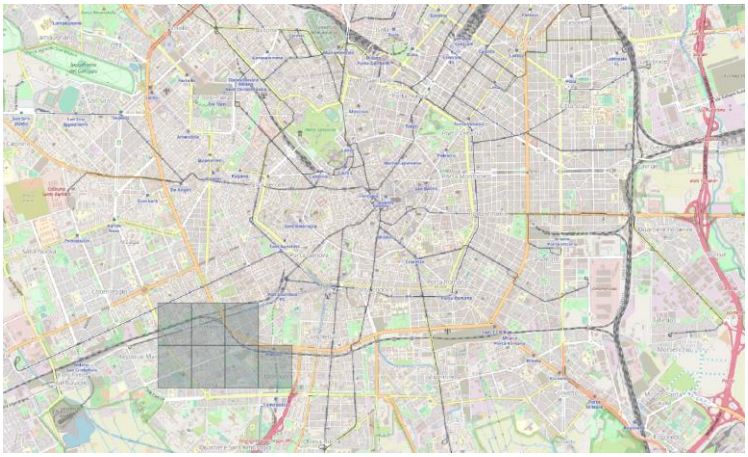


*Figure 24 A **Residential -2** (second) area in the southern part of the metropolitan area*

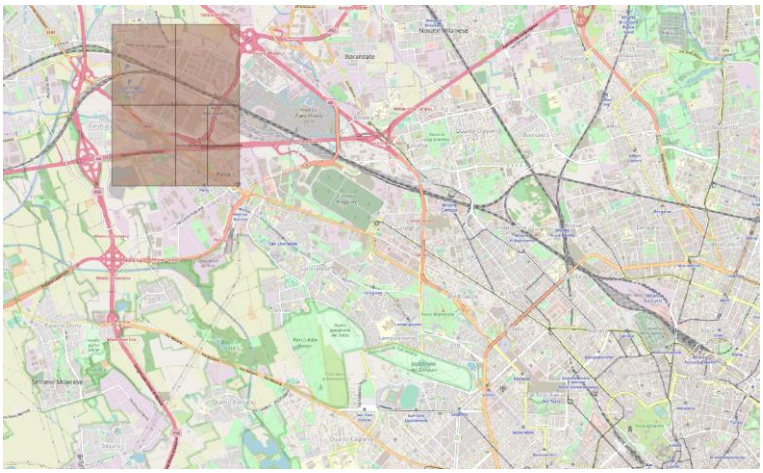


*Figure 25 The area around the second most important train station in the municipality of **Rho***

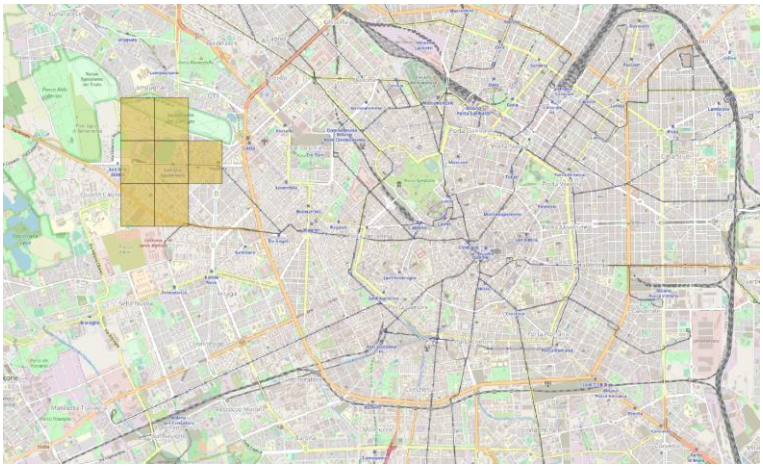


*Figure 26 The area around the football stadium of **SanSiro***

The analysis for the whole set of BSs is reduced to these 11 scenarios to show the potential of the algorithms in a more comprehensive manner. The important metrics of the analysis are loss of *coverage loss* C_L and *loss of energy efficiency* E_L taken across the 336 hours, as follows.

If a switching decision has been taken, and the traffic demand exceeds that of the load that is serviceable with the BSs that are kept on, a loss of coverage, or unfulfilled QoS occurs. This ratio of these two values represents the served traffic opposed to the potential traffic:

$$C_L = \frac{C_{total} - C_{served}}{C_{total}} \cdot 100 \quad (4.1)$$

The measurement of the energy efficiency requires a more thorough analysis in order to not be misleading. However, as we use the energy expenditure model of [10] to fit for our BS abstractions, it might be misleading to use flat values for the potential algorithm. This is because we do not really know the real placement of BSs and their real energy efficiency. But, as we have the calculations of [10] after assuming the role of an area as a BS we can have a flat value for the potential savings of the scenario. As a consequence of that, the values for energy savings are calculated as a percentage of the total potential savings for that scenario. In more detail, the loss of energy efficiency E_L for a tested scenario:

$$E_L = \frac{E_{served} - E_{ideal}}{E_{real} - E_{ideal}} \cdot 100 \quad (4.2)$$

Where:

- E_{real} – the power spent in the original scenario when no switching is applied
- E_{served} – the power spent when switching is applied with an ANN
- E_{ideal} – the power spent when switching is done ideally knowing the instantaneous traffic, this is the minimal possible energy spending without compromising QoS

The result of this equation are numbers in the band of 0 – 100, for all scenarios that don't impact the QoS. There is a possibility of exceeding 0% loss of savings in the cases where have a certain penalty on the QoS. This is done since using the values of E_{real} and E_{ideal} for describing the performance might be misleading as real scenarios will vary vastly in size.

In conclusion to the approach above, the performance measurements for each scenario show that our main goal is keeping both values C_L and E_L to their minimal values of 0%. This approach is very well normalized also towards analyzing the tradeoff between both values. It is important to note that many other metrics might be important to this analysis, of which all are dedicated to understanding the impact of energy saving onto QoS. For this reason, in some scenarios, a more detailed analysis is shown. However, to make a general performance discrepancy between approaches, the two values C_L and E_L are most commonly used.

4.2.1 Estimating with ANNs

In this part we test the performance of the two-part solutions where the ANN machine is used only for traffic estimation. Here, the best combination of switched off station is optimized in a traditional manner as shown in Chapter-2.4. All the following scenarios are trying to achieve maximal energy efficiency by allowing the network to base decisions on the estimation as if we had known the instantaneous. This resulted in very good energy efficiency performance which was very often at the cost of coverage. For this reason, we omit the significance of energy efficiency loss and better performance means coverage loss that is closer to zero.

4.2.1.1 Analyzing the Classic ANN scenario

At first, when testing the developed model [17], throughout this thesis named ***classic ANN***, it was hard to detect any overfitting as the learning rate for the gradient descent was too low to reach a minimum. However, there was a problem that the learning rate could not be increased as the main output of the network did not go through an activation function to regulate the band output. This results in the network needing to undergo several steps to normalize the weights of the last layer to relate to the normalized numbers that better represent the values we want to forecast. For those few steps while the network normalizes the weights the Mean Square Error (MSE) reaches values that are abnormal and can be avoided by passing the output through an activation function or allowing a low learning rate for the first few steps and later increase it. After successfully addressing the convergence issue it was obvious that overfitting occurs early in the training, due to lack of data. This was an important information to understand that using only 1177 timeslots for training was not feasible. However, even regarding this information, if we stop the training before it starts overfitting, it provides descent results. This two-layered ***classic ANN*** scenario resulted in an average across all 11 scenarios:

$$E_L = 0.82, \qquad C_L = 1.17$$

Also, in the relevant work of (Lee, 1992) separates the estimation in two machines one for the weekends and one for the weekdays. Here, unfortunately the data is split and resulted into even worse overfitting due to data and did not bring any improvements. In the end, as averaged across all 11 scenarios the result was nearly the same as in the ***classic ANN***

$$E_L = 0.55, \qquad C_L = 1.19$$

Therefore, resulting in nearly exact performance they are taken as equivalent solutions where using one Machine per BS is considered more feasible approach and therefore is taken as a reasonably better solution considering the size of our dataset.

4.2.1.2 Introducing an improved ANN-new solution

In search of solution for the overfitting problem in the first scenario, the training dataset needed extension. To answer this, a single machine for the whole selected scenario, Network of one macro and six micro base stations, was trained. This managed to improve the performance of the algorithm. But as the overfitting due to the amount of data was partly solved, the neural network started to decrease in performance due to its size, which was adequately addressed by reducing the number of neurons per layer.

Finally, the usefulness of other activation functions was tested against the classic sigmoid, and the Rectifying Linear Unit (RELU) [20] conducted very slightly better estimations, and improved the convergence time of the ANN.

The final ANN structure, named ***ANN--new***, was slightly improving the implementation by decreasing the average coverage loss (as calculated from all 11 network scenarios) from 1.19 to 1.03, and increasing the value of the maximal coverage loss of a scenario; from 1.82 that occurs at scenario Rho in ***classic-ANN*** to 1.39 that occurs at scenario Linate in the ***ANN-new***. All with a reasonable increase in the network power consumption.

$$E_L = 1.58, \qquad C_L = 1.03$$

Note that although the difference of 1.19 and 1.03 coverage loss may seem small but is nonetheless significant. This is because 0.15 percent difference of coverage translates into a decrease in the probability to lose potential traffic of 0.0015 which in communications is hardly admissible. Therefore, the ***ANN-new*** solution is chosen as the best performing DNN implementation for estimation and is analyzed in detail in Chapter-4.3.

The results of the aforementioned implementations lead to a loose conclusion that the training data is insufficient for a comfortable training of an ANN for each individual BS estimation. There is a clear pattern that if the dataset is expanded on BSs that have correlation in the load, such as the ones within one scenario, improved results were achieved. And if the training data is cut down such as in the case of splitting it for weekend and weekday sections, the coverage is therefore reduced. As a matter of fact, from this point on, all the implementations will have their training dataset increased as using only the single BS's traffic did not give any worthwhile results.

4.2.1.3 Testing LSTM cells

From this point on, the RNN for estimation solutions were tested proposed in Chapter-3.3., starting with the ***simple LSTM Cell*** scenario. Where, as presented in Chapter-3.3.1.2, a replica of the DNN with the use of LSTM cells which creates an input array of *[_, 1, 5]*. This scenario resulted in unsurprisingly poor result, as it does not exploit the recurrent-behavior of the RNN. As averaged on all 11 scenarios, it achieved:

$$E_L = 0.56, \qquad C_L = 1.43$$

To continue and test if the average coverage can be improved, an increase in the feature size of the implementation above was tested. This implementation, named ***extended LSTM cell***, still does not exploit the recurrent behavior of the RNN but is

useful to test the importance of the dataset and its size. Here, the input array has a size of *[_, 1, 50]* and only one machine is trained to estimate the traffic for the whole Milan area. In the end the result for ***extended LSTM cell*** improved significantly upon the ***simple LSTM cell*** implementation and resulted in averages of:

$$E_L = 1.05, \qquad C_L = 0.99$$

4.2.1.4 Testing the stateful LSTM

However, as noted, the aforementioned implementations of RNNs do not exploit the potential of the neural network to use information from previous states. Implementing such network proved a very serious feat and is computing heavy, as RNNs are inherent to be as deep as the time series that we try to predict. In the end, the created network accepted data in the form *[Number_of_sequential_batches*size_of_batch, Length_of_Sequence, Number_of_features]* and in the testing runs were, *[336,50,1*], since batch size is 1 when testing (only one BS) and has 336 timeslots, which expands the depth of the ANN to 50, as it is shown on Figure 10. As presented in Chapter-3.3.1.2, an advanced data inputting in RNNs was implemented by inserting the output state of each training step in the following batch's zero state. This led to change the batching of the features to be inserted in sequential times and exploit the recurrent behavior in RNNs. All the previous, in combination with using the data of all base stations, improved the performance of the implementation to 0.84 coverage loss while maintaining very good energy spending as in:

$$E_L = 1.24, \qquad C_L = 0.84$$

It is worth to note that this same machine is ready to estimate the traffic of any type of BS from the whole Milano Metropolitan area. Which means that the whole 1419 BSs dataset was used for training, regardless for which BS we want to estimate the traffic for.

Finally, swapping the LSTM cell for a GRU cell didn't impact the result in a significant manner. This was so far the most successful solution in relation to other estimating machines, named ***stateful LSTM***, regarding providing more reliable network coverage at no cost to energy efficiency.

To better compare all the implementations for the estimation based neural networks, the following Energy versus Coverage graph on Figure 27 shows the coverage percentage (Y - axis) as provided by having values of energy efficiency (X – axis).

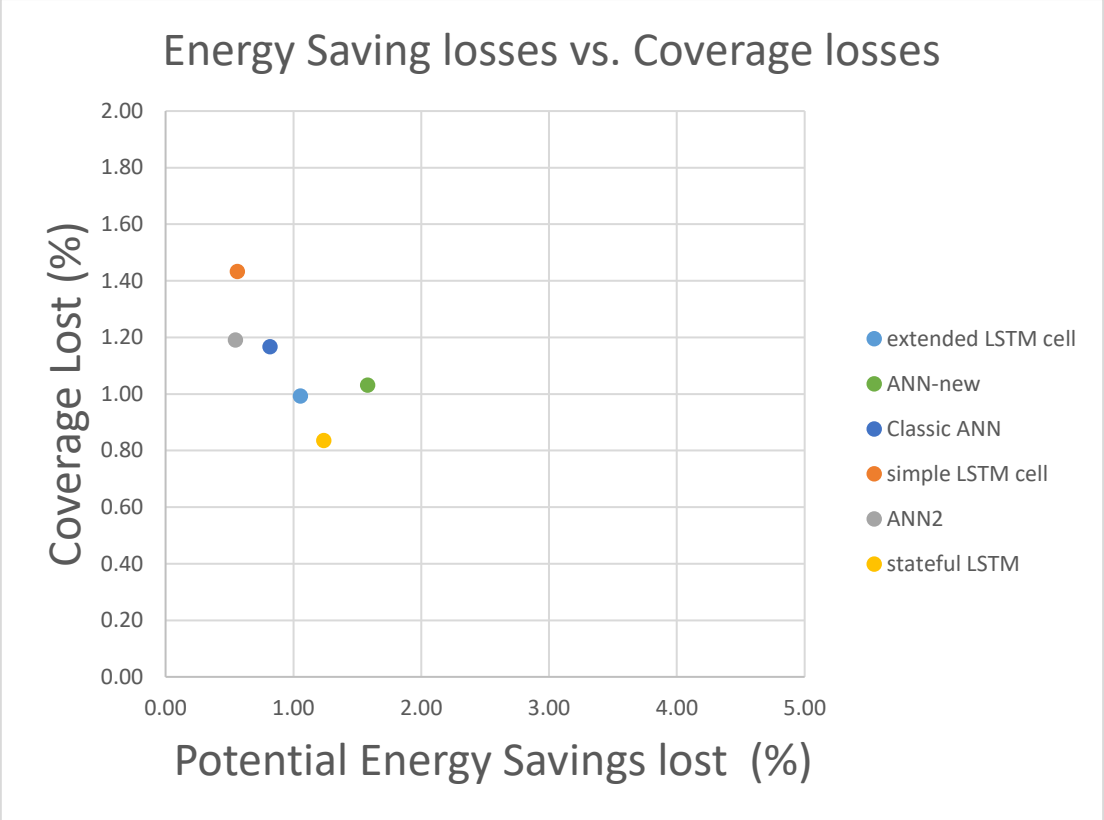


Figure 27 An Energy versus Coverage graph for easier comparison

4.2.2 Troubleshooting the Offline Optimization ANN:

In this part, the process of testing for an ANN that performs only the Offline Optimization only is presented. This of course does not help directly towards our goal of base station switching as no estimation is done. However, analysis of this case helps greatly understand the capacity of an ANN solution to solve the problem of finding the perfect combination of turned off BSs.

Initially, trying to solve the problem, an issue appeared when the neural network of chapter 3.3.2 is expected to only solve the best energy saving combination of on-off BSs. At the time, the average performance for the NN in finding the best combination of turned off/on BSs given the exact values for traffic (no forecasting) was around 98.5%, when trained for only solving the 11 scenarios mentioned at the beginning of this chapter. This accuracy was incredibly poor concerning that this test had the real instantaneous data on input. Although not immediately obvious, when analyzing the difference in performance in training versus testing data, showed only weak signs of overfitting that deemed interesting. Trying to solve the weak overfitting, dropout was implemented with very different percentages of neurons eliminated during training.

This offered no gains in the final results, and therefore, the next hypothesis to test was if the phenomenon was occurring due to lack of data.

To understand if this was true, the big dataset mentioned in chapter 3.3.2 was created. It consisted of 100 BS network scenarios, from it was expanded to 7 different scenarios where each base station in the list is the Macro BS. In the end all 711 (including the initial 11 base station scenarios) were expanded to a combination of 15 different positionings of the feature in the input array. In the end, the dataset expanded from the initial 11 cases to 711*15 different scenarios. When fed during training with this data, the network successfully solved the problem resulting in 99.98% accuracy (accurately guessing the perfect combination) for the training data and 99.8% for testing data. As mentioned, this way of combining data was implemented in the unified solutions as well. When using the ANN to evaluate its performance on the 11 scenarios, resulted in one or two errors for the whole testing duration of 336 hours that had barely noticeable performance impact. This was a confirmation that with the aforementioned procedure of creating a massive dataset, creating a unified solution was possible.

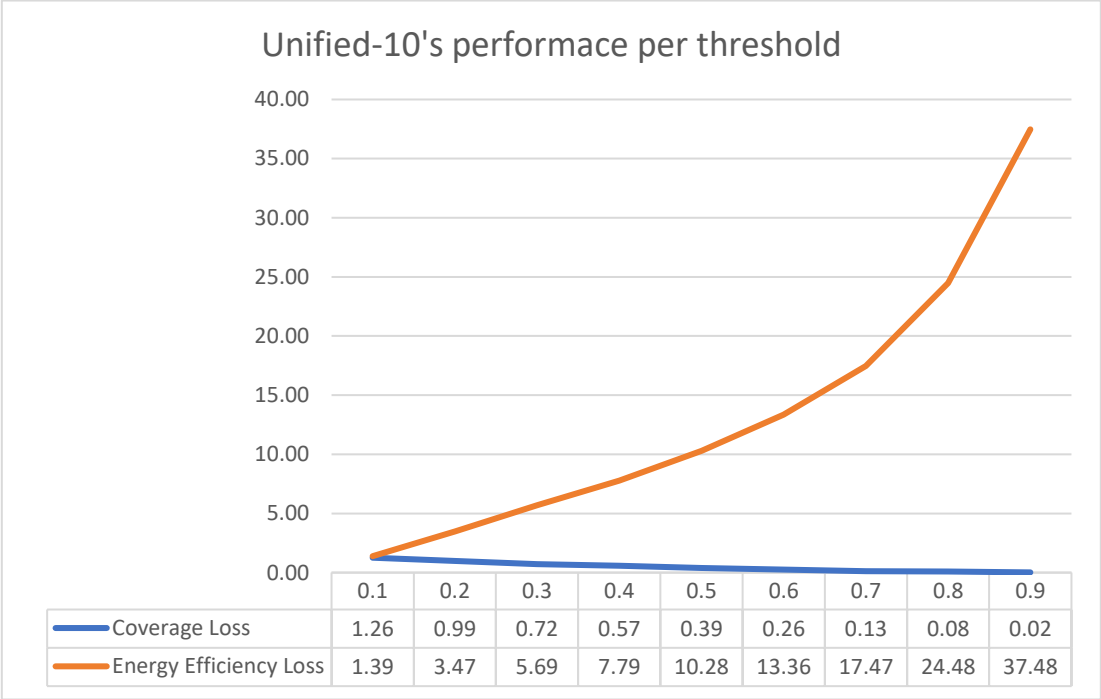
4.2.3 Unified solutions with ANNs

4.2.3.1 DNNs as a unified solution, testing Unified-10 and Unififed-50

A unified solution is when the ANN does both the estimation and switching decision within the network. In the first solution, named ***Unified-10***, the input features were: 10 features per Micro Base Station, 5 features of the traffic and the in best state of on/off BSs for the aforementioned 5 slots resulting in 65 features for the whole set. In the second solution, named ***Unified-50***, the input features are 50 consecutive timeslots up to the estimated hour. The entire description of both implementations is present in part 3.3.3.1. However, worthy reminders are that both ANNs mentioned are DNNs that: have three hidden layers, all neurons are activated by sigmoid, trained by the expanded dataset introduced in chapter 3.3.2, can solve any scenario in the Milan metropolitan area, and most importantly the output can be put through a threshold as in Equation-3.17 to be more switching aversive.

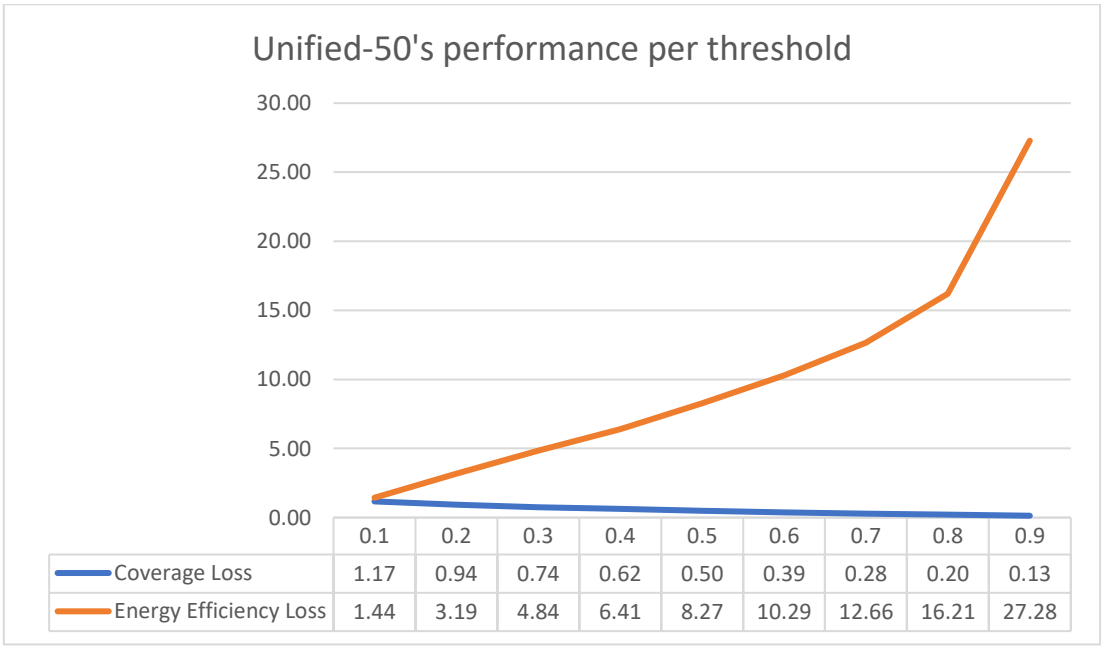
When tested for energy efficiency maximization both networks showed very similar results to each when averaged for all 11 BSs in analysis. These results seem to be equal to those of the two-part solution where a DNN does only the estimation. However, the benefit of this model is scaling the greedy-ness of the algorithm without

having to declare arbitrary time periods of high and low variance. It provides increases in coverage for marginally increase in energy savings as shown in the following graphs on Figure 28 and Figure 29.



	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Coverage Loss	1.26	0.99	0.72	0.57	0.39	0.26	0.13	0.08	0.02
Energy Efficiency Loss	1.39	3.47	5.69	7.79	10.28	13.36	17.47	24.48	37.48

Figure 28 The Unified-10's Energy versus coverage evolution per change in the threshold value upon which is decided in favor of keeping ON the micro BS



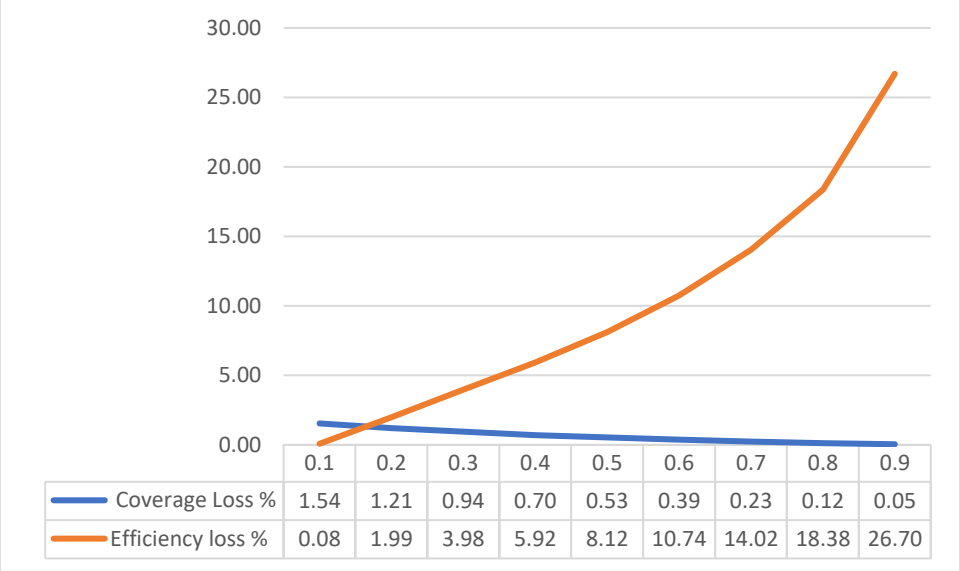
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Coverage Loss	1.17	0.94	0.74	0.62	0.50	0.39	0.28	0.20	0.13
Energy Efficiency Loss	1.44	3.19	4.84	6.41	8.27	10.29	12.66	16.21	27.28

Figure 29 The Unified-50's Energy versus coverage evolution per change in the threshold value upon which is decided in favor of keeping ON the micro BS

The thresholds set were arbitrary and may not resemble the exact certainty of the network for both cases. However, from the results shown, the Unified-10 does scale better when increasing the threshold. The threshold increments of 0.1 were chosen just in an arbitrary fashion of splitting the possible space of [0.1] into ten parts. From both Figures it can be inherited that by setting the lowest aversion towards switching off a micro BS to 0.1 the algorithms provide similar performance of the two-step DNN implementations. However, the possibility of scaling the output goes in favor of the Unified-10 solution. Here, setting a very strict threshold of 0.9 results in only 0.02% of the traffic to be lost. In other words, the probability of QoS decline is 0.0002 which is very usable knowing that the system still manages to fill 62.52% of the total energy saving potential. This translates into, if the service provider has a scenario where 100W can be saved ideally, it will manage to save 62.5 W

4.2.3.2 Testing the Unified-RNN

As in the order presented in the chapter 3, an RNN version of the unified solution was performed called ***Unified-RNN***. Exploiting the expanded dataset of chapter 3.3.2, and the stateful structure of the RNN as shown on Figure 12 it conducted the results of Figure 30 when adjusting the threshold for the switch-off sensitivity.



	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Coverage Loss %	1.54	1.21	0.94	0.70	0.53	0.39	0.23	0.12	0.05
Efficiency loss %	0.08	1.99	3.98	5.92	8.12	10.74	14.02	18.38	26.70

Figure 30 The Unified-RNN's Energy versus coverage evolution per change in the threshold value upon which is decided in favor of keeping ON the micro BS

Adding the three unified solutions to the Energy versus Coverage graph results in Figure 31 where we can see that the unified solutions all perform nearly the same to

each other, with very insignificant differences. For now, it is important to notice that using a unified solution is much more flexible than having a two-step solution because of the exploitation of the final certainty of the ANN by implementing thresholding.

Each of the unified solutions presented are able to solve the energy problem for every possible scenario in the Milan area. This, rationalizes the extra training time required for all these scenarios, as they can be used to fully solve the problem more efficiently on a massive scale. Additionally, the unified solutions are put under the scope for their high coverage guarantee which can be achieved by establishing an arbitrary threshold. As it can be noted from Figure 30, the best high coverage guaranteeing algorithm that provides very good energy savings of around 73.3% is the ***unified-RNN*** with output threshold set to 0.9. For this energy efficiency the algorithm manages to not exceed 0.05% coverage lost.

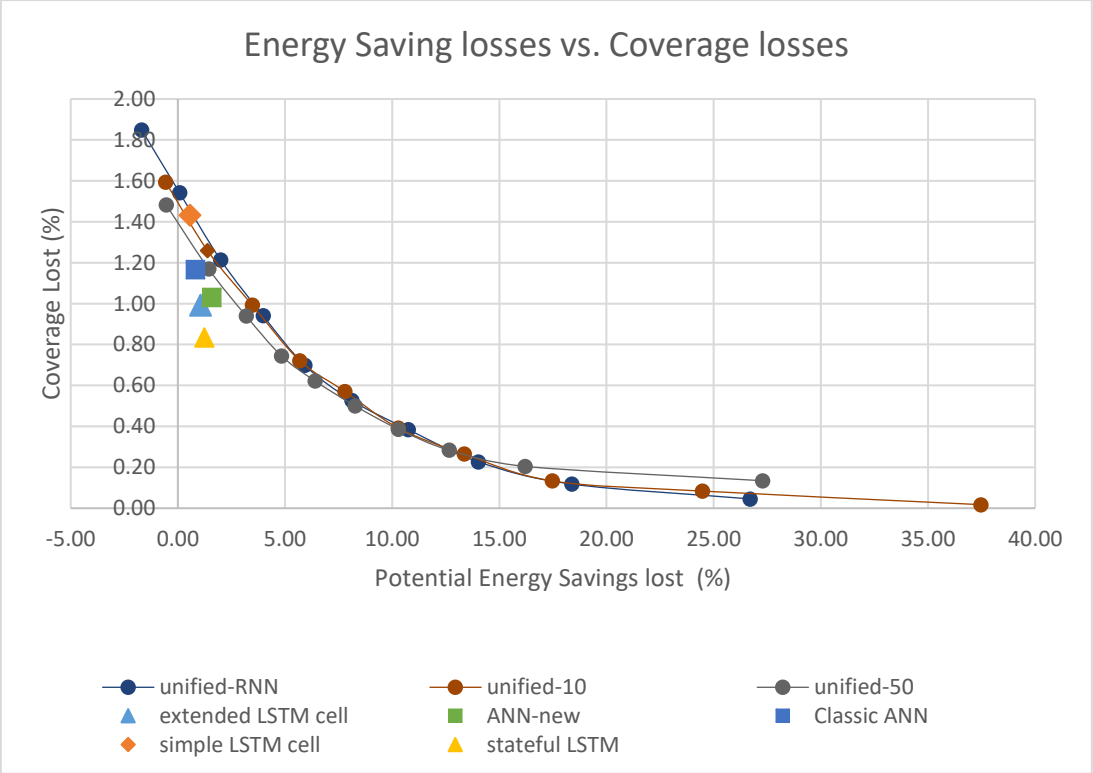


Figure 31 The Energy versus Coverage graph comparing all the implementations tested

A final note of the unified solutions as presented on Figure 31 is that their performance is very identical to each other. This is an odd result as all three unified implementations have different internal structures, which is even more applicable in the case of unified-RNN. This may be a show of the inability of the dataset to enable the performance of stronger estimation mechanisms when also tasked to perform optimization.

4.3 Detailed analysis

Looking into each of the 11 scenarios in detail and analyzing the best and worst results is important to understanding the applicability of each implementation into a real-life scenario. As throughout the whole thesis the goal was to maximize the energy efficiency of the network, it must be strongly noted that it must be done with minimal to no impact on the Quality of Service. Therefore, the averages and percentages of the coverage lost need to be taken very seriously, in that, the algorithm should never obstruct the normal operation of the cellular network. Considering this, the only case that provides 100% certainty that all calls will be guaranteed is an asymptotic scenario where there is infinite capacity. Therefore, it is reasonable to assume that by careful planning by the service provider small QoS losses are allowed for some areas. As consequence, this implementation can be taken as a tradeoff where increasing Energy Efficiency is at the expense of increasing the probability of decreased QoS. Considering this, solutions that have 1.5-1 percent loss of coverage are considered to induce inadequate service with probability of 0.01 per connection, which is unacceptable. Considering this, going back to the Energy versus Capacity graph in Figure 31, the only solutions that support such strict demand are the unified solutions where the decision boundary is set to support only certain switch-offs. In the continuation of this analysis the following successful scenarios are considered:

- ANN-new
- Stateful -LSTM
- Unified-10
- Unified-RNN

The following choice was made as they are the best performing algorithms of their class where each of the classes considered were: ANN estimation, RNN estimation, ANN – unified and RNN – unified, respectively to their order in the list. In more detail, the ANN-new can be analyzed as a base scenario where the efficiency of the approach can be shown to excel or fail. The *Stateful – LSTM* is the implementation that achieves the best coverage loss from all the estimation-based machines.

4.3.1 ANN-new

The ANN-new is an estimation machine that is able to estimate the traffic for all BS of a single scenario. Therefore, per each of the 11 scenarios there is a machine that is trained to estimate the traffic for all 7 of the underlying BSs.

When trained, ANN-new in more specifics resulted in averages for each of the 11 scenarios as shown in Table 2. From the table, we can see that the area with best coverage is the scenario over the main train station of Milan, referred to as FS. Even for the FS scenario the coverage loss achieved is 0.62% which is unacceptable for this algorithm to be considered as a viable solution. However, even aside the underperformance of this implementation, analyzing the best and worst performing scenarios will give us insight into understanding the results of all other viable implementations.

With our way of presenting the energy efficiency, getting negative values for the energy loss is easy as long as the coverage loss is above zero. Therefore, the EL= 0% is not an asymptote of the system but must be regarded as overly optimistic approach of the algorithm. The difficult task is to come as close as possible to the asymptote of 0% coverage loss. And for this, the whole performance onto this scenario is observed only through the coverage.

ANN-new		
	Efficiency Loss %	Coverage Loss %
Business	0.94	1.00
Duomo	0.56	1.09
FS	3.66	0.62
Highway	2.33	1.00
Industrial	0.51	1.17
Linate	1.07	1.39
Polimi	2.71	0.72
Residential	3.44	1.12
Residential2	2.54	0.90
Rho	0.41	1.25
SanSiro	-0.78	1.08

Table 2 Coverage and Energy Efficiency values per each scenario for the implementation of ANN-new

ANN-new, is the worst performing implementation from all the implementations that are analyzed under details, but as such, it still pinpoints the shortcomings of the three worst performing scenarios for all solutions. The scenarios of Rho, Linate and SanSiro constantly show up as the worst performing scenarios of the implementations, and for this their ideal performance must be analyzed in detail. Considering the underperformance of the ANN-new implementation

Rho is a community spanning an area around a train station that is very popular for commuters that go to Milano and back. The ideal and the existing power consumption for the station is given on the graph on Figure 32. As the power consumption is a good representative of the activity of the base stations, having equal or worse than the ideal

power consumption is preferable. Adequately, to show the potential of the energy savings in a proposed scenario we use the power consumption across time graph, as on Figure 32 for scenario Rho.

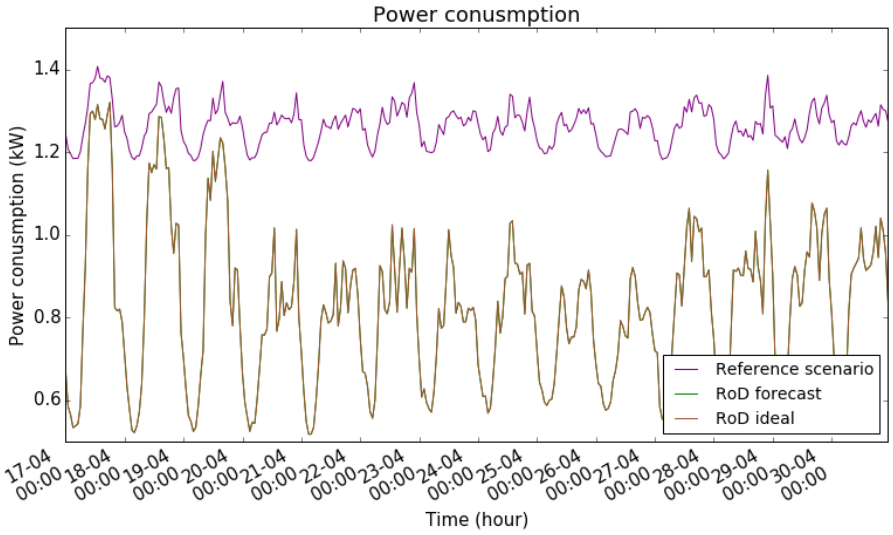


Figure 32 The ideal power consumption for the reference scenario of Rho

Here, it can be seen why the machine has a difficulty predicting the ideal power consumptions for scenario Rho. The peaks of load are very different across the days and the variance during the active hours becomes extremely volatile to be easily predicted.

As for the other bad performing scenario, the SanSiro stadium, the ideal and the reference power consumption behaves as in Figure 33.

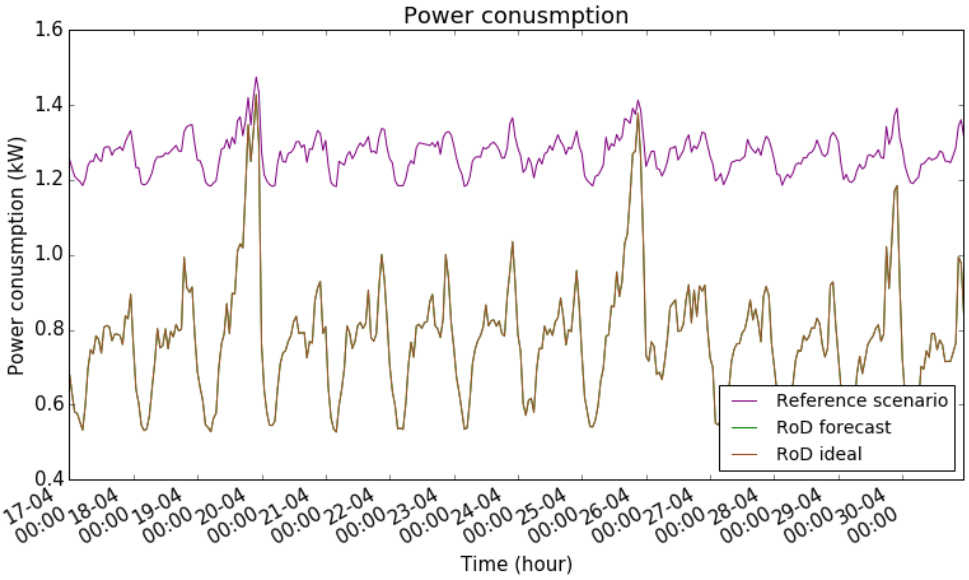


Figure 33 The ideal power consumption for the reference scenario of SanSIro

Here, it can be easily noted when there is an event on the stadium that the demand for coverage even peaks to the maximal serviceable power demand. It is common, in such cases, for the service provider to already control the micro/Pico stations that are on the stadium by already knowing in advance if an event occurs. This is also a reason for a motivation to implement a possible additional feature for such stations. Such feature may consist of the timing of an extraordinary event and probably it's character. This will help predict spikes for the areas of interest such as in the scenario of San Siro.

The third worst performing scenario is the one of Linate. As in the power consumption graph 34.

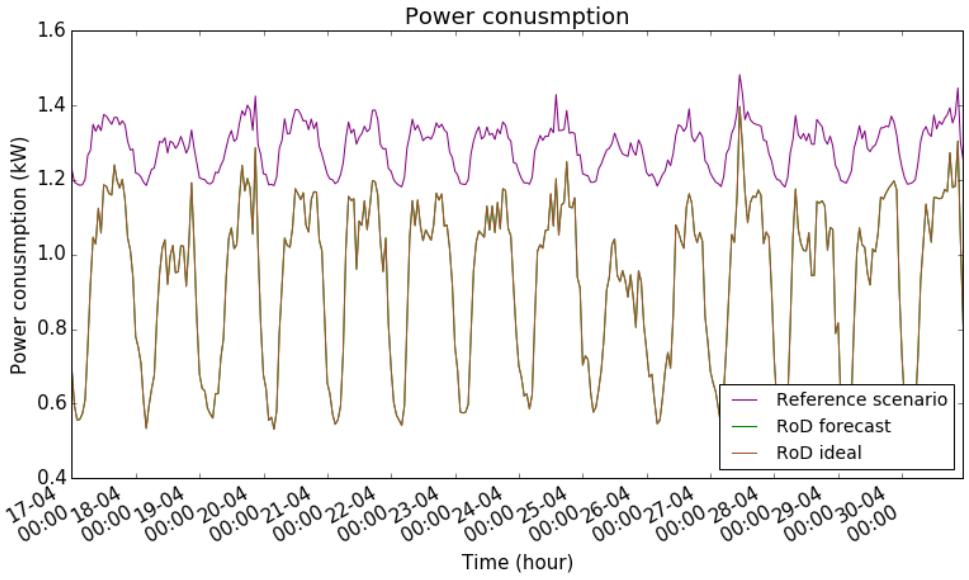


Figure 34 The ideal power consumption for the reference scenario of Linate

In this scenario, the variance of the high activity timings is very high, and it is understandable that the estimation of the traffic does not yield very good results. It is evident that since the area within and around the airport that is closest to the city center is very erratic and unpredictable. It is very active while there are flights, and the surrounding area working hours have started, but it is even interesting to notice that the lowest consumption during the nights is not very stable as well. This means that the inactivity during the night hours is not equivalent to each previous day. As we have looked upon the worst and best performing areas, we will delve into the more feasible solutions and understand how to avoid the bad QoS periods.

4.3.2 Stateful - LSTM

In Table 3 we also again care about the coverage percentage as our only performance metric when estimating the traffic. This also applies for the Stateful – LSTM scenario, since, in all the applicable scenarios the energy efficiency loss is well below 5%. This is the most interesting implementation to analyze since it achieves the best Coverage versus Energy Efficiency ratio.

stateful LSTM		
	Efficiency Loss %	Coverage Loss %
Business	0.71	0.72
Duomo	0.54	0.78
FS	2.08	0.47
Highway	1.64	0.83
Industrial	0.75	0.70
Linate	1.68	1.18
Polimi	1.41	0.65
Residential	3.15	0.82
Residential2	2.64	0.85
Rho	-0.18	0.96
SanSiro	-0.81	1.23

Table 3 Coverage and Energy Efficiency values per each scenario for the implementation of ANN-new

As in the previous scenario of ANN-new, the three worst performing scenarios are SanSiro, Linate and Rho.

The worst performing is the scenario SanSiro, the algorithm fails for the cases where there is an obvious spike which is not estimated neither well nor exactly on time, as shown on Figures 35, 36. Here, it can be assumed that the RNN approach did not have enough data to discriminate the periods in which there might be a football game on the stadium. This assumption is viable as the other ANN implementations manage to somehow battle this problem by being much simpler and not requiring a lot of history to discriminate such events. However, regarding the normal behavior of this scenario combined with all the other ones, the RNN shows great potential when sufficient timeslot history is provided. This is evident since this machine was trained on all available BSs and is able to estimate the traffic of each BS when that BS's traffic is the input of the machine. This means, that since other BS's which have highly regular traffic regarding each other, the scenarios that do not show any extravagant behavior such as FS, are extremely well estimated by the unified – RNN.

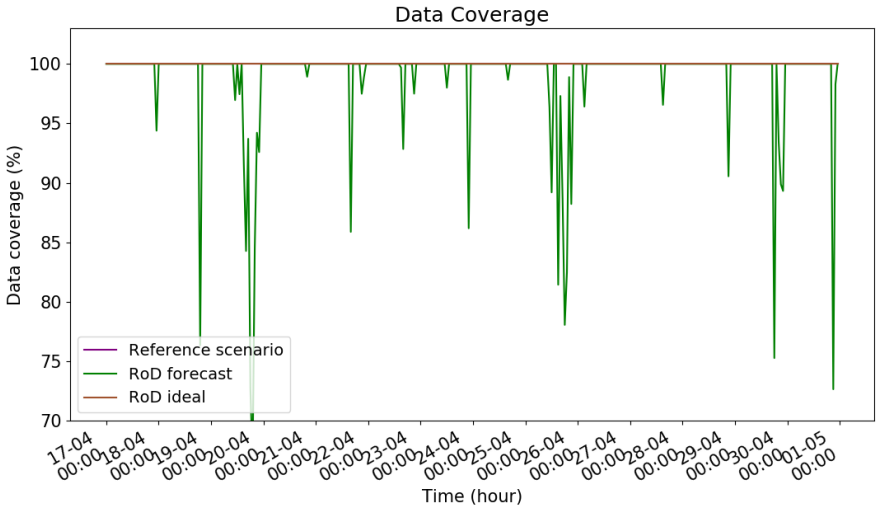


Figure 35 The capacity across time for the scenario of SanSiro

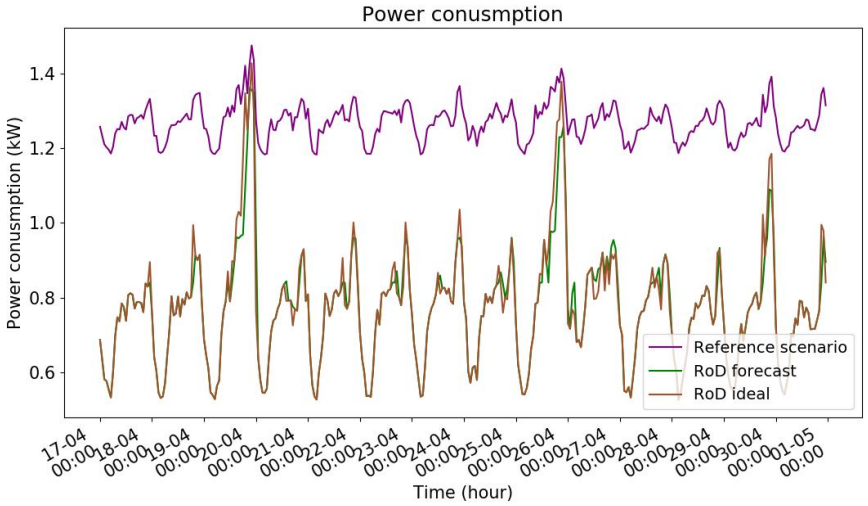


Figure 36 The power consumption across time for the scenario of SanSiro

It is conclusive as from Figure 35, that the stateful-RNN totally missed estimating the traffic spikes which resulted in losing close to one third of the traffic during those timeslots. This may only be analyzed if there was more data available for each individual BS.

Analyzing the second worst covered area, Linate, the performance for the power consumption over time is presented as in Figure 37.

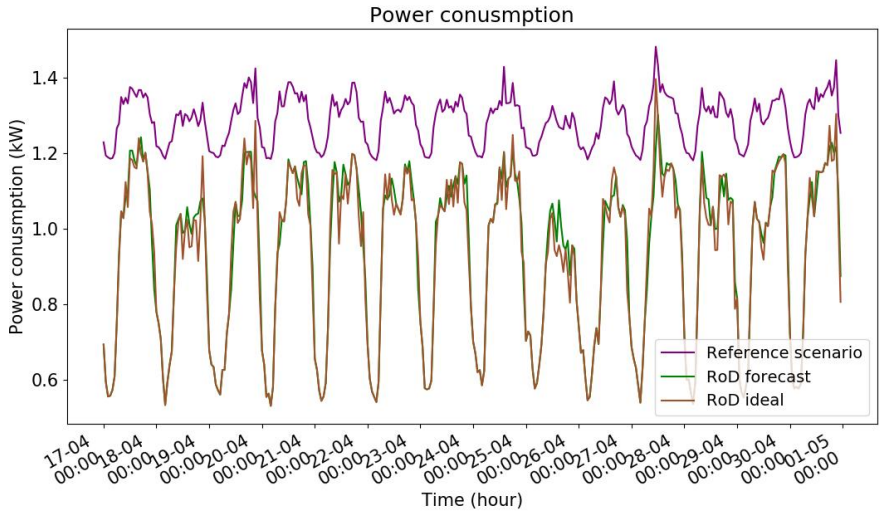


Figure 37 The power consumption across time for the scenario of Linate

Although in Figure 37 shows are no obvious dips for extreme savings in the area, the erratic behavior of the power during the peak hours cumulates to losing 1% of the total service on the area. To show the coverage drops the following Figure shows the dips in coverage per timeslot. From the Figure 38, we can understand that the implementation inaccurately estimated the borderline cases where the station might be switched off.

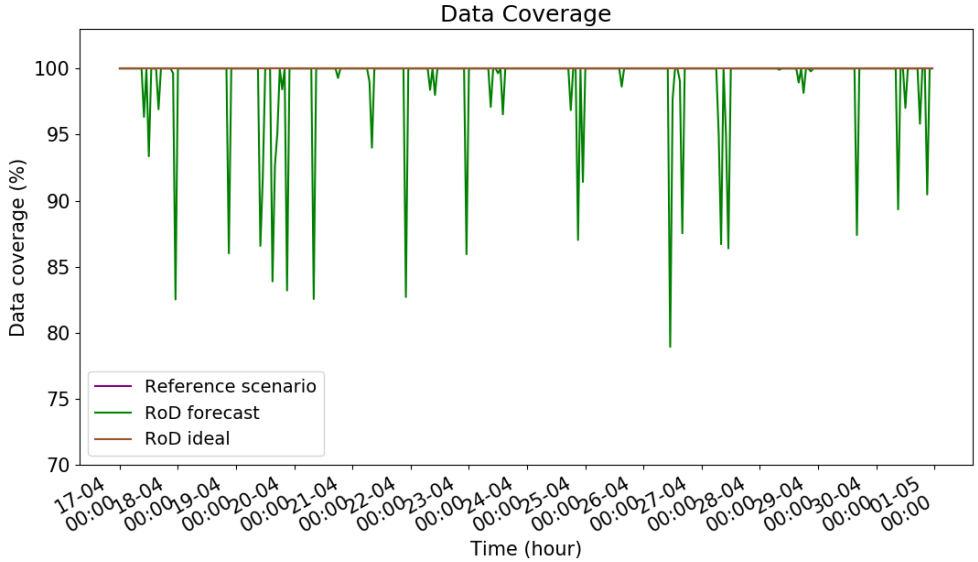


Figure 38 The coverage across time for the scenario of Linate

From Figure 38, we can see that contrary to the SanSiro scenario, in Linate the coverage drops are only of around 15% mark but occur more often. It can be noticed

that even with nearly 99% coverage using the ML solution only as a tool for estimation results in descent coverage for the scenarios where the traffic has high variance.

And finally, a very curious case of this implementation, that no other estimation machine except the stateful LSTM achieved is the FS case. In the FS scenario, the machine estimated so well that it managed to provide 0.5% coverage loss for 2.8% energy efficiency loss. To look closer, we analyze the power consumption graph to understand this behavior.

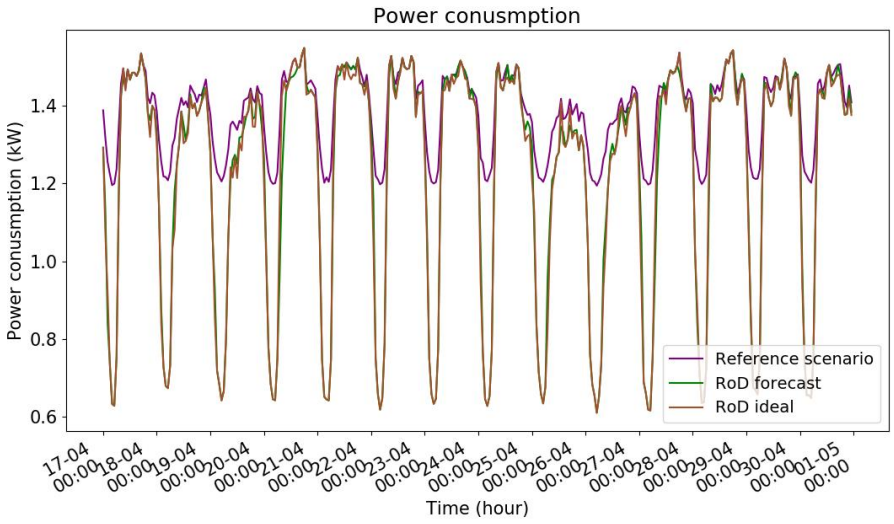


Figure 39 The power consumption across time for the scenario of FS

As it can be inferred from the graph, the Central train station of Milan ***FS***, has highly regular traffic that has very little to no noise. But also, for this station the difference between the timeslots where the station is active versus the ones where it is not, is very distinct. Therefore, it can be understood that RNN's such as the stateful one, have an incredible power to estimate highly regular traffic such as the one of FS. As a matter of fact, as it can be seen from the coverage graph for the case, there has been only one major dip in performance.

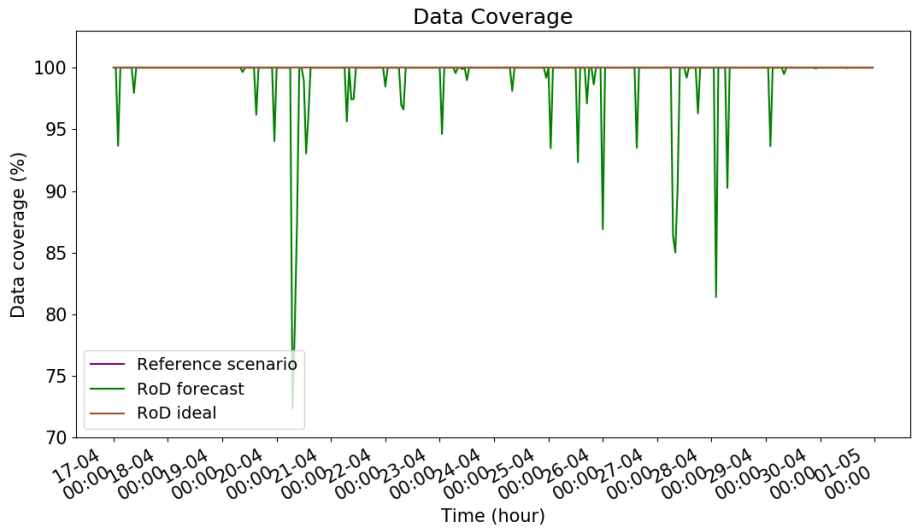


Figure 40 The coverage across time for the scenario of FS

A final note on stateful RNNs can be that they have an immense power to use a long-term estimation by passing the states and therefore use it to extract information from the past. Even with this power of the RNN network on the best performing scenario of 0.5% coverage loss had a significant drop in QoS, one which would not be allowed in common practice. Therefore, we need to turn the attention towards the unified solutions to understand if a reasonable tradeoff between coverage loss and energy efficiency can be achieved.

4.3.3 Unified – 10

unified-10											
	1	2	3	4	5	6	7	8	9	10	11
C	1.87	1.45	1.06	1.72	0.85	1.29	1.84	1.39	1.28	2.81	1.99
E	0.49	-0.54	-1.56	-0.82	2.46	-0.81	-0.44	0.24	-0.56	-2.83	-1.90
0.1											
C	1.47	1.11	0.80	1.19	0.75	1.15	1.32	1.08	1.20	2.00	1.78
E	3.98	1.41	0.36	1.69	3.83	-0.36	3.04	2.42	0.51	-0.35	-1.22
0.2											
C	0.96	0.86	0.66	0.78	0.53	0.96	1.02	0.94	0.92	1.67	1.64
E	7.59	3.76	2.58	4.33	6.65	0.15	5.82	3.97	2.10	1.63	-0.38
0.3											
C	0.73	0.61	0.47	0.58	0.42	0.76	0.82	0.68	0.74	0.89	1.25
E	10.17	6.03	5.86	5.90	8.58	0.71	8.92	6.68	3.62	5.14	0.94
0.4											
C	0.52	0.51	0.34	0.44	0.36	0.68	0.44	0.55	0.63	0.76	1.03
E	13.81	7.59	8.48	8.07	10.71	1.22	11.84	9.14	5.67	7.02	2.14
0.5											
C	0.23	0.44	0.25	0.26	0.18	0.51	0.25	0.43	0.50	0.51	0.75
E	17.47	9.34	11.38	10.89	13.70	1.99	15.31	11.49	8.11	9.81	3.65
0.6											
C	0.13	0.31	0.14	0.19	0.10	0.40	0.17	0.35	0.28	0.25	0.58
E	21.57	12.00	14.12	13.67	17.07	2.68	20.03	14.13	12.77	13.44	5.52
0.7											
C	0.08	0.11	0.10	0.04	0.08	0.27	0.11	0.18	0.13	0.13	0.22
E	25.65	17.08	17.89	18.70	19.68	4.33	26.02	17.62	19.14	17.63	8.41
0.8											
C	0.02	0.05	0.10	0.02	0.07	0.23	0.05	0.16	0.01	0.09	0.12
E	34.07	23.95	22.53	28.22	24.85	7.08	34.58	22.99	32.33	25.57	13.11
0.9											
E	0.00	0.00	0.05	0.02	0.06	0.00	0.00	0.04	0.00	0.01	0.02
C	45.77	36.14	30.55	44.87	36.71	13.63	43.68	33.88	52.31	47.30	27.40

Table 4 Coverage (C) and Energy (E) values per each scenario for the implementation of unified-10, per each threshold adjustment (referencing to the row below)

The scenario numbering is as in: 1 - Residential, 2- Business, 3 - Polimi, 4 - Highway, 5 -FS, 6 - SanSiro, 7 - Residential2, 8 - Duomo, 9 - Industrial, 10 - Linate, 11 – Rho. As from the table above, the scenarios that the unified-10 solution had trouble with are quite different than those that the two-part solutions struggle with, as it follows.

The machine is able to compute all scenarios in the Milano metropolitan area. As such, the threshold values can be adjusted per scenario to better reflect the ability of the machine for that scenario. For this reason, on Table 4 there are green squares marked

to show all coverage values that managed to meet at least 0.1% coverage loss for the underlying threshold. If we assume that this as a minimal value for which we should provide network coverage, we can try to estimate the performance of the implementation on top of these scenarios, as if each scenario had a different threshold adjustment. If we only apply the thresholds that guarantee less than 0.1% coverage loss but have the best energy efficiency for the unified-10 implementation we can average as:

Coverage loss = 0.05 and Efficiency Loss = 24.6

The average values above for the efficiency were computed off all fields marked with yellow in Table 4, and the coverage with their accompanying coverage values. Of course, this type of cherry picking the results would not be possible in a real scenario as it may lead to lack of QoS if any unexpected traffic passes the network which may require a more robust threshold. This result is however useful to more closely, later, compare this solution to the ***unified-RNN*** one.

However, as it can be seen from the above three of the 11 scenarios need the threshold set to 0.9 to achieve less than 0.1% coverage losses. Therefore, we only take as viable the threshold of 0.9 for fully solving the switching problem, as it guarantees energy efficiency while containing the coverage losses. For the decision threshold of 0.9, the averages of implementation unified-10 are:

Coverage loss = 0.02 and Efficiency Loss = 37.48

The value of 37.48 average energy efficiency loss is a descent result considering the conservative thresholds that all scenarios need to undertake. As a matter of fact, most of the tested scenarios, when limited by the threshold of 0.9, resulted in 100% coverage or come extremely near to it. When imposing such strict decision threshold, the implementation behaves rather oddly towards the previously critical scenarios, Rho, SanSiro and Linate. As this solution provides one of the best results, it is beneficial to investigate few of the scenarios in more details to understand the behavior of the implementation.

Beginning with the formerly worst scenario in terms of energy efficiency, Rho. Here, with the threshold of 0.9 it appears to make useful 72.6% of the possible energy saving while maintaining the QoS losses at 0.02%. As it can be seen from Figures 40 and 41 the NN takes very careful approach to the energy efficiency when having the threshold set to 0.9. The power consumption is always very high in comparison to the power needed to provide ideal coverage, however, as it can be inferred from Figure 41 there was only one dip of QoS which was due to the first timeslot of the active hours, when

the traffic shape started to recover to the previous form. This dip in performance will not be very noticeable as it only impacts only 7-8% of the traffic within one timeslot.

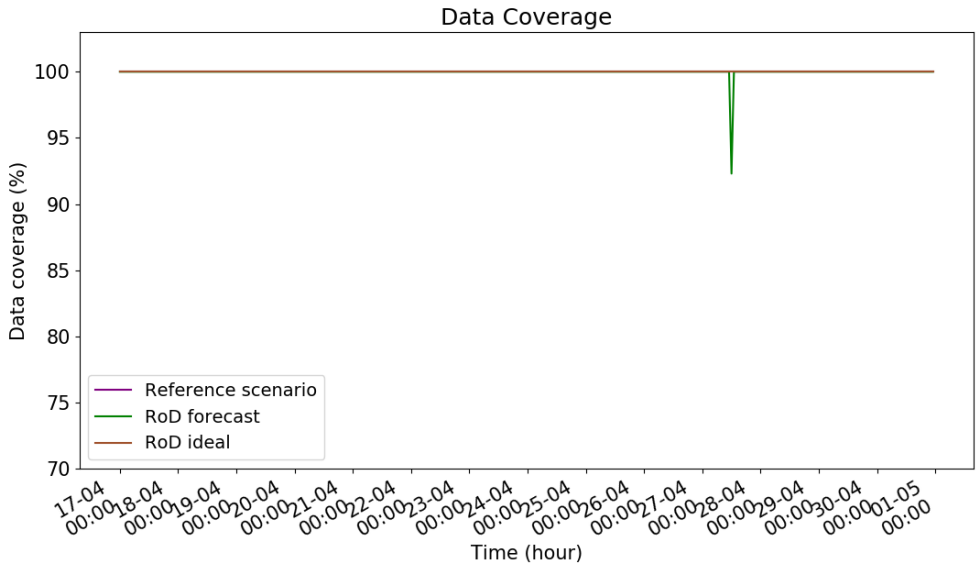


Figure 41 The coverage across time for the scenario of Rho for the unified-10 solution with threshold of 0.9

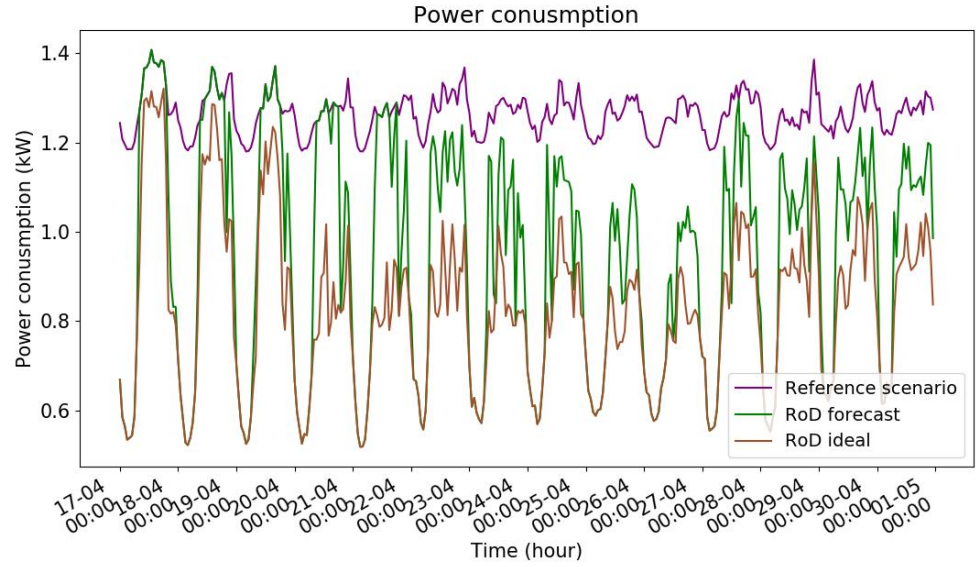


Figure 42 The power consumption across time for the scenario of Rho for the unified-10 solution with threshold of 0.9

Looking at Figure 42, we can see how well the algorithm approaches the volatile active hours of the BS. At the leftmost size of the graph we can see how high the active traffic gets, where we need all the BSs ON. However, as the holiday approaches, the traffic gets smaller incrementally until the 21st of march, the day before Easter. Even though

the algorithm was never trained to tackle holidays, it manages to combat the sudden decrease in traffic. It addresses it by having high insecurity to decrease the estimated traffic demand at the first two days but becomes certain that keeping all the BSs is useless. However, at the first day when the traffic demand returns for a small amount and that is the time where the ANN does its first mistake. This shows the limitation of the dataset to train the network in estimating a more dynamic traffic scenario such as Rho. But, even knowing the aforementioned drawback, the implementation does fit well with the ideal power consumption and serves while serving the customers in an efficient manner.

If we go back to Table 4 we can see that the unified-10 with decision threshold 0.9 shows 0.04 traffic losses for the scenario of Duomo. This is surprising since Duomo usually has not been part of the weakly performing algorithms while here it manages to show a spike in decreased QoS as shown in Figure 43.

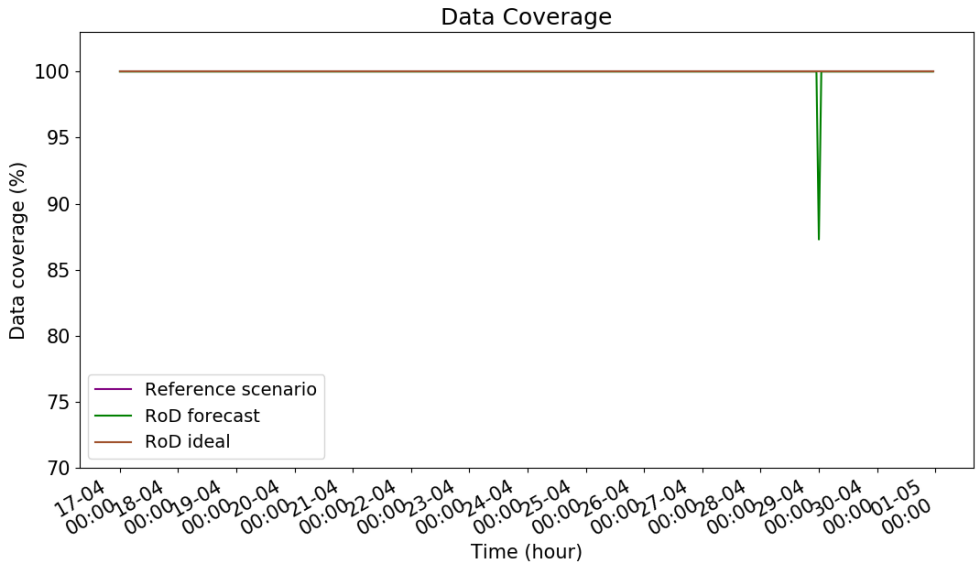


Figure 43 Coverage across time for the scenario of Duomo for the unified-10 solution with threshold of 0.9

Again, the loss of coverage is due to a single drop in QoS, which in this case is much more serious than the Rho case. Here, if seen from Figure 44 the drop of performance seems to not have breached the ideal power consumption. In other words, reaching better than ideal power consumption due to undesired BS switch off is hard to spot. This is due to the coverage drop being part of a premature prediction of the end of the active hours.

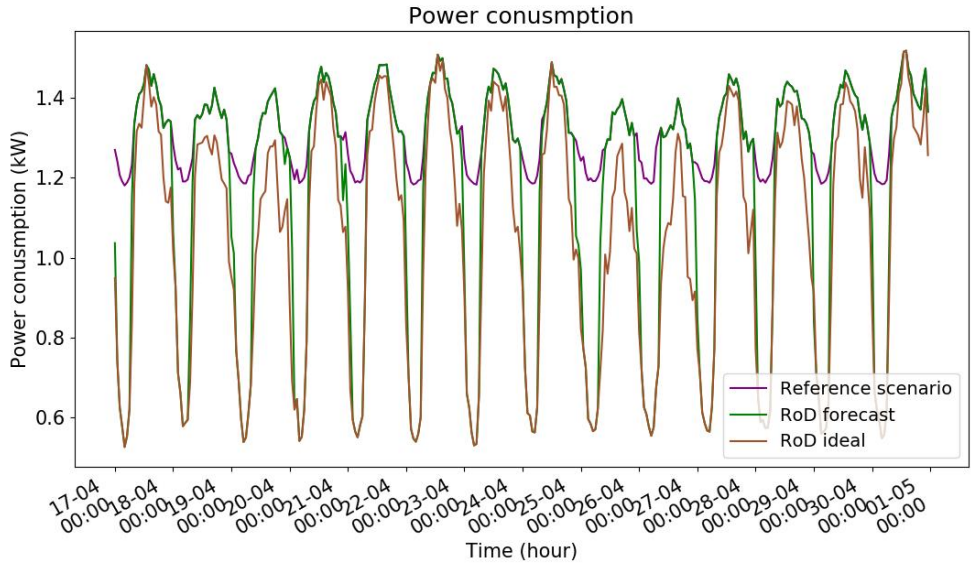


Figure 44 The power consumption across time for the scenario of Duomo for the unified-10 solution with threshold of 0.9

This phenomenon, of having a single drop that has a significant impact on the coverage of the scenario is repeating across many of the scenarios. This behavior is very odd and although it may lead to some unexpected drops in coverage, for now it shows that it is capable of battling the problem of energy efficiency versus coverage guarantee.

Finally, a good scenario to look into is the behavior of SanSiro where it manages to reach 100% coverage, for which showing the coverage graph is useless. This is an interesting result as SanSiro has been an estimation problem for all previous implementation, while here it is one of the best performing scenarios with 86.5% of the potential energy savings harnessed. Here in Figure 44, it can be very well seen the power of the adjusted threshold to make the BS to be slightly more careful when having distinct patterns that may lead to a big burst of traffic such as in the case of a football match on the stadium.

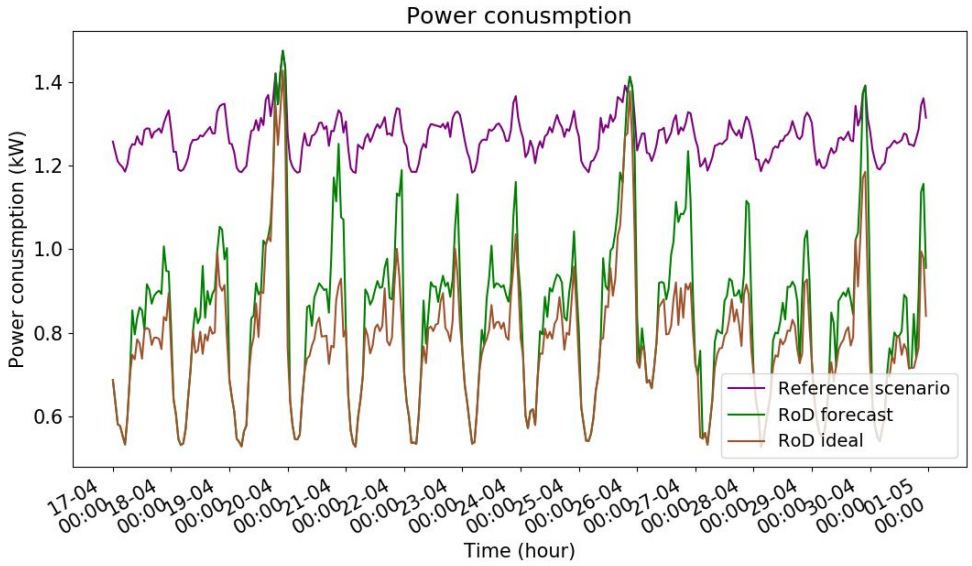


Figure 45 The power consumption across time for the scenario of SanSiro for the unified-10 solution with threshold of 0.9

4.3.4 Unified – RNN

The final solution under the scope is the Unified-RNN. To understand the shortcomings of the algorithm we again will look for the three scenarios that are hardest to be estimated, Rho, Linate and SanSiro; but also, the highly regular case of FS is investigated.

unified-RNN											
	1	2	3	4	5	6	7	8	9	10	11
C	2.01	1.57	1.26	1.70	0.71	1.74	2.15	1.39	1.49	3.43	2.89
E	-0.06	-2.64	-3.15	-0.49	3.76	-1.41	-2.44	-0.52	-2.74	-5.08	-3.83
0.10											
C	1.60	1.21	1.03	1.43	0.58	1.51	1.60	1.06	1.42	2.83	2.69
E	2.92	-0.46	-1.42	1.57	5.14	-0.89	0.64	1.51	-1.78	-3.09	-3.26
0.20											
C	1.17	0.86	0.79	1.13	0.41	1.12	1.21	0.77	1.20	2.25	2.46
E	6.13	2.19	0.06	3.65	7.17	-0.18	3.07	2.99	-0.14	-0.59	-2.45
0.3											
C	0.86	0.64	0.50	0.93	0.29	1.03	0.89	0.63	1.00	1.71	1.87
E	9.24	3.63	2.80	5.44	9.32	0.32	5.93	5.07	1.46	1.59	-1.04
0.4											
C	0.48	0.47	0.45	0.74	0.24	0.80	0.55	0.43	0.72	1.39	1.42
E	13.04	4.95	4.54	7.93	11.02	0.94	8.88	6.78	3.44	3.59	0.05
0.5											
C	0.24	0.31	0.39	0.67	0.16	0.66	0.41	0.30	0.49	0.94	1.21
E	16.53	7.63	6.07	9.33	13.56	1.57	11.31	8.57	6.32	7.04	1.40
0.6											
C	0.18	0.24	0.29	0.51	0.11	0.47	0.33	0.23	0.36	0.53	0.99
E	20.43	9.58	8.93	12.09	15.27	2.64	14.49	11.40	8.87	11.52	2.96
0.7											
C	0.10	0.06	0.15	0.22	0.09	0.41	0.09	0.14	0.19	0.33	0.73
E	24.22	13.39	12.68	15.35	18.34	4.13	19.02	13.69	13.42	15.21	4.80
0.8											
C	0.07	0.04	0.13	0.05	0.09	0.19	0.00	0.09	0.04	0.21	0.39
E	28.58	18.12	17.62	19.14	20.52	5.79	25.25	17.84	20.38	20.13	8.76
0.9											
C	0.07	0.00	0.02	0.00	0.07	0.05	0.00	0.00	0.01	0.10	0.18
E	35.03	26.38	26.46	29.58	26.29	9.33	35.60	23.99	35.69	27.44	17.92

Table 5 Coverage (C) and Energy Efficiency (E) values per each scenario for the implementation of unified-RNN, per each threshold adjustment

The scenario numbering in Table 5 is as in: 1 - Residential, 2- Business, 3 - Polimi, 4 - Highway, 5 -FS, 6 - SanSiro, 7 - Residential2, 8 - Duomo, 9 - Industrial, 10 - Linate, 11 – Rho. Here it is obvious that the network struggles with solving the traffic for scenario Rho and Linate. It does not even manage to achieve above 99.9% even with the strictest threshold adjustment for the scenario of Rho. But even with this in mind, if we average the best thresholds that provide at least 99.9% coverage for every scenario, in the same way in unified-10, the final average is:

Coverage loss = 0.08 and Efficiency Loss = 19.40

Comparing this result with the Unified-10's same results we can notice that the unified-10 receives slightly better ratio for coverage versus efficiency. Note that this cherry picking is only used as a normalized comparison between both threshold enabled solutions to show that Unified-10 shows better scaling ratio. Furthermore, even regarding that the implementation fails to provide less than 0.1% coverage for Rho, the average across all stations is still below 0.1% for the threshold of 0.9. The useful result for this scenario is the one when threshold 0.9 is applied. Even in this unscaled version the Unified-10 performs with a better, coverage versus efficiency ratio. Applying this decision threshold allows us to analyze the case where the service is guaranteed. The averages for this threshold are:

Coverage loss = 0.05 and Efficiency Loss = 26.7

To understand why the unified-RNN didn't manage to reach 0.1% we will look onto the performance graphs for the Rho scenario.

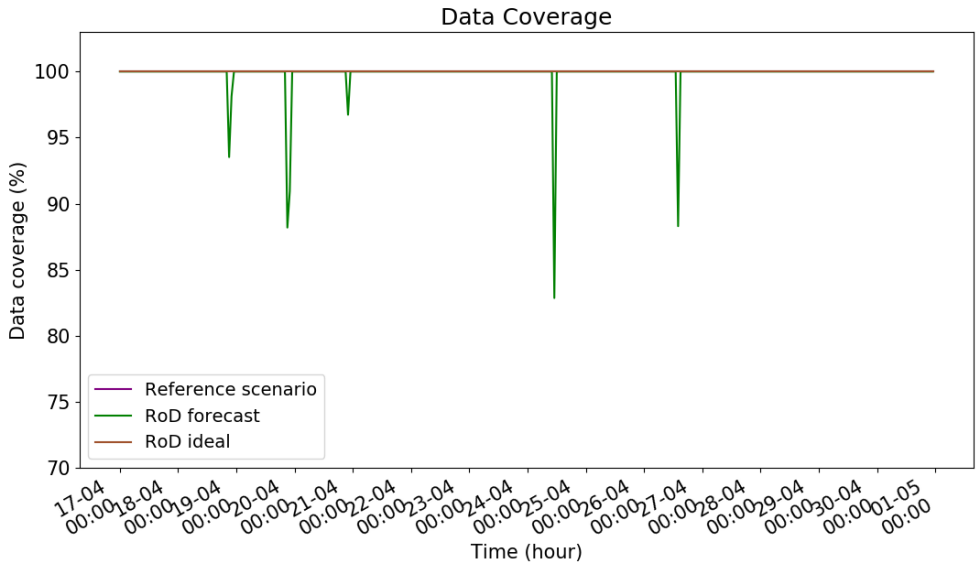


Figure 46 The coverage across time for the scenario of Rho for the unified-RNN solution with threshold of 0.9

OnFigure 45, we can notice that across time there have been three dips of performance where less than 90% of the data demands were served, and two additional small dips that contribute to the poor performance on this case. If we turn to the power consumption graph of Figure 46, we can see that some of the dips in performance were due to the network not coping well with the noisy active hours while trying to be greedier than the unified-10 solution. However, we can be very optimistic that the RNN background of the algorithm will be able to understand the phenomenon if a bigger training dataset was available.

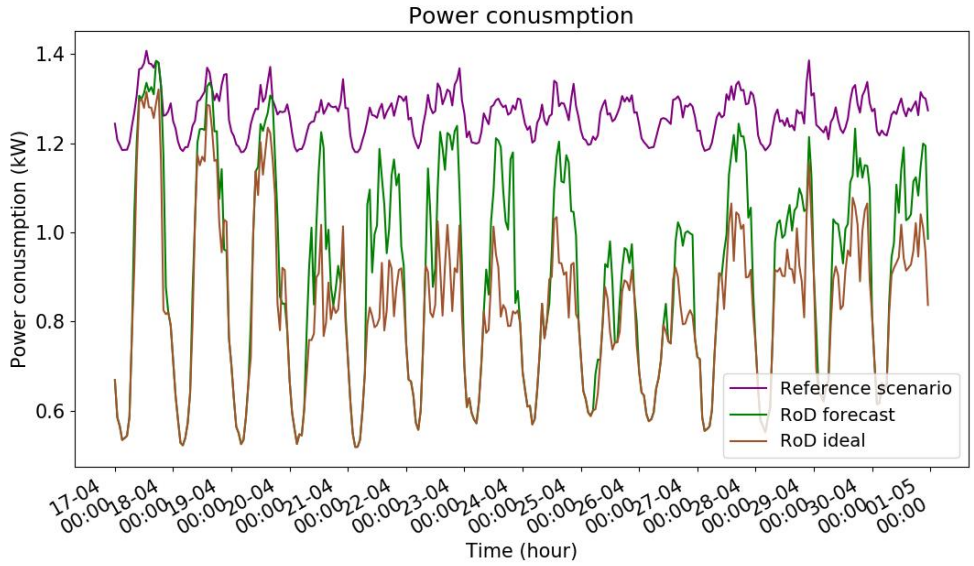


Figure 47 The power consumption across time for the scenario of Rho for the unified-RNN solution with threshold of 0.9

However, coming back to the graph of Figure 47, it is noticeable that while the unified-RNN managed to guarantee a solid coverage for Rho, the Energy saving is a lot higher than the one in the unified-10's solution. The adaptation towards the lower traffic of the first day of the Easter Holiday is almost instant and it also performs very well when the traffic stabilizes back to the more active state as in after the holiday.

The other weak case for the unified-RNN is the scenario of Linate. Here the solution managed to barely cover the 0.1% coverage loss mark for the threshold of 0.9, while not maintaining very good efficiency, by losing 27.44% of the potential savings. A very similar phenomenon can be observed in the Figure 48 to the one occurring in the other solutions. There are two minor drops in QoS, but one drop, on the 18th of march is unacceptably big.

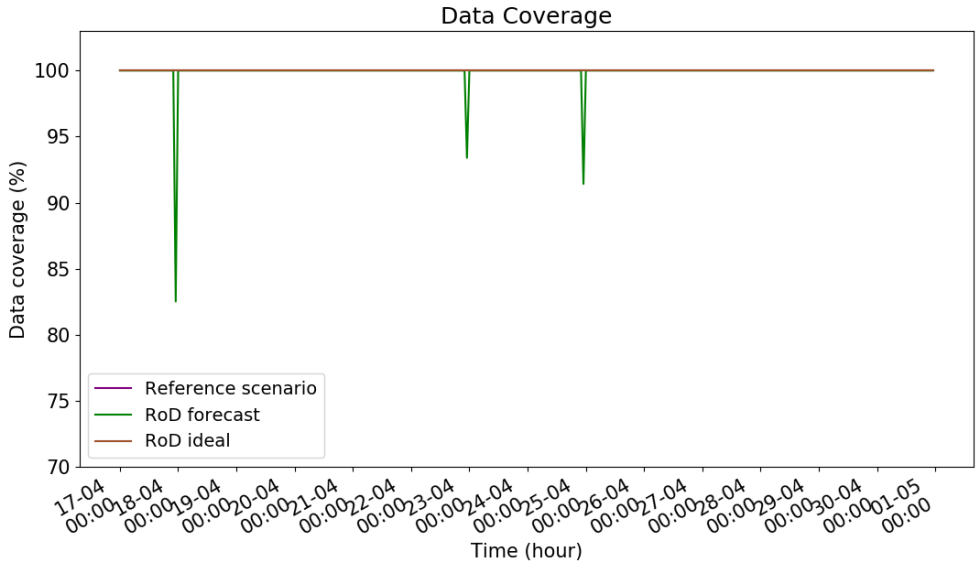


Figure 48 The coverage across time for the scenario of Linate for unified-RNN solution with threshold of 0.9

OnFigure 48 we can see that the significant dip in coverage performance reduces the coverage by 15% for that timeslot. Investigating the situation on the power consumption graph of Figure 49, we can notice that the big dip in performance is due to wrongful assumption of the solution that the active hours for the zone have finished one timeslot earlier. This is the most common mistake of the high certainty unified algorithms.

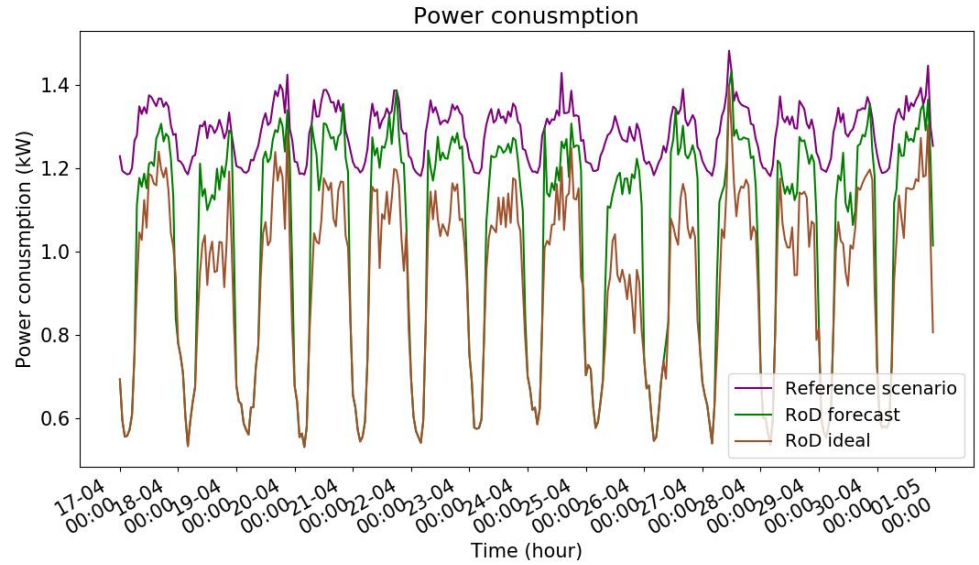


Figure 49 The power consumption across time for the scenario of Linate for the unified-RNN solution with threshold of 0.9

Even in this solution the mistake of misestimating the beginning and the end of the active hours happens from time to time and is the main drawback of the high certainty unified estimations. This phenomenon is very odd since the threshold is put to a very high value of 0.9 and continues with high certainty to claim the eligibility of that switch. This is the only concerning problem because when we try to guarantee the coverage, a single dip of performance that disables more than 10% of the potential of the network for that timeslot is detrimental for the unified-RNN to be taken as an unfeasible solution.

Although this type of NN implementation shows incredible performance in many implementations and even when performing only estimation, as in the case of stateful-LSTM, fails to provide better results than the ANN implementation of Unified-10. This was scrutinized and many performance tweaks to the algorithms were made, but the final conclusion is that the performances of all unified solutions are only limited by the available training dataset. This especially applies for the unified-RNN solution which does not manage to achieve stable internal structure to battle the complex problem of training a RNN for both estimation and optimization. For this reason, it is safe to conclude that the unified-10 solution outperforms the other solutions in terms of its high reliability.

4.4 Summary of results

With the final four implementations taken as best performing, dissected into their performance per scenario, few final remarks can be made about the approach of the Machine Learning estimation, optimization and unitary solutions.

The first and most important remark is that estimation ANNs tend to be very good at providing that, simple approximate estimations. As in the active hours of a BS the traffic is so volatile, even in their best performing scenarios it is possible to have timeslots with very little coverage because of a single error. This is since the two-part solutions are extremely greedy in nature. What that means is that the two-part solutions rely on having near perfect estimations and try to truly maximize, as the problem of energy efficiency is not that complex to solve, with the estimations performed. And when having those estimations, a very small difference in the critical regions, the value of keeping a BS ON or turning it OFF can mean a full one sixth of the underlying traffic to be transferred to the macro, which may or may not be able to serve. In that sense, the unitary solutions provide exquisite performance as their output is very different than a strictly binomial value of on and off.

In the unitary solutions the performance for coverage versus energy efficiency is understandable as those propagate a degree of certainty towards the value for which the activity of a BS is decided. This value is exploited by applying thresholds to the numbers of the turn off output neurons. Additionally, it can introduce large amounts in coverage gains at very marginal expense at energy expenditure. In the end, both scenarios performed very similarly which slightly goes in favor of the unified-10 solution as it requires less computing power to train the underlying neural network and provides more stable results for the guaranteed QoS cases.

In the end, the final note of the summary of results is that the RNN versus DNN scenarios when estimating resulted in an obvious advantage in favor of RNN. This behavior is not present in the unitary networks. In that sense, both unified-10 and unified-RNN performed nearly the same for the only reason that they displayed slightly different stage of the efficiency versus coverage curve. This behavior may be due to not having long enough training time sequences for the RNN to show its potential. The RNN, especially the stateful RNN, tend to show their potential when the internal states can be passed along for longer periods of time. Therefore, since only estimating is many times simpler task than both estimation and optimization, we require longer sequences than the two months provided.

Concerning the unified-10 solution as the most eligible one, it still suffers from the issue that all other solutions do. It tends to recommend turning off base stations at the beginning or the end of an active period when in fact there is still a big traffic demand. This is also very peculiar in the unified solutions where they output a real number between 0 and 1 to show their certainty towards that switch. In other words, in order for this phenomenon to occur in the unified-10 solution with high threshold of 0.9, the underlying implementation needs to be wrongfully certain that a BS switch-off is possible. This leads to solitary, but nonetheless, big drops in the Quality of Service.

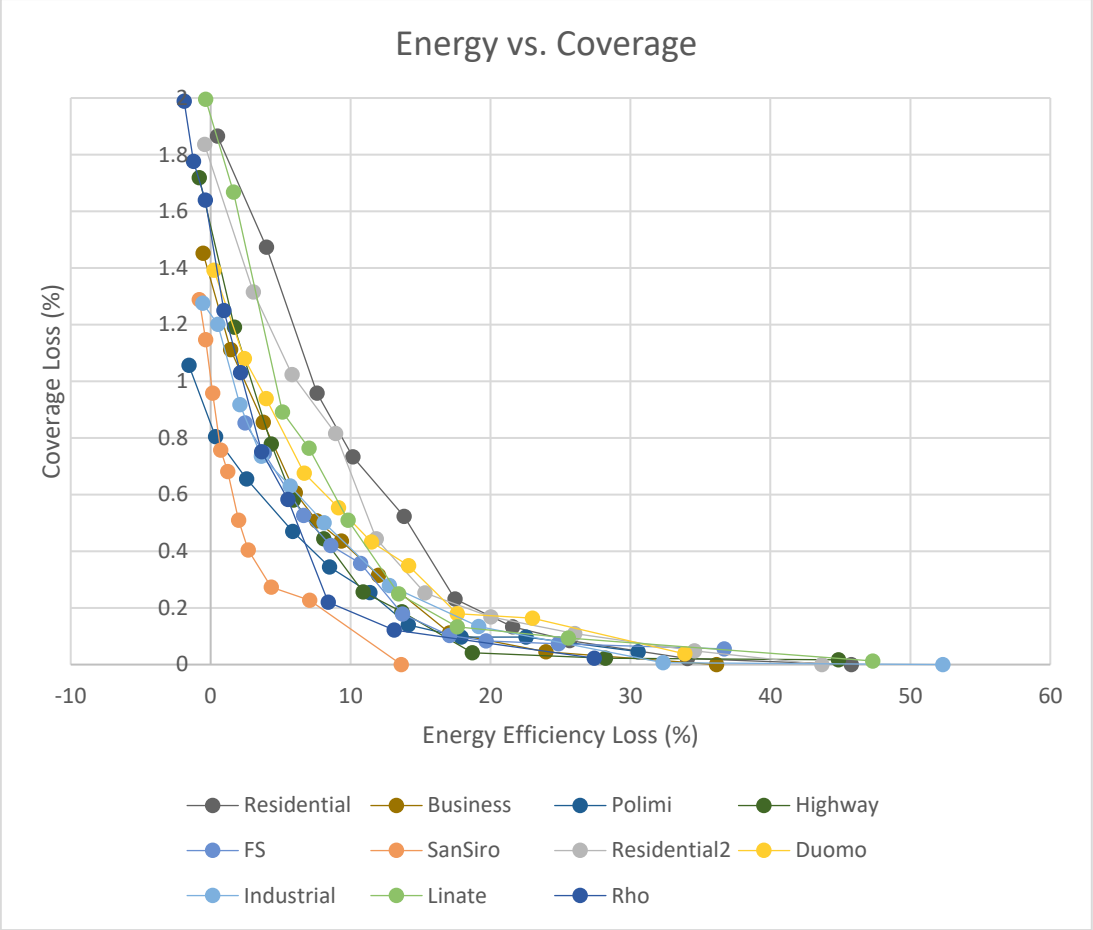


Figure 50 The Unified-10 performance for each scenario

The per scenario evolution of the unified-10 solution's thresholds is shown on Figure 50. Finally, the work can be summarized as a success since with the unified solutions the coverage percentage can be constantly above 99.9% while trading off some of the energy efficiency potential. Taking the unified-10 with decision threshold of 0.9 as the eligible solution results in average energy efficiency gains of 62.52% which may provide extreme reduction in operation costs for the service provider while guaranteeing 99.98% of the coverage.

5 – Conclusion

This research begun inspired to investigate decreasing the energy expenditure of heterogeneous networks that may be present in current and future cellular architectures. The inspiration was drawn as a responsibility towards the environment, but it would not have been possible if this did not translate into decreasing operating costs for the service provider as well. By computing the energy demands as of state of the art measurements for Macro and Micro cellular base stations, a novel approach to efficiently calculating the most optimal combination of the instantaneous traffic was suggested.

After analyzing the optimality of switching off BSs, Machine Learning techniques were analyzed under many different constraints. By considering both the two-part solution where estimation is for the following timeslot is done only by a machine and where both the optimization and estimation are done though ML. The combined machines proved very hard to train as they needed a bigger dataset than the only estimation counterpart. However, in the end the task was finished successfully and both types resulted in a rather satisfying performance.

The unified solutions provided very flexible and promising performance that could be customized per each scenario. With that potential, it was possible to provide at least 99.9% coverage across all scenarios, by only not exploiting 37% of the possible energy savings. However, how this percentage can translate to real world savings is up to the realistic implementation of the network. Although this may be very different from different service providers to different areas of the same service, the result still promises to provide significant impact on the networks. But with this, using ML for BS switching is proved to be a viable solution for both current overlapping networks and future ones.

Most importantly, allowing the BS switching paradigm to be understood as a feasible solution to the daily variations in traffic, may change the current paradigms in planning a cellular network. Having BS switching as an option while planning, may change the economic model of the service providers to extract better value for the customers by providing QoS by only increasing the capital expenses.

Although the feasibility of the paradigm was proven, the progress of such an implementation does not stop here. A complex ML paradigm that shows to provide better understanding in logit output such as in the unified solutions can be tested. This ML paradigm, going under the names of probabilistic NNs or Bayesian NNs, exploits the structure to pass down not only solid values but statistical ones. With such approach, the uncertainty off an output will be standardized and more tactically used to trade-off coverage and energy efficient.

Another paradigm that is surely to be implemented if ML enabled BS switching is implemented is reinforcement learning. This is a technique of training the system as an agent that observes its surroundings (past load neighboring BSs) and extracts the decision to switch off only then. This would be useful if a machine that has some weights is implemented in the cellular network but needs to be trained on-the-run by using a system of rewards and punishments.

Bibliography

[1] Index, Cisco Visual Networking. "Global mobile data traffic forecast update, 2012-2017." (2013).

[2] Van Heddeghem, Ward, et al. "Trends in worldwide ICT electricity consumption from 2007 to 2012." Computer Communications 50 (2014): 64-76.

[3] Mills, Mark P. "The cloud begins with coal: Big data, big networks, big infrastructure, and big power." Digital Power Group (2013).

[4] Andrae, Anders SG, and Tomas Edler. "On global electricity usage of communication technology: trends to 2030." Challenges 6.1 (2015): 117-157.

[5] Gruber, Markus, et al. "EARTH—Energy aware radio and network technologies." Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on. IEEE, 2009.

[6] López-Pérez, David, et al. "Towards 1 Gbps/UE in cellular systems: Understanding ultra-dense small cell deployments." IEEE Communications Surveys & Tutorials 17.4 (2015): 2078-2101.

[7] Han, Fengxia, et al. "Survey of strategies for switching off base stations in heterogeneous networks for greener 5G systems." IEEE Access 4 (2016): 4959-4973.

[8] Zeng, Ruhong, et al. "An artificial neural network based cell switch-off algorithm in cellular system." Computer and Communications (ICCC), 2016 2nd IEEE International Conference on. IEEE, 2016.

[9] Wang, Luhao, Shuang Chen, and Massoud Pedram. "Context-driven power management in cache-enabled base stations using a Bayesian neural network." 2017 Eighth International Green and Sustainable Computing Conference (IGSC). IEEE, 2017.

[10] Auer, Gunther, et al. "D2. 3: Energy efficiency analysis of the reference systems, areas of improvements and target breakdown." Earth 20.10 (2010).

[11] Correia, Luis M., et al. "Challenges and enabling technologies for energy aware mobile radio networks." *IEEE Communications Magazine* 48.11 (2010).

[12] Liljeqvist, I. (2018). *The Essence of Artificial Neural Networks – Ivan Liljeqvist – Medium*. [online] Medium. Available at: https://medium.com/@ivanliljeqvist/the-essence-of-artificial-neural-networks-5de300c995d6 [Accessed 3 Nov. 2018].

[13] Samuel, Arthur L. "Some studies in machine learning using the game of checkers." *IBM Journal of research and development* 3.3 (1959): 210-229.

[14] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." *Neural networks* 2.5 (1989): 359-366.

[15] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

[16] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).

[17] Lee, K. Y., Y. T. Cha, and J. H. Park. "Short-term load forecasting using an artificial neural network." *IEEE Transactions on Power Systems* 7.1 (1992): 124-132.

[18] Oxford Dictionaries | English. (2018). *overfitting | Definition of overfitting in English by Oxford Dictionaries*. [online] Available at: https://en.oxforddictionaries.com/definition/overfitting [Accessed 25 Oct. 2018].

[19] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research* 15.1 (2014): 1929-1958.

[20] Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks." *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011.

Appendix:

In the interest of simplifying the problem we display the linear load as A+Bx as in:

$$P = \mathrm{A} + \mathrm{B} \cdot \left(\rho_M + \sum_{i=1}^{N_s} \rho_{si}(1 - \tau_i) \right) + \sum_{i=1}^{N_s} [\tau_i \cdot (\mathrm{A} + \mathrm{B} \cdot \rho_{si})]$$

Now, the energy minimization problem can be expressed as trying to <u>minimize</u> the energy when deciding for a switch decision τ_i:

$$E = \mathrm{A}_M + \mathrm{B}_M \cdot \left(\rho_M + \sum_{i=1}^{N_s} \rho_{si}(1 - \tau_i) \right) + \sum_{i=1}^{N_s} [\tau_i \cdot (\mathrm{A}_s + \mathrm{B}_s \cdot \rho_{si})]$$

$$- [\, \mathrm{A}_M + \mathrm{B}_M \cdot \left(\rho_M + \sum_{i=1}^{N_s} \rho_{si}(1 - (1 - \tau_i)) \right) + \sum_{i=1}^{N_s} [(1 - \tau_i) \cdot (\mathrm{A}_s + \mathrm{B}_s \cdot \rho_{si})]]$$

Which can be simplified to <u>minimization</u> of:

$$E = \mathrm{B}_M \cdot \sum_{i=1}^{N_s} \rho_{si} - \mathrm{B}_M \cdot \sum_{i=1}^{N_s} \rho_{si}\, \tau_i + \mathrm{A}_s \cdot \sum_{i=1}^{N_s} \tau_i + \sum_{i=1}^{N_s} \tau_i \cdot \mathrm{B}_s \cdot \rho_{si}$$

$$- [\mathrm{B}_M \cdot \sum_{i=1}^{N_s} \rho_{si}\, \tau_i + \sum_{i=1}^{N_s} (\mathrm{A}_s + \mathrm{B}_s \cdot \rho_{si}) - \mathrm{A}_s \cdot \sum_{i=1}^{N_s} \tau_i - \sum_{i=1}^{N_s} \tau_i \cdot \mathrm{B}_s \cdot \rho_{si}]$$

$$E = \mathrm{B}_M \cdot \sum_{i=1}^{N_s} \rho_{si} - 2 \cdot \mathrm{B}_M \cdot \sum_{i=1}^{N_s} \rho_{si}\, \tau_i + 2 \cdot \mathrm{A}_s \cdot \sum_{i=1}^{N_s} \tau_i + 2 \cdot \sum_{i=1}^{N_s} \tau_i \cdot \mathrm{B}_s \cdot \rho_{si}$$

$$- \sum_{i=1}^{N_s} (\mathrm{A}_s + \mathrm{B}_s \cdot \rho_{si})$$

$$E + \sum_{i=1}^{N_s} (\mathrm{A}_s + \mathrm{B}_s \cdot \rho_{si}) - \mathrm{B}_M \cdot \sum_{i=1}^{N_s} \rho_{si}$$

$$= -2 \cdot \mathrm{B}_M \cdot \sum_{i=1}^{N_s} \rho_{si}\, \tau_i + 2 \cdot \mathrm{A}_s \cdot \sum_{i=1}^{N_s} \tau_i + 2 \cdot \sum_{i=1}^{N_s} \tau_i \cdot \mathrm{B}_s \cdot \rho_{si}$$

$$c = \sum_{i=1}^{N_s} (\mathrm{A}_s + \mathrm{B}_s \cdot \rho_{si}) - \mathrm{B}_M \cdot \sum_{i=1}^{N_s} \rho_{si}$$

$$E + c = -2 \cdot \mathrm{B}_M \cdot \sum_{i=1}^{N_s} \rho_{si}\, \tau_i + 2 \cdot \mathrm{A}_s \cdot \sum_{i=1}^{N_s} \tau_i + 2 \cdot \sum_{i=1}^{N_s} \tau_i \cdot \mathrm{B}_s \cdot \rho_{si}$$

For convenience multiply the problem by -1 and seek maximization:

$$-\frac{E + c}{2} = \sum_{i=1}^{N_s} \tau_i (\rho_{si} \cdot (\mathrm{B}_M - \mathrm{B}_s) \cdot -\mathrm{A}_s)$$

$$-\frac{E + c}{2\,(\mathrm{B}_M - \mathrm{B}_s)} = \sum_{i=1}^{N_s} \tau_i (\rho_{si} \cdot (\mathrm{B}_M - \mathrm{B}_s) \cdot -\mathrm{A}_s) \frac{1}{(\mathrm{B}_M - \mathrm{B}_s)}$$

$$-\frac{E + c}{2\,(\mathrm{B}_M - \mathrm{B}_s)} = \sum_{i=1}^{N_s} \tau_i \left(\rho_{si} - \frac{\mathrm{A}_s}{(\mathrm{B}_M - \mathrm{B}_s)}\right)$$

Where we have proven $\frac{\mathrm{A}_s}{(\mathrm{B}_M - \mathrm{B}_s)}$ to be a constant named $t_{eff} = 0.369$, the problem reduces to minimizing the negative value, or maximizing the positive value as in:

$$\sum_{i=1}^{N_s} \tau_i (\rho_{si} - t_{eff})$$

Under the constraints that:

$$\rho_M + \sum_{i=1}^{N_s} \rho_{si}(1 - \tau_i) \leq 1$$

$$0 \leq \rho_M \leq 1$$

$$0 \leq \rho_{si} \leq t_{eff}, \quad i \in 1,2,3 \ldots N_s$$

$$\tau_i = 0,1 \qquad i \in 1,2,3 \ldots N_s$$