University of Western Macedonia

Department of Informatics & Telecommunications Engineering

Design & Implementation of a Long Short-Term Memory Recurrent Neural Network for Traffic Prediction in Cellular Networks

Thesis Author

Anestis Dalgkitsis

Supervisor

Dr. Malamati Louta

March 1, 2018

Design & Implementation of a Long Short-Term Memory Recurrent Neural Network for Traffic Prediction in Cellular Networks

by

Anestis Dalgkitsis

Submitted to the Department of Informatics & Telecommunications Engineering
on March 1, 2018, in partial fulfillment of the
requirements for the degree of
Diploma Thesis of Informatics and Telecommunications Engineering.

Abstract

In this thesis, we designed and implemented a Neural Network which can identify recurrent patterns in various metrics used in cellular network traffic forecasting and conduct sufficient conclusions about the traffic in the future. Thanks to custom architecture and memory, the proposed Neural Network can handle prediction faster and even more accurate, in real life scenarios. An architecture like this can divide forecasting process in two phases, offloading training for a faster centralized system and giving the opportunity for low processing power devices to make instant predictions in a more power efficient way than other algorithms used by the industry today. This proposal may offer a solution for service providers to enhance cellular network performance, by utilizing effectively all available resources with smart strategic planning, that uses predictions by this proposed Neural Network architecture. This specific Neural Network is implemented with state-of-the-art, open-source software libraries developed by Google, offering seamless execution on both high and low end hardware. Multiple predictions were made in the same data-set, in order to provide a robust conclusion about the performance and precision of the proposed Neural Network against other algorithms from similar literature.

Acknowledgments

I would like to thank my supervisor, Dr. Malamati Louta, Associate Professor at University of Western Macedonia, for the patient guidance, encouragement and advice she has provided throughout my time as her student. I would also like to thank Argirios Darzentas, Access Transmission Design Manager, Vodafone-Panafon SA, for kindly providing the data needed to complete the research behind this thesis. A very special gratitude goes out to my close friends and fellow students, George Theodoros Kalampokis and George Angelopoulos for their support and friendship. For working together and building the right mindset to achieve higher goals in life and science. Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of writing this thesis. This accomplishment would not have been possible without them. Thank you.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

Chapter 1

Introduction

1.1 Background

Traffic forecasting is important for cellular service providers, in order to strategically plan their resources in an efficient and future proof way. Resources from energy consumption to link bandwidth are becoming more and more valuable, due to the exponential increase in cellular data usage. Research suggests that the increase will continue with faster pace, so Service Providers must be ready to deal with this phenomenon in the near future.

1.1.1 Core problem

There have been several works in the past that study subscriber generated traffic forecasting in cellular networks. Most of these prior studies try to understand wireless spectrum usage and characterize network performance and capacity using small scale measurements using a few mobile clients and outdated forecasting algorithms. To understand the network usage pattern and subscriber[1] behavior, a large scale comprehensive prediction algorithm comparison and analysis of a variety of metrics must be performed. A detailed algorithmic performance and accuracy study of subscriber Internet generated data traffic forecasting is still lacking and it is a need that must

[1]The terms client and subscriber are used interchangeably in this thesis.

be covered as quickly as possible.

1.1.2 Related work

It is noteworthy that only a few studies have investigated data traffic of cellular mobile Internet with modern and efficient algorithms.

Most related studies address only well-known prediction models that have challenged a few crucial problems. An appropriate example are all traditional time-series forecasting models including ARIMA[2], that due to the lack of a dynamic learning mechanism, cannot fit data sequences very well for irregular or non-periodic time series. AutoRegressive Moving Average[3], for example, is sufficient only for Poisson distributed values and has short-range dependence [29]. More complex models, such as Support Vector Machine and wavelet-based models, mainly use data during a short range of time near a predicted time point, they are not the first choice to model a time-series with self-similarity or long-range dependence [29].

Instead, Neural Networks, trained with the right data-set can outperform competing forecasting models, by range and accuracy.

1.1.3 Proposed solution

Due to the rich multifractal[4] behavior of cellular Internet traffic, modern signal processing algorithms like Artificial Neural Networks[5] can easily outperform model based algorithms [12].

Artificial Neural Networks, firstly introduced in 1940s as a set of functions that can model complex functional relationships, by utilizing interconnected mathematical nodes, called neurons, due to their similarity with the neurons of human nervous system. Instead of modelling the given data, Artificial Neural Networks are used as an alternative mathematical tool for classification, pattern recognition, predictions

[2]ARIMA stands for Auto Regressive Integrated Moving Average.

[3]Also known as ARMA.

[4]Multifractal is an abstract object used to describe and simulate naturally occurring objects. Fractals commonly exhibit similar patterns at increasingly small scales.

[5]The term Artificial Neural Networks will also be referred as ANNs for short in this thesis.

and other tasks performed in auto-correlated data.

In particular, Long-Short Term Memory[6] recurrent neural network[7] architecture remembers values over arbitrary intervals. LSTM is well-suited to classify and predict time series given time lags of unknown size and duration between important events. Relative insensitivity to gap length gives an advantage to LSTM over alternative RNNs, hidden Markov models and other sequence learning methods.

A big advantage Long-Short Term Memory Recurrent Neural Networks offer in modern cellular network architectures is that part of their computation can be offloaded to a centralized system. Since training phase can be performed by a server, all base stations can perform predictions in simple embedded hardware, allowing service providers the option to completely automate strategic resource planning in base stations themselves. That offers an energy, resource and cost efficient way for service providers to address the problem of resource planning and allocation.

1.2 Structure and purpose

In this thesis, we analyze subscriber generated Internet traffic data, collected by a 4G network. We then predict various metrics that can provide useful information for Service Providers, in order to strategically manage resources for future events. Before that, a brief introduction to competing algorithms and useful metrics will be made, in order to provide readers a spherical view of the proposed solution. Experimental results will be presented and discussed to show the superiority of Machine Learning algorithms in time-series prediction for data-sets with multifractal behavior, following their implementation in current infrastructure.

Brief outline of Thesis chapters

The structure of this Thesis includes the following chapters:

Chapter 1 introduces the proposed scheme. Shows the advantage of the proposed,

[6]Long-Short Term Memory will be referred as LSTM for short.
[7]Recurrent Neural Network is also know as RNN.

in-house build and tested, Long Short-Term Memory Recurrent Neural Network for predicting subscriber generated traffic, compared to competing algorithms.

Chapter 2 is a brief theoretical introduction about the metrics that can be extracted from the existing cellular network and how we can use them for forecasting.

Chapter 3 is an in-depth review of the most commonly used algorithms for time-series forecasting. We classify them into two big important categories. In the end, since we are aware how every algorithm works, we compare all of the advantages and disadvantages every algorithm has.

Chapter 4 begins with a brief introduction to Machine Learning and then focuses on Neural Networks. Then, we give clear and simple explanation of how Neural Networks work and how they can improve forecasting. After mentioning the advantages and the actions that require attention, we can safely read the proposed scheme of this Thesis and understand the advantages of it.

Chapter 5 is the heart of this Thesis. It is an extremely detailed analysis of how the proposed scheme works. Step-by-step explanation of the forecasting procedure and in-depth analysis of all the components needed to build our proposed Neural Network based forecasting mechanism will be given.

Chapter 6 begins with an introduction of how cellular networks work. It is really important to know all the network components and how they connect together before proceeding to the proposed scheme. Gradually we dive into the importance of Traffic-aware Network Architectures and then, we take a look at proposals in similar literature, as well as the way our proposed scheme can be implemented in the existing infrastructure.

Chapter 7 is a complete summary of every issue raised in this Thesis and a brief insight at the future work.

Chapter 2

Traffic Dynamics

Traffic forecasting in cellular networks is becoming increasingly important, as the rapid demand for radio resources requires an energy-efficient, network-driven architecture. We study the behavior of subscribers on the part of their traffic, their mobility and their activity on a time scale.

Traffic forecasting is based on the periodic similarity of traffic itself and requires a certain amount of previous information to reduce uncertainty. Generally, when the amount of previous information increases, the performance of the forecast improves.

2.1 Subscriber Dynamics

In general, we design the relationship between traffic generated by subscribers with their mobility and their level of activity in every dimension [20]. Then, we present the categories of data from the subscribers' side, which can be used to extract information and forecast their behavior in the future.

A brief reference will be made to the most common behaviors of each category as well as to their impact on resource design in cellular data networks.

2.1.1 Distribution of Subscriber Traffic

The simplest thing to do is to analyze the amount of data generated by subscribers on the network. The result is a data and time volume diagram.

A logical issue with this technique is that there are users who generate huge amounts of data traffic per day, as well as users that generate almost no data traffic at all per day. Studies [21] show that only 1 of subscribers generate more than 60 of daily network traffic and less than 10 of subscribers generate 90 of network traffic. This shows a significant imbalance in network usage among subscribers, with few subscribers using most network resources.

2.1.2 Subscriber Time Activity

We describe the subscribers' activity time by the number of days in a week or the number of hours of a day that generate data traffic. This approach deals with basic questions such as whether subscribers often or occasionally create traffic. To understand subscribers' time activity in detail, we are studying the distribution of the transmission time used by each subscriber.

In the commonly used 3GPP or 3GPP2 standards, a subscriber requests and in turn allocates a radio channel whenever it has data to send. The assigned radio channel is recalled by the network when the subscriber is idle for a certain period known as the sleeping period that can be configured for different networks. A subscriber can switch between active and idle status multiple times within a single IP mobile IP session. We mention the time a subscriber holds on a radio channel, regardless of whether he actually communicates, as the broadcast time [4].

In fact, active time gives us the time a subscriber uses radio and radio resources. In general, studies [21] show that a typical subscriber occupies the radio channel only for a short duration of the whole day.

2.1.3 Associate Subscriber and Traffic Activity

In this section, we define the relationship between the traffic generated by the subscribers and the frequency of their occurrence in the trace. In particular, we focus on demanding users, as they are the ones that carry most of the traffic. Studies [21] show that the appearances of demanding users on the trail are uncommon but in fact quite sporadic.

It is interesting to look at how effectively subscribers use radio resources and whether there is a difference between small and large volume users. To do this, we define a measurement called effective bit-rate. This is the ratio between the traffic volume generated by subscribers in transmission time[1] used by them. This measurement tells us how efficiently the assigned radio channel is used to send the traffic.

In general, it seems that most applications produce much less traffic [21] compared to other applications for the same consumption time consumed. The most likely reason for this is that applications tend to use the network sporadically [32] [21].

2.1.4 Subscriber Mobility

The signaling packets we receive provide enough information to track the base station and the cell sector connected to the mobile terminal at all times. This provides us with a rich set of data to study subscriber mobility based on the time stamp sequence of the base station to which it is connected. We have this data at any time, regardless of whether the subscriber is communicating or not [14] [5].

2.1.5 Base Stations Visited

In general, mobility is low in terms of the number of separate base stations visited [5] [8]. The data only includes the number of base stations visited, but not the physical extent of the trip.

[1]The actual time it actually occupies the radio channel, regardless of the traffic generated

2.1.6 Radius of Gyration

To capture the physical distance we are using, we use a concept called a radius of rotation or otherwise Radius of Gyration [14].

The radius of Gyration is the linear magnitude occupied by the subscriber trajectory. It is calculated by the average of the displacement of the recorded subscriber locations from a central point. The central point is the center of mass of the entire orbit. This records the extent to which subscribers move, as opposed to the actual distance traveled.

In general, the radius of rotation indicates the periodic nature of human mobility with a period of 24 hours and the tendency to return periodically to the same position [14] [24].

2.1.7 Associate Mobility of Subscribers and Traffic

Another way is to link the mobility of subscribers and the volume of traffic they produce. This simply categorizes subscribers based on their degree of mobility. The trend is that more mobile subscribers generate more traffic, as the average traffic generated by subscribers in the higher mobility class is about twice as high as the subscribers of the lowest mobility class [8]. This association of subscriber mobility and traffic has implications for resource planning and spectrum management [24].

2.2 Base Station Dynamics

In this section, we focus our attention on the behavior of the network as a whole or on base stations rather than focusing on subscribers.

2.2.1 Aggregate Load

We define load as the amount of data moving across all of the base station components at a given point of time. When forecasting, we usually point our interest in the total amount of data served by the given base station within a defined time frame.

The most common metric used for load forecasting is Throughput[2]. Throughput is the rate of successful message delivery over a communication channel. The throughput of the system can be affected by various factors, including the limitations of underlying analog physical medium, available processing power of the system components, and end-user behavior. When various protocol overheads are taken into account, useful rate of the transferred data can be significantly lower than the maximum achievable throughput. Those factors are mainly non variable, so the total throughput used by the base station can be proven a robust metric for load measurement.

Throughput measured at any time frame is a time-series. Time-series share many similarities with one dimensional signals. Given that they include many signal properties, that make them ideal for analysis and forecasting with more complex signal processing algorithms and not only as statistical mathematical models.

We will take advantage of this complex nature those signals have in later chapters in the thesis, to make forecasts and take a step further to understand the hidden patterns behind them.

2.2.2 Load Distribution

We can also characterize the total load across the entire network under review. By analyzing the daily traffic volume for each base station, we can discern recurrent geometrical patterns in the two dimensional space. This is a know behavior since statistics [21] show that about 20% of the base stations account for the total traffic load.

Carriers place cells to meet their best guess about where the most people will use their phones and other devices, but network demand can flare up suddenly in certain areas.

[2]Throughput is essentially synonymous to digital bandwidth consumption.

2.2.3 Auto Correlation

For rigorous analysis of the periodic behavior describing the network load we evaluate temporal correlation for load metrics. This will enable us understand the underlying trends and seasonal variations better.

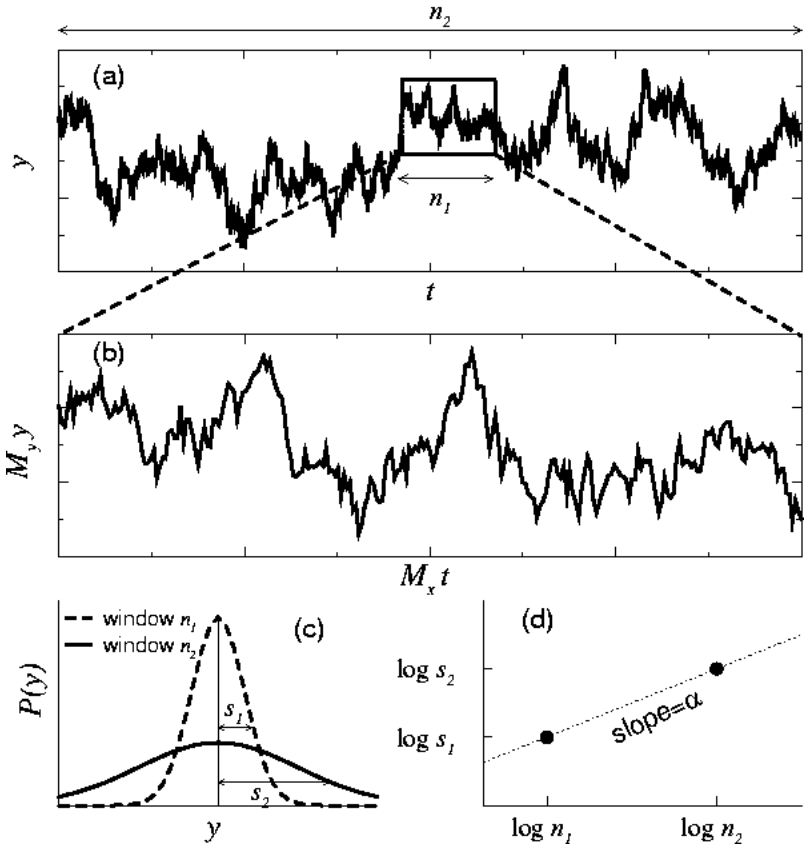


Figure 2-1: Self-Similarity of time-series.

Self-similarity is an adaptability of traffic in networks. Many factors are involved in creating this characteristic. The scaling region for traffic self-similarity is divided into two timescale regimes:

1. short-range dependence (SRD).

2. long-range dependence (LRD).

Experimental results [12] show that the network transmission delay separates the two scaling regions. This gives us a physical source of the periodicity in the observed traffic.

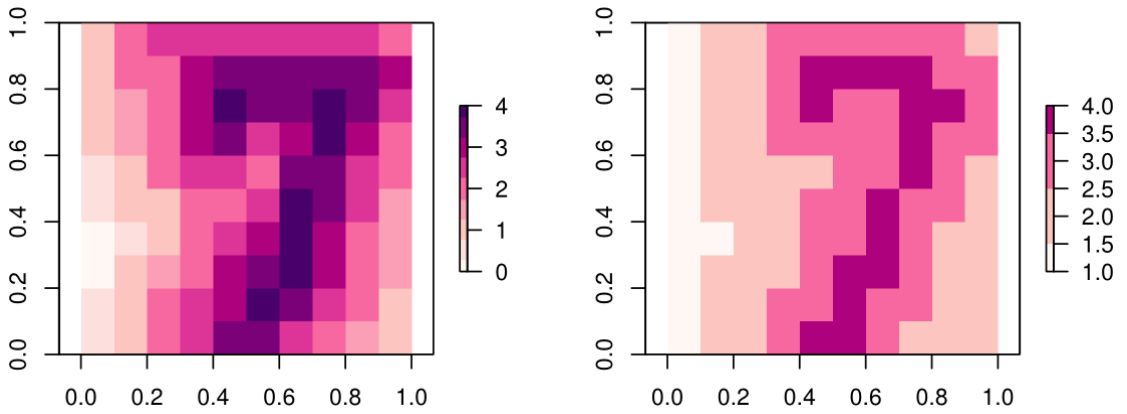


Figure 2-2: Spatial correlation between images.

2.2.4 Spatial Correlation

Our main goal is to determine whether the network load is spatially related. This can help the provider to properly distribute the resources.

It may also be useful [6] to predict the load of a single area, given the load of another area. To ensure that the area of interest is fully covered and that Quality of Experience[3] is at all times, mobile operators develop overlapping blanks, which in turn leads to similarities between traffic in the underlying cells.

[3]Quality of Experience is also referred as QoE for short.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Forecasting Models

Time series forecasting methodologies used for traffic prediction, fall into two main categories and the combination of those. Linear, Non-Linear or Hybrid models.

Linear and Non-Linear Models

A linear model is one in which the independent variable is added or multiplied together with the parameters. A non-linear model has exponents, logarithms, or other complicated functions of the independent variable and parameters. Some non-linear models can be reduced to linear models to make it easier to do the fitting.

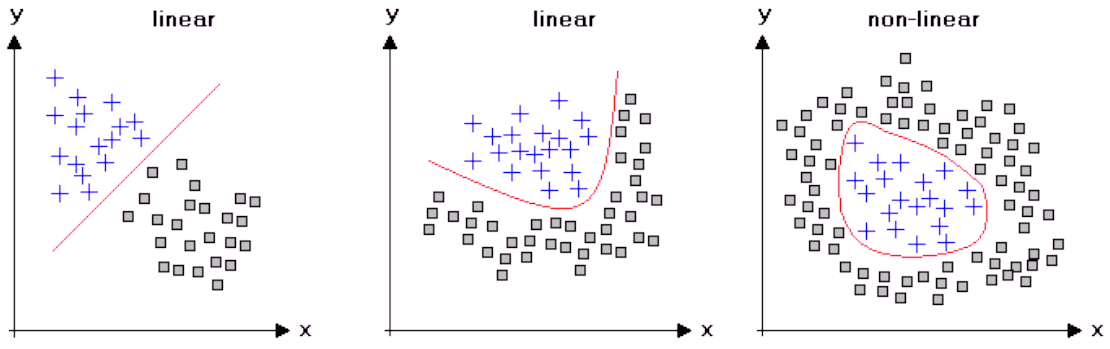


Figure 3-1: Linear and non-linear systems. The middle of the three figures above shows a linear discrimination line between the two classes, although the line is not linear in the sense of a straight line.

In order to choose a suitable model for traffic forecasting in the existing cellular

network architecture, we first need to analyze the nature of the data. Below we present various forecasting algorithms that can be applied to almost any type of time-series based information we extract from the cellular network, mentioned in the previous chapter.

3.1 Linear time series techniques

Linear models have drawn much attention due to their relative simplicity in understanding and implementation. They are used to model and predict the behavior of complex systems.

Linear models describe a continuous response variable as a function of one or more predictor variables. They describe the relationship between a dependent variable[1] y as a function of one or more independent variables[2] X_i.

3.1.1 Autoregressive Moving Average

The ARMA model is combined from both AR and MA models, to obtain an accurate model. The AR(p) model is written:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t \tag{3.1}$$

The notation MA(q) refers to the moving average model of order q:

$$X_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \tag{3.2}$$

The notation ARMA(p, q) refers to the model with p autoregressive terms and q moving-average terms. This model contains the AR(p) and MA(q) models:

$$X_t = c + \epsilon_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \tag{3.3}$$

[1]Dependent variable is also called 'response'.
[2]Independent variables are also called 'predictors'.

The general model was described in the 1951 thesis of Peter Whittle, who used mathematical analysis[3] and statistical inference. Autoregressive Moving Average is statistical model use with time series model for prediction [9]. The ARMA model applied to well-behaved time series data. ARMA model are considered in adequate for predicting data that integrate random. The ARMA linear time series model is very significant to predict network traffic [9].

3.1.2 Autoregressive Integrated Moving Average

In the statistical analysis of time-series, ARIMA is a generalization of an auto-regressive moving average[4] model, that provides a parsimonious description of a stationary stochastic process in terms of polynomials.

The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged values. The MA part indicates that the regression error is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The I, which stands for 'integrated', indicates that the data values have been replaced with the difference between their values and the previous values and this differencing process may have been performed more than once. The purpose of each of these features is to make the model fit the data as well as possible.

3.1.3 Fractional AutoRegressive Integrated Moving Average

Fractional AutoRegressive Integrated Moving Average[5] can be deployed to model and predict network traffic demand [28], which exhibits both long-range and short-range dependence.

The main difference between FARIMA and ARIMA is that the former uses fractional values instead of integers. specifically, FARIMA model shares the same form of representation as the ARIMA(p,d,q) process:

[3]Laurent series and Fourier analysis.

[4]Autoregressive Moving Average is also referred as ARMA for short in this Thesis.

[5]Fractional AutoRegressive Integrated Moving Average also referred to as FARIMA or ARFIMA in similar literature.

$$(1 - \sum_{i=1}^{p} \phi_i B^i)(1 - B)^d X_t = (1 + \sum_{i=1}^{q} \theta_i B^i) \tag{3.4}$$

In contrast to the ordinary ARIMA process, the "difference parameter", d, is allowed to take non-integer values.

When the time-series show statistical self-similarity with long-range dependence, FARIMA is the best option due to its fractional nature [12].

3.1.4 Seasonal AutoRegressive Integrated Moving Average

The seasonal part of an ARIMA model has the same structure as the non-seasonal part. It may have an AR factor, an MA factor, and an order of differencing. In the seasonal part of the model, all of these factors operate across multiples of lag[6]. A seasonal ARIMA model is classified as an ARIMA(p,d,q)x(P,D,Q) model, where P is number of Seasonal Auto-Regressive[7] terms, D is the number of seasonal differences and Q is the number of Seasonal Moving Average[8] terms [9].

3.1.5 Kalman Filtering

Also known as linear quadratic estimation, it can be used in any place containing statistical noise and uncertain information about any dynamic system. Linear quadratic estimators produce educated guesses of unknown variables.

Kalman filters use a series of measurements, observed over time and they are ideal for systems which are continuously changing. They have the advantage that they are light on memory, and they are very fast, making them well suited for real time applications.

[6]Lag, or S, is the number of periods in a season
[7]Seasonal AutoRegressive is SAR for short.
[8]Seasonal Moving Average is also called SMA in similar literature.

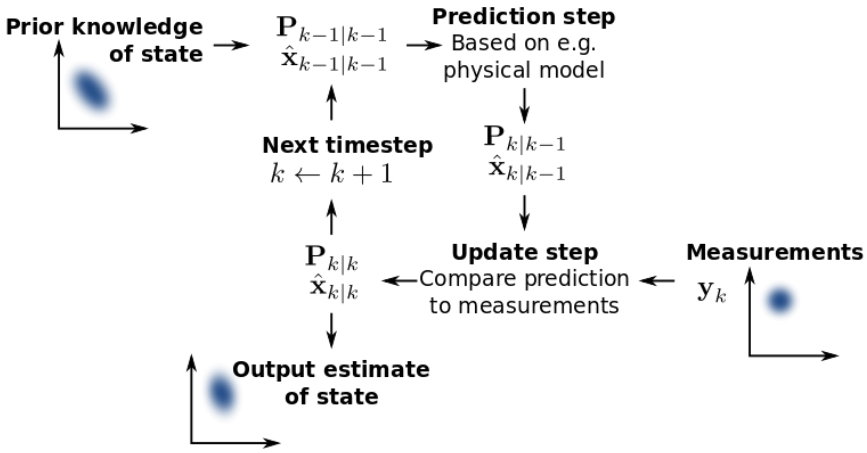


Figure 3-2: The Kalman filter keeps track of the estimated state of the system and the variance or uncertainty of the estimate.

3.2 Non-Linear processing techniques

Non-linear time series are generated by non-linear dynamic equations. They exhibit features that can't be modeled by linear processes such as, time-change variance, asymmetric cycle, higher-moment structures, thresholds and breaks.

3.2.1 GARCH model

The Generalized AutoRegressive Conditional Heteroskedasticity model[9] is an extension of the ARCH model. The GARCH is a non-linear time series model introduced to analyze financial data [3]. Based on this model, Nikkie C. Anand et al., have developed a forecast algorithm in similar literature [3], where a one step prediction recursively determines the forecast value of the traffic for the next time interval [9]. It can capture the bursty nature of Internet traffic [9]. The GARCH model can capture the conditional variance effectively, because of its dependence on variance at every time instant.

[9]Generalized AutoRegressive Conditional Heteroskedasticity is also known as GARCH

3.2.2 Artificial Neural Networks

Studies [15] show that nonlinear prediction based on Neural Networks, is more appropriate for network traffic prediction, than linear prediction models. In general, Neural Networks are widely used for modeling and predicting because they can learn complex patterns through powerful self-learning and self-adaptability. Neural Networks are able to predict virtually any function in an efficient and stable way, when the underlying data relationships are unknown.

Neural networks are a non-parametric adaptive modeling approach, based on the observed data and not on a detailed mathematical model. The architecture and parameters of the neural network are determined solely by the data-set. Neural Networks are known [15] for the ability to non-linear mapping and generalization, fault tolerance, adaptability and parallel processing capability.

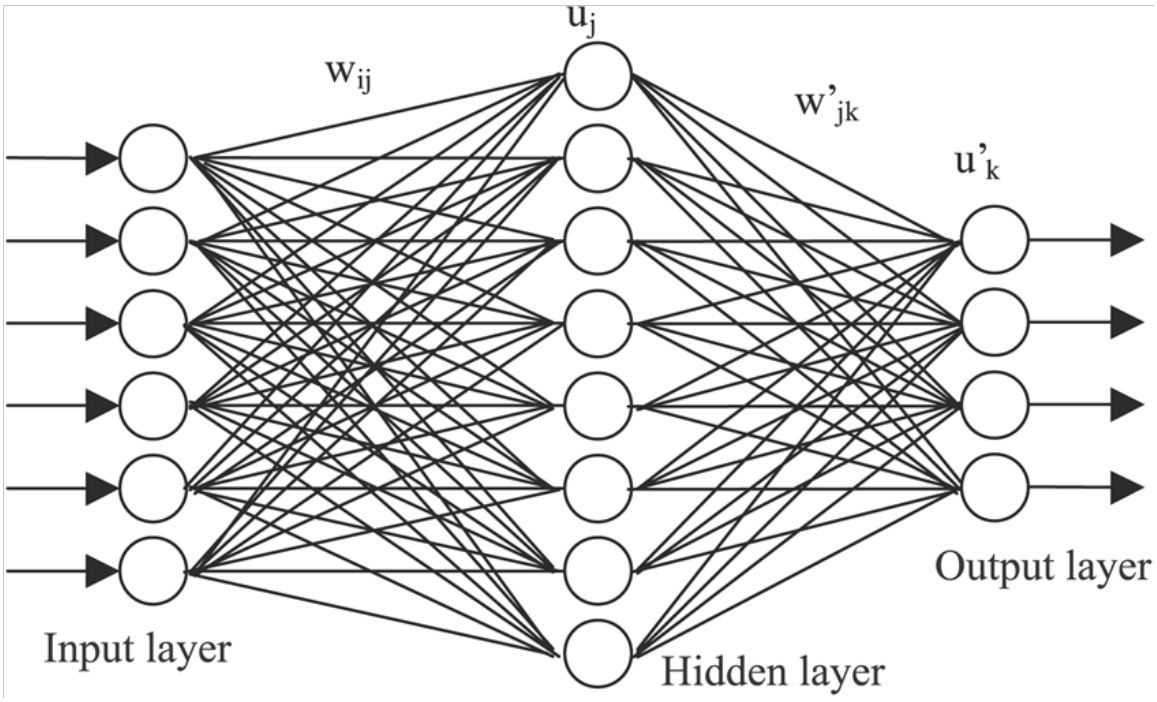


Figure 3-3: Simplified Neural Network.

Neural networks consist of interconnected nodes, called neurons. Each connection is characterized by a weight. The neural network comprises several layers of neurons:

1. An input layer.

2. One or more hidden layers.

3. An output layer.

The most popular neural network architecture is the feed-forward flow, in which the information moves through the network only forward, in direction from the input to the output layer.

The use of a neural network as a prognostic tool involves two phases:

1. The training phase.

2. The prediction phase.

Through the training phase, the training data-set is presented in the input layer and the neural network parameters are dynamically adjusted in order to achieve the desired output value for the input set. The most commonly used learning algorithm is the back propagation algorithm [26], where weights are continuously adjusted until the output error falls below the predetermined value. In this way, the neural network can learn correlated patterns between the input sets and their respective desired outputs. The prediction phase represents the testing of the neural network. A new input, not included in the training set, is presented to the neural network and the output is calculated, thus predicting the outcome of the new input data.

The number of hidden layers and the number of nodes in each layer are usually empirically selected. To be able to predict nonlinear values, the neural network must have at least one hidden level. Many hidden layers slow down the training process and increase the complexity of the Neural Network. In order to improve the non-linearity of the solution, the activation functions of the neurons in the hidden layer are sigmoid functions, while the output nodes have linear transport functions.

3.2.3 Support Vector Machines

Also know as SVM, is a linear supervised machine learning algorithm which, according to recent studies [16], can be used for both classification or regression challenges.

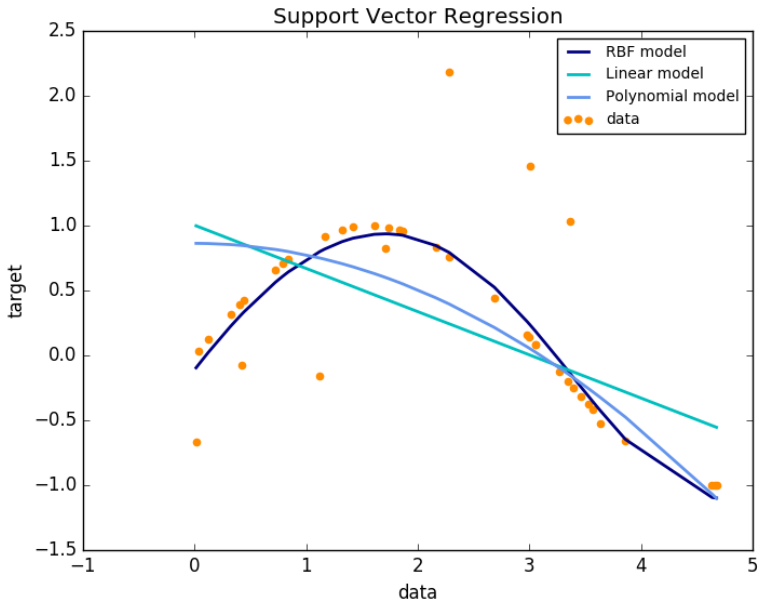


Figure 3-4: Support Vector Regression kernel comparison.

Given a data-set of training examples, each marked as belonging to one or the other of two categories, a SVM training algorithm builds a model that assigns new examples to one category or the other. A SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible [22]. The SVM is very costly in terms of time and memory consumption [9].

New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

3.3 Hybrid Model techniques

Hybrid Models are a combination of two or more approaches, for analysis of the network traffic.

3.3.1 ARIMA/GARCH hybrid model

The ARIMA/GARCH is a combination of linear ARIMA with GARCH variance [31]. Parameter estimation is the first step in fitting an ARIMA/GARCH model to observed data.

In a standard ARIMA(r, d, m)/GARCH (p, q) model, there are five parameters to be estimated.

1. The first step is to compute differenced parameter d. This parameter determines the time series stationary. For the network traffic data that behave non-stationary, we can use differenced operation to change it from non-stationary to stationary.

2. ARIMA model is a linear time series model that the mean is conditional changed but the variance is constant. The ARIMA back shift parameters order of r and m can be determined by auto-correlation function and its partial auto-correlation function. For the characteristic of auto-correlation function describes the correlation between the current states of the time series with the past, we can determine the moving average (MA) parameters order m straight. For the characteristics of the partial auto-correlation function describes the correlation between the current states innovation of the time series with the past, we can determine the auto regressive (AR) parameters order r directly.

3. The initial GARCH parameters p and q should be estimated independently [3] comparing with the ARIMA parameters r and m.

The parameter estimation procedure of ARIMA and GARCH models are both based on Box-Cox methodology. The ARIMA/GARCH model uses the lagged actual traffic values to predict one-step-ahead traffic value [31].

However there bound to be some prediction errors that cannot be avoided by the prediction model. This is because the real trace dynamic variance is out of expectation of GARCH model variance prediction. The traffic predictability depends on the traffic nature, the considered time scales and prediction step length. The non-linear time

series model ARIMA/GARCH can model and forecast the network traffic better than the traditional linear time series model. However, its prediction methodology is more complex and unstable [31].

3.4 Brief comparison

Traffic prediction plays an integral role in telecommunication network planning and optimization. The effectiveness of statistical models in forecasting future metrics related to network traffic has been proved useful. Several researchers introduced different approaches to forecasting network traffic in previous studies, most of them mentioned in this thesis.

In a general rule, if the traffic follows a Poisson distribution and the series exhibits short-range dependence, linear techniques including Auto-Regressive Moving Average and its variants, may be sufficient. If the traffic is self-similar or it exhibits long-range dependence, more complex models such as Artificial Neural Networks, Support Vector Machines, and wavelet-based combined models need to be employed [2].

According to this thesis, some certain types of Artificial Neural Networks like Long Short-Term Memory Recurrent Neural Networks, can outperform competing linear techniques, including Auto-Regressive Moving Average[10], due to the rich multifractal spectra [12] of the traffic data. FARIMA model also provides a solid option, because of its ability to eliminate long-range dependence by means of fractional differencing.

Traffic data are not only self-similar, but they also exhibit significant multifractal properties [12], so they can be exploited by complex algorithms based on signal processing techniques.

[10]Note that for maximum performance, a long data-set is required in order to fit the Neural Network in a efficient way.

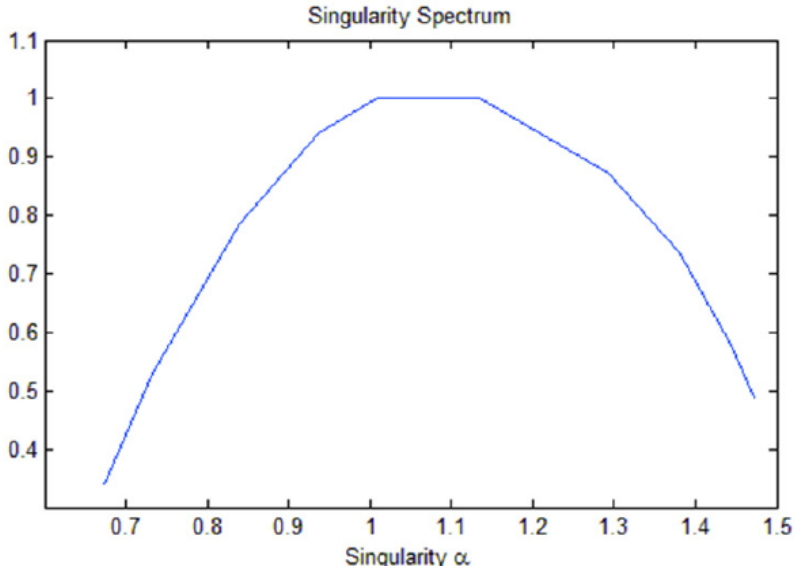


Figure 3-5: 3G Traffic multifractal analysis, as shown by Singularity Spectrum.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Machine Learning Approach

With machine learning, we are training a function f to map input X to output Y with minimal loss on the test data. In machine learning that is called, supervised learning.

Machine learning tasks are typically classified into three main categories.

1. Supervised learning. Neural Network is presented with inputs and their desired outputs. The goal is to learn a general rule that maps inputs to outputs.

2. Unsupervised learning. No outputs are given to the Neural Network, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself, as discovering hidden patterns in data.

3. Reinforcement learning. A Neural Network interacts with a dynamic environment in which it must perform a certain goal, without being programmed to do so.

In fact, the majority of practical machine learning uses supervised learning. It is called supervised learning because the process of an algorithm learning from the training data-set can be thought of as a teacher supervising the learning process. The data-set contains all the correct answers. The Neural Network iteratively makes predictions on the training data and is corrected by making updates. Learning stops when the algorithm achieves an acceptable level of performance.

Supervised learning problems can be further grouped into another two categories.

1. Classification. A classification problem is when the output variable is a category. A really common example is visual classification. Visual classification divides objects from images into categories, such as 'cats' or 'dogs'.

2. Regression. A regression problem is when the output variable is a real value. Regression approach mainly used for stock market price forecasting.

For subscriber generated traffic forecasting, we are mainly interested in regression. Given a sequence of numbers for a time series data-set, we can restructure the data in order to convert them into a supervised learning problem. We can do this by using previous time steps as input variables and use the next time step as the output variable [23].

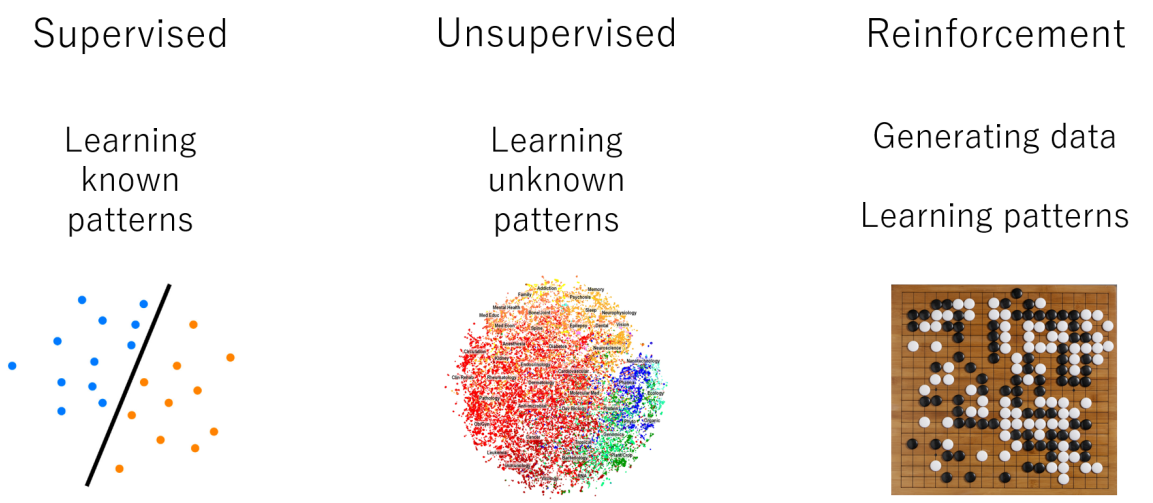


Figure 4-1: Basic Machine Learning categories.

Artificial neural networks are known as universal function approximators because they are able to learn any function, no matter how scattered input data are, with just a single hidden layer.

4.1 Deep Learning for prediction

Deep Learning is a branch of Machine Learning based on a set of algorithms that attempts to model high-level abstractions in data by using Neural Network architectures composed of multiple non-linear transformations.

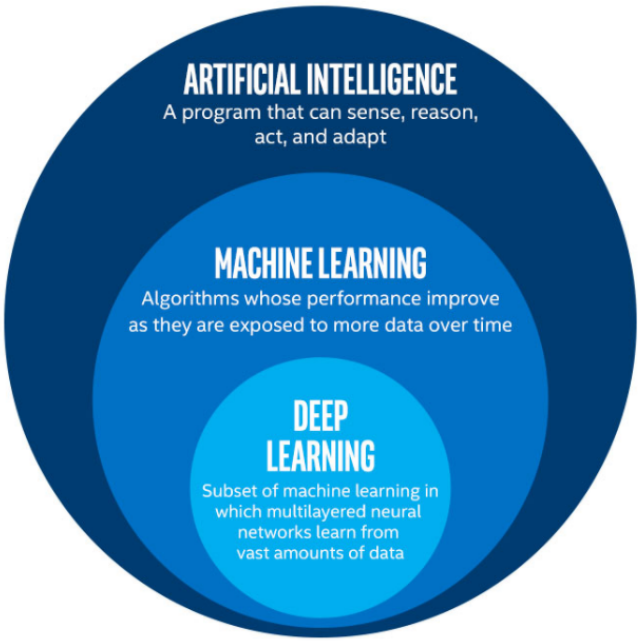


Figure 4-2: Artificial Intelligence is an umbrella term, encompassing machine learning and deep learning.

4.1.1 Neural Networks, the core of deep learning

Neural networks are capable of learning complex non-linear relationships and have been successfully applied to the problem of time series prediction. Recently they have been applied to the problem of traffic prediction.

Furthermore, a simple scheme that predicts the next value to be the equal to the current value will pass these statistical tests, since the sequences are essentially the same but shifted one step in time [23]. Several others claim [2] that neural networks can successfully predict traffic, however in each one questions must be raised about the validity.

4.1.2 Recurrent Neural Networks

Recurrent neural networks are a type of neural network that add the explicit handling of order in input observations.

This capability suggests that the promise of recurrent neural networks is to learn the temporal context of input sequences in order to make better predictions [30]. That is, that the suite of lagged observations required to make a prediction no longer must be diagnosed and specified as in traditional time-series forecasting, or even forecasting with classical neural networks [30]. Instead, the temporal dependence can be learned, and perhaps changes to this dependence can also be learned.

4.1.3 Long Short-Term Memory Recurrent Neural Networks

Recurrent neural networks, like the Long Short-Term Memory network, add the explicit handling of order between observations when learning a mapping function from inputs to outputs. The addition of sequence is a new dimension to the function being approximated. Instead of mapping inputs to outputs alone, the network is capable of learning a mapping function for the inputs over time to an output.

In addition to the general benefits of using neural networks for time series forecasting, recurrent neural networks can also learn the temporal dependence[1] from the data. Recurrent neural networks are able to learn the temporal dependence in the input data, without the need to specify a fixed set of lagged observations.

Technically, the available context may allow recurrent neural networks to learn:

1. Trend. An increasing or decreasing level to a time series and even variation in these changes.

2. Seasonality. Consistently repeating patterns over time.

[1]Temporal Dependence is the context of observations over time.

4.2 Deep Learning for traffic prediction

As mentioned before, there are various metrics that can be extracted from the cellular network.

Most of those metrics are easy to predict, because they present a strong correlation between chronologically ordered values. Time-series problems, such as throughput, time activity, radius of gyration and others, can be presented as signal and be forecasted by signal processing techniques, like Neural Networks [15]. Their predictability is mainly determined by their statistical characteristics.

4.2.1 Data-set length significance

Due to the nature of cellular networks, an enormous data-set of measurements can be extracted. That data-set can be used by a single or layers of multiple neural networks in order to harvest more detailed and complex conclusions about subscriber activity and usage trends.

The most significant thing to enable neural networks use this advanced prediction functionality, is an appropriate training set required [15].

A small training set can result into a prediction model that cannot extend over new unseen data[2] [25]. On the other hand, a large training set can result into a more accurate prediction model at the expense of a prohibitively high computational cost [15]. Hence, there is a trade-off concerning the selection of the training set for the prediction model. As a result, it becomes apparent that the proper selection of the training set is of great importance for the client performance of the prediction model.

[2]The effect of losing forecast accuracy because of insufficient data-set is also known as underfitting

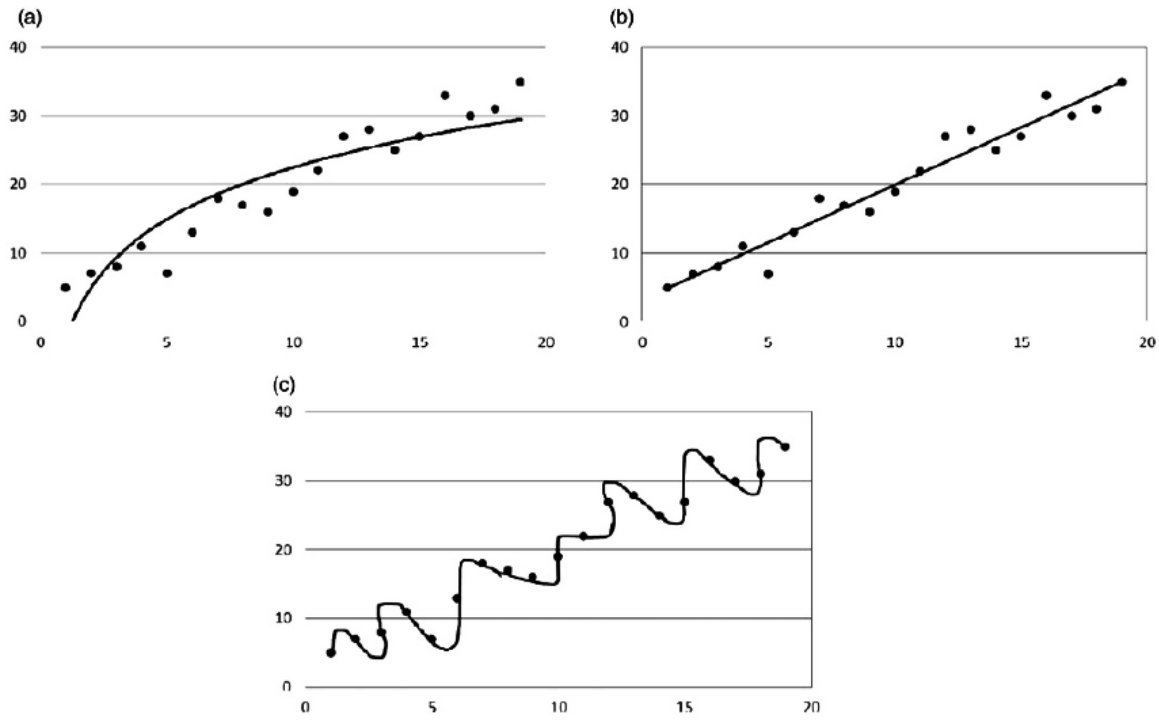


Figure 4-3: Neural Network under-fitting and over-fitting effects. Over-fitting offers better results in some scenarios.

Chapter 5

Design & Analysis of the Proposed Forecasting Application

In this work we build a terminal based software application, entirely based on the proposed machine learning approach analyzed in previous chapter, that can easily be executed in both embedded Unix and server systems combined. Given the data-set, our software can evaluate, normalize and forecast future values of any given metric, used by modern standards in network forecasting.

This application demonstrated, includes both the fitting phase[1] of the LSTM Neural Network and the testing[2] phase for demonstration.

5.1 Traffic Data

Subscriber traffic data was kindly provided by Vodafone, which is a major Service Provider in Greece. User personal information was removed from the data-set, in respect to user privacy.

Data-set includes cells spatially located in Crete, during the months of March till June, of 573 cells in total. We analyze subscriber generated traffic data collected from a deployed 4G network in 2016.

[1]The phase that the Neural Network is being trained.
[2]Testing phase, is the actual forecasting, held by the trained Neural Network.

5.1.1 Database Setup

All data samples are encoded in a csv file format. Csv files are commonly used in industrial applications, due to the fact that they are easy to handle by modern software libraries, and keep file size close to raw data size.

The csv file is organized as described bellow.

1. Time-stamp of the sample.
2. Downloaded traffic bits.
3. Activity time, in milliseconds.
4. Cell throughput, in megabits per second.
5. Cell ID, for classification.

This way, data are easy to filter and classify, in order to normalize and forecast.

5.1.2 Data Analysis

Before proceed to the actual forecasting, it's really important to analyze the nature of our data. Sample gaps and unwanted spikes, may affect the quality of the forecasts.

Rudimentary Seasonality

By performing data evaluation with Seasonal ARIMA with eXogenous regressors algorithm[3], we can clearly identify a rudimentary seasonality. It is important to note that a Seasonal ARIMA with eXogenous regressors algorithm is implemented in the application, developed for this Thesis.

Clear Trend

In a general rule, weekend behavior is slightly different than weekday behavior. The load distribution is very similar in the weekdays, while the distribution in the week-

[3]This algorithm is also known as SARIMA X.

ends are somewhat different. As expected, weekends see increased load than weekdays.

Noise

Spikes and sample gaps in all metrics cause algorithms to perform poor, forecast with lower accuracy than expected.

```
[i] Benchmark Report
 |- RBF     MSE: 3.24234140323 @ 18.16293692588806 (s)
 |- ARIMA   MSE: 6.53373309305 @ 25.01144289970398 (s)
 |- SARIMAX MSE: 11.2641659602 @ 9.040218114852905 (s)
 |- LSTM    MSE: 1.68576362799 @ 0.23576593399047852 (s)
```

Figure 5-1: Accuracy and computation time benchmark of the most popular forecasting algorithms, against of the proposed LSTM RNN. Less is better.

Linear time-series prediction algorithms like ARIMA and its extensions, seem to be affected by unwanted artifacts in the signal. These algorithms use a technique called moving average to calculate the trend in the series and it is directly connected[4] to the forecasting results.

Long Short-Term Memory Recurrent Neural Networks, due to their ability to memorize and forget weights [25] according to their activation function, they can forget[5] and re-evaluate their weights according to the correlation with rest of the signal. That ability makes Long Short-Term Memory Recurrent Neural Networks more versatile and resistant to errors, noise and sample gaps.

5.2 Data Filtering

As stated in the previous section, artifacts in the signal can cause forecasting algorithms to lose accuracy. Defending against this problem, data filtering is required before forecasting.

[4]It is a linear subsystem.
[5]This function of LSTM RNNs is called Dropout.

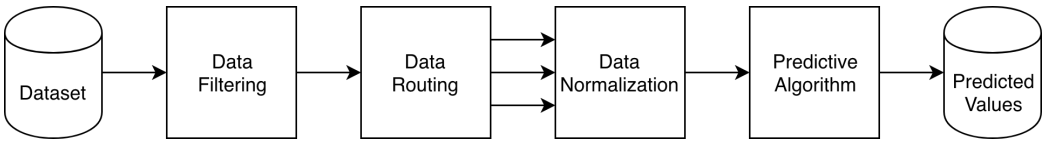


Figure 5-2: Data Flow of the Demonstrated Application.

5.2.1 Spike Filtering

The main problem that causes forecasting inaccuracy are, the so-called, value spikes. By definition, a spike is a comparatively large upward or downward movement of a value in a short period of time.

Statistical Spike Elimination

In this Thesis we propose an algorithm suitable for spike elimination, which does not distort the signal. It is very important to keep the signal intact, as close to the raw data.

The algorithm is called Excess Average and is based on simple mathematical calculations to eliminate the spikes with the least computational circles possible. Excess Average consist of two main parts, the Upper and the Lower Filter Loop.

The Upper Filter Loop, classifies all values above the signal. If a_i is greater than the a_e, then is classified as excessive.

$$a_e = max_j(a_j) * p \tag{5.1}$$

Where p is the excess percentage and $max_j(a_j)$ the statistical maximum.

The Lower Filter Loop, classifies all values below the signal. If a_i is less than the a_e, then is classified as excessive.

$$a_e = (max_j(a_j) + min_j(a_j)) - (max_j(a_j) * p) \tag{5.2}$$

Where p is the excess percentage, $min_j(a_j)$ the statistical minimum and $max_j(a_j)$ the statistical maximum.

If a value is classified as Excessive from the Upper or the Lower Filter Loop, a

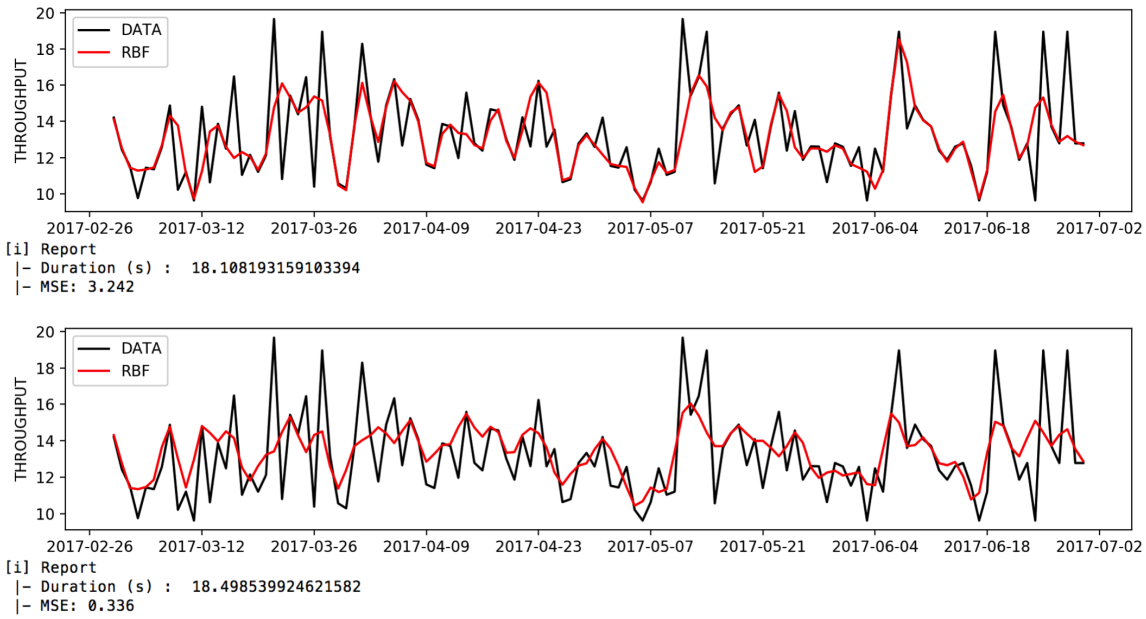


Figure 5-3: Evaluation accuracy comparison with Radial Basis Function forecasting algorithm. First Figure, evaluation without Excess Average. Second Figure, evaluation with Excess Average.

simple average value between the next and the previous value, replaces the current.

$$a_i = \frac{a_{i-1} + a_{i+1}}{2} \tag{5.3}$$

The only parameters Excess Average algorithm requires, are an array that holds the data-set and the percentage between the minimum and maximum value, that if passed by a threshold, it replaces that value with the average of the next and previous values.

```
function excessAverage(array, excess):
    for i in range(1, length(array)-1):
        if array[i] > (max(array) * excess):
            array[i] = (array[i-1] + array[i+1]) / 2
        if array[i] < (max(array) + min(array)) - (max(array) * excess):
            array[i] = (array[i-1] + array[i+1]) / 2
    return array
```

As we can see from the pseudo-code above, this algorithm is fast and efficient.

Despite that, keep in mind that this algorithm must be applied with caution not to distort the rest of the data.

Keep in mind that the following results presented in this Thesis, do not make use of the Excess Average spike elimination algorithm, in order to exhibit a clear view of the efficiency that the proposed machine learning algorithm has. It is advised to always filter data before continuing to the training phase, especially if additional information are available.

Advanced Spike Elimination

Simple spike filtering algorithms, like the Excess Average presented in the previous section, may be fast and energy efficient, but for a real world scenario a more advanced algorithm is needed.

In particular, Excess Average algorithm can be easily tweaked in order replace the excess values by a percentage, calculated from the average trend of the time-series and not the statistical minimum and maximum. Also, if additional information are available, a series of Boolean values that indicate specific dates with spikes, can be used to trigger the algorithm only on those dates. That way, we drastically reduce the possibility to distort values from the rest of the data and make use of the algorithm as a proper advanced spike elimination tool.

Beyond the tweaked Excess Average algorithm proposed, there are even more complex statistical algorithms that can give more accurate results, but with a major trade-off. Computational complexity and completion time.

5.3 Smart LSTM Training

As noted in previous section, it is demonstrated that an initial statistical processing of the collected data and the subsequent selection of the training set can efficiently improve the performance of the prediction model.

In similar literature [13], the authors use the notion of the Relative Standard

Deviation[6] so as to depict and exploit the statistical properties of the collected data. The RSD is a measure of precision regarding the collected data and is defined as

$$RSD(\%) = \frac{\sigma}{\mu} \times 100 \tag{5.4}$$

where is the standard deviation and is the average value of the data set.

Practically, a small RSD for a set of collected data implies that the measurements are averaged around their mean value, while a high RSD refers to collected data with great variations. The former may correspond, for example, to busy times in a crowded cell where the bandwidth is shared among all the subscribers, while the latter may refer to quiet times where a small amount of subscribers exploits a large portion of the available bandwidth [13].

We split our data-set in even blocks of data and select the one with the lowest Relative Standard Deviation.

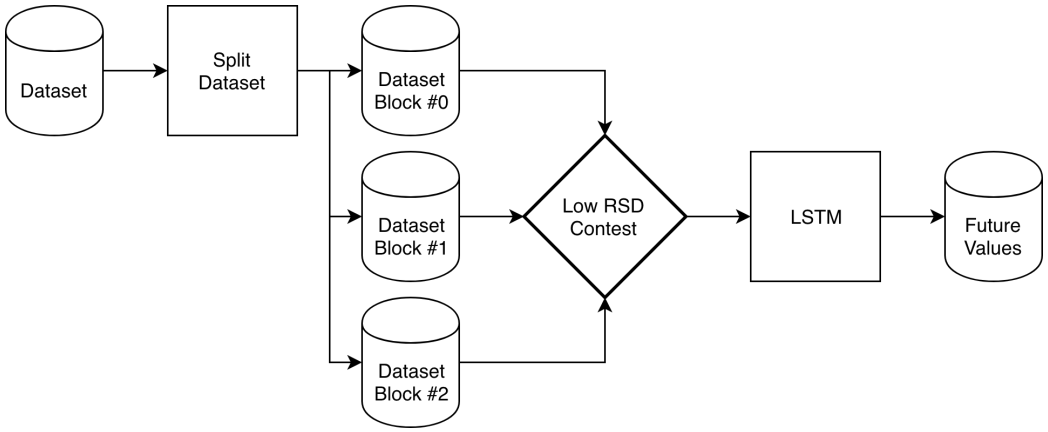


Figure 5-4: Proposed Smart LSTM Training.

That method produced higher forecasting accuracy in most cases, than the standard way of training the Neural Network with the whole data-set.

[6]Relative Standard Deviation, known also as RSD, is a standardized measure of dispersion of a probability distribution or frequency distribution.

5.4 Analytic Forecasting Software Breakdown

Finally, we dive into the heart of this Thesis, the proposed Long Short-Term Memory Recurrent Neural Network. Since the proposed architecture is complex enough, every aspect of it must be analyzed and understood.

5.4.1 Libraries

Multiple libraries used to reduce the complexity of building the proposed Neural Network. The most important libraries are listed below.

1. TensorFlow. An open-source software library for Machine Intelligence made by Google.

2. Keras. A high-level neural networks API, capable of running on top of TensorFlow, CNTK or Theano.

3. Pandas. An open source library providing high-performance, easy-to-use data structures and data analysis tools for Python programming.

4. sklearn. Simple and efficient tools for data mining and data analysis.

5. NumPy. Fundamental package for scientific computing with Python.

Specifically, Google Tensorflow enabled us to achieve maximum performance, even on embedded low-power devices.

5.4.2 Data Normalization

It is important to analyze every part of the code in our proposed application, in order to give readers the ability to continue our work.

Our software comprises of many important complementary functions, that transform the data before proceeding to the fitting phase of the neural network.

In our proposed scheme, Long Short-Term Memory data preparation consists of three main phases.

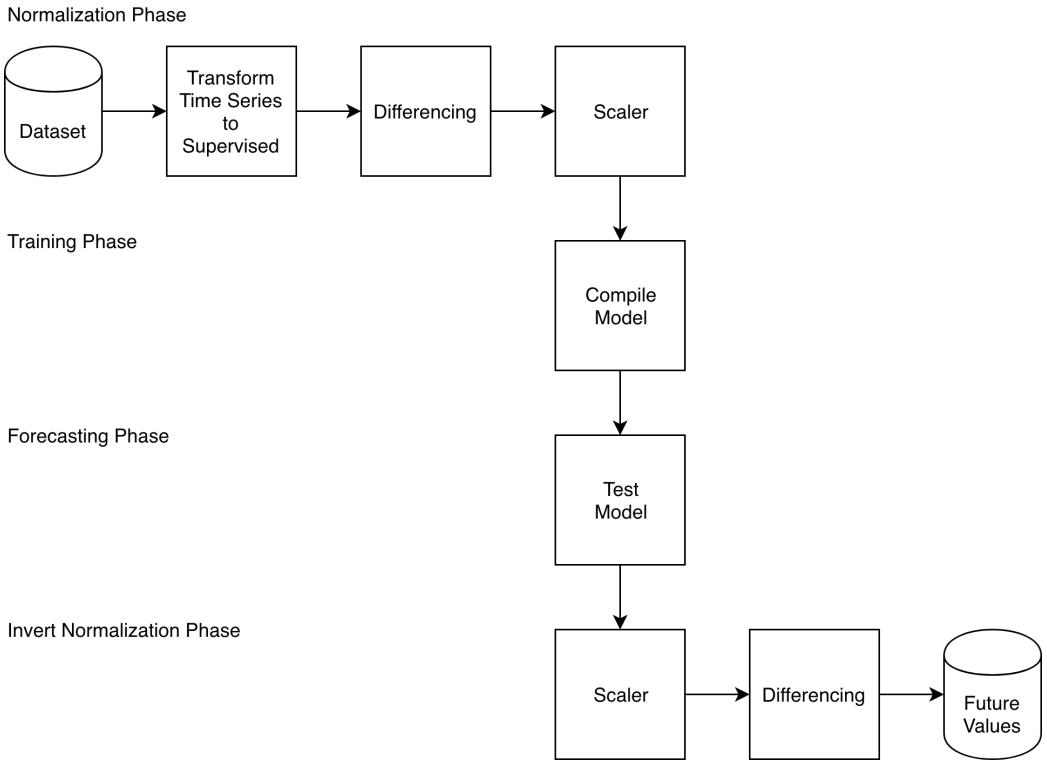


Figure 5-5: Complete Data Transformation Flow Chart.

1. Transform time series metric data into a supervised learning problem.

2. Transform the time series metric data to stationary data.

3. Transform the samples to have a specific scale.

Keep in mind that we should keep track of every transformation applied to data, in order to reverse it after the testing phase.

Transform Time Series to Supervised Learning

According to Keras library documentation, supervised learning data should be divided into input and output components.

As stated in previous chapters of this Thesis, traffic prediction is a time-series problem. In a time-series problem, we achieve this division by using the samples from the last time-step as the input and the sample at the current time-step as the output.

Pandas library function $shift()$ quickly push all values in a series down by a specified number of steps. We require a shift of one step, which will become the input variables. The time-series as it stands will be the output variables.

Then we concatenate those two series together to create a Data-Frame, ready for supervised learning. The series will now have a new position at the top without any value. This is called a NaN[7] and will be used in this position. Later, we replace these NaN values with zeros, which the LSTM model will have to learn.

Transform Time Series to Stationary

It is clear that our data-set is not stationary.

In short, this means that there is a structure in the data that is dependent on the time. Specifically, there is a specific observed trend in our data, mentioned in the previous chapter.

In order to proceed, the trend must be removed from the samples, then added back to forecasts later to return the prediction to the original scale and calculate an error score, in order to evaluate our forecast.

A quick and reliable way to remove trend from our data-set is by differencing the data. Simply, we subtract the previous time-step from the current sample. This removes the trend and provide us a difference series. A difference series are the changes to the samples from one time-step to the next.

Pandas library includes a completely automatic way to implement differencing, but in our proposed application a build-in-house differencing function is implemented. This is preferred for flexibility and further control over data.

Transform Time Series to Scale

Similar to other neural networks, Long Short-Term Memory Neural Networks expect data to be within the scale of the activation function used by the network.

[7]NaN stands for Not a Number. It is a numeric data type value representing an undefined or unrepresentable value.

The default activation function for Long Short-Term Memory Neural Networks is the hyperbolic tangent[8], which outputs values between -1 and 1.

It is very important the scaling coefficients values to be calculated only on the training data-set and be applied to scale the test data-set and the forecasts. This is to avoid contaminating the forecast with data from the test data-set.

By using scikit-learn transform classes, we transform our data-set to the range $[-1, 1]$ using the $MinMaxScaler$ class. This class requires data provided in a matrix format with rows and columns, so we reshape our arrays before transforming.

5.4.3 Long Short-Term Memory Model Development

The reason we prefer Long Short-Term Memory neural network over other types of neural networks is that it is a type of Recurrent Neural Network.

Building LSTM Layers with Keras

Recurrent Neural Networks have the ability to learn and remember over long sequences and do not rely on a pre-specified window lagged samples as input. This is referred as stateful, and it needs to be specified as $True$ in Keras when defining an LSTM layer.

LSTM layer in Keras keeps the state between data within one batch by default. A batch of data is a fixed-sized number of rows from the training data-set that defines how many patterns to process before updating the weights of the network.

In order to build an LSTM layer with Keras, input must be shaped to a matrix with the specified dimensions.

1. Samples. These are independent data samples, typically rows of data.

2. Time-steps. These are separate time-steps of a given variable for a given data sample.

3. Features. These are separate measures observed at the time of observation.

[8]Hyperbolic tangent is mentioned as tanh in mathematiical equations.

We will frame the problem as each time-step in the original sequence is one separate sample, with one time-step and one feature.

Proposed LSTM Neural Network Architecture

The batch size should be much smaller than the total number of samples in the dataset. It, along with the number of epochs, defines how often the weights are updated. Which practically means, how quickly the neural network learns the data.

Another important parameter used for defining the LSTM layer is the number of neurons[9]. This is a reasonably simple problem, given that is a time-series forecasting problem, so any number between 1 and 5 should be sufficient. Neurons number is a parameter set mostly empirically. Keep in mind, that there is a trade-off between optimal fitting time and forecast accuracy involved in this parameter.

We must set manually the neural network to be stateful, because the state in the LSTM layer between batches is cleared by default.

```
LSTM(neurons, batch_input_shape, stateful)
```

The network requires a single neuron in the output layer with a linear activation to predict the given metric at the next time-step.

Once the network is specified, it must be compiled into an efficient symbolic representation using a backend mathematical library. In this proposal, we use TensorFlow as backend, due to the state-of-the-art algorithms, flexibility and performance it offers.

In order to compile the network, we must specify a loss function and optimization algorithm. In this Thesis we use $'mean\_squared\_error'$[10] as the loss function as it closely matches Mean Squared Error that we use compare the algorithms. As an optimization algorithm we use $'adagrad'$, because of the efficiency it offers in time-series forecasting. According to Keras documentation $'adagrad'$ optimization algorithm is ideal for recurrent neural networks as the LSTM we use.

[9] Neurons, are also called memory units or blocks in similar literature.
[10] 'Mean Squared Error' is 'mse' for short.

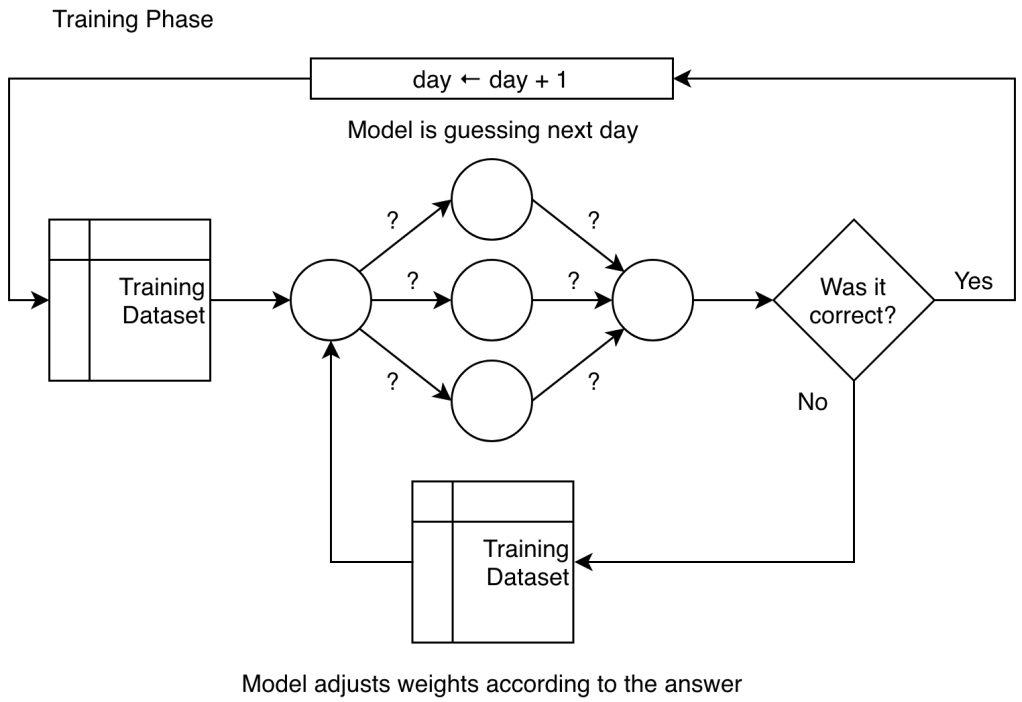


Figure 5-6: Proposed Training Phase.

Now that we finally defined our neural network, we use the Sequential Keras API to create and compile our proposed LSTM RNN analyzed in this section.

```
model = Sequential()
model.add(LSTM(neurons, batch_input_shape, stateful = True))
model.add(Dense(1))
model.compile(loss = 'mean_squared_error', optimizer = 'adagrad')
```

Manually Operated Neural Network Training

Once our proposed neural network is compiled by the Sequential Keras API, it is ready to be fit to the training data-set. Because our network is stateful, we must manually control when the internal state is reset. In order to achieve this we must manually manage the training process one epoch at a time across the desired number of epochs.

Due to Keras API the samples within an epoch are shuffled prior to being exposed to the network by default. Of course, this is undesirable for our proposed LSTM

neural network because we want it to build up state as it learns across the sequence of samples of the training data-set. So, sample shuffling must be disabled by setting $'shuffle'$ to $'False'$. Since we manually control the fitting operation we must reset the internal state at the end of the training epoch, ready for the next training iteration.

Keep in mind that increasing the number of epochs may result to overfitting. The dangers of overfitting discussed in a previous chapter in this Thesis.

The possibility of overfitting exists because the criterion used for selecting the neural network parameters is not the same as the criterion used to judge the suitability of it. For example, a specific neural network architecture might be selected by maximizing its performance on some set of training data, and yet its suitability might be determined by its ability to perform well on unseen data; then overfitting occurs when the neural network begins to 'memorize' training data rather than 'learning' to generalize from a trend.

In order to prevent overfitting in our proposed scheme, epoch number must stay low[11], depending on the data.

Also, batch size must be set to 1, because it must be a factor of the size of the training and test data-sets. This is ideal for forecasting time-series, because that way we make one-step forecasts on the test data-set.

Proposed LSTM Forecast Phase

After our proposed LSTM Neural Network is fit to the training data-set, it can be used to make accurate forecasts.

We decide to fit the Neural Network at once, on all of the training data and then predict each new time-step one at a time from the test data-set. This is called fixed approach. We also have the flexibility to fit our Neural Network dynamically, at each time-step of the test data-set as new samples from the test data-set are made available. This is called dynamic approach.

Keep in mind that dynamic approach can give more accurate forecasts, since

[11]Epoch number is usually set empirically through test or by using algorithms to guess the best parameters.

it dynamically adapts to the training data-set trend, but fixed approach can give complete forecasts faster and with considerably less computing power.

In this Thesis, we propose to follow fixed approach to reduce computational complexity and powerful hardware demand, in order to give Intelligent Agents the ability, not only to forecast, but also to fit data locally. As the embedded raw computing power increase over time and hardware prices drop, it is highly recommended to switch to dynamic approach, also analyzed in this Thesis.

We sum up the forecasting operation in a synonymous function. The function returns an array of predictions, one for each input row provided. Because we are providing a single input, the output will be a 2 dimensional array with one value.

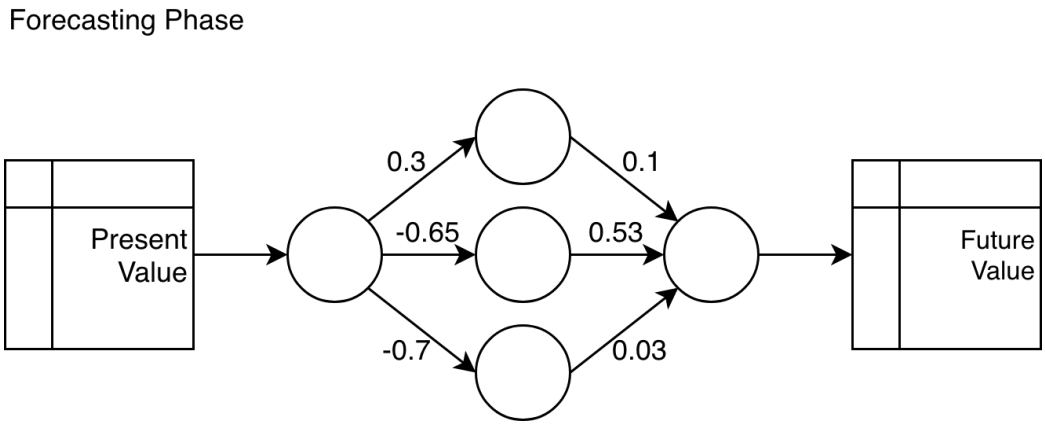


Figure 5-7: Proposed Forecasting Phase.

Given a fit neural network, a batch-size used when fitting the neural network, and a row from the test data-set, the function will separate out the input data-set from the test row, reshape it, and return the prediction as a single floating point value.

As stated in the current chapter before, during training the internal state is reset after each epoch. This is something we clearly do not want while forecasting. We do not want to reset the internal state between forecasts. In fact, we would like the model to build up state as we forecast each time-step in the test data-set.

In our proposed work, we seed the state by making a prediction on all samples in the training data-set. That way the internal state is set up ready to forecast the next time-step.

5.5 Forecast Analysis

Now that our Neural Network is complete, it is time to fit it on our data-set and evaluate its performance.

5.5.1 Evaluating Forecast Performance on Existing Data

In our proposed work, we include a tool called 'Evaluation'. Evaluation is the tool that helps the user to evaluate the accuracy performance of the Neural Network in the given data-set and parameters.

During the Evaluation, we set the Neural Network to forecast all samples from the data-set it is trained to.

Evaluation is a really important tool, that helps the user understand if the parameters given, can cause overfitting or underfitting in the data-set, that will be used for training. As stated before in this Thesis, overfitting or underfitting can may cause a large deviation from the expected forecasted values. Not only in this case, but generally it is extremely important to configure Neural Networks correctly and evaluate their performance before proceeding to real world forecasting.

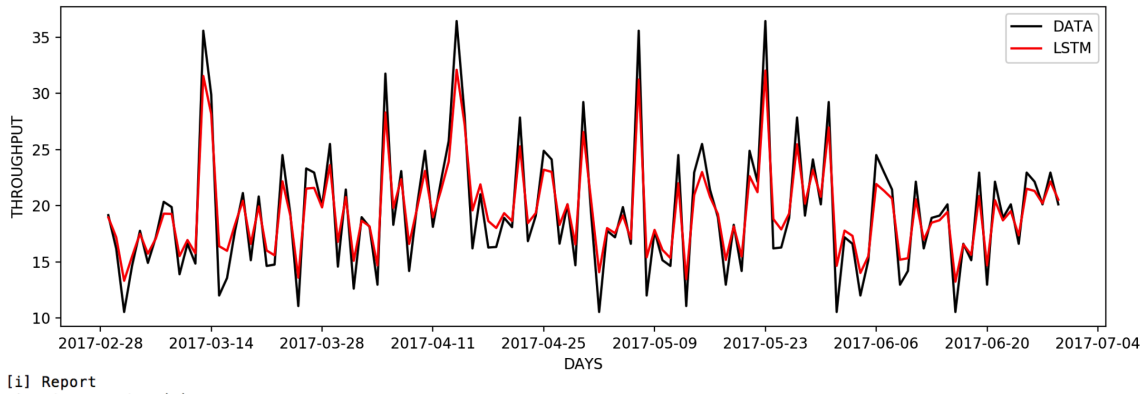


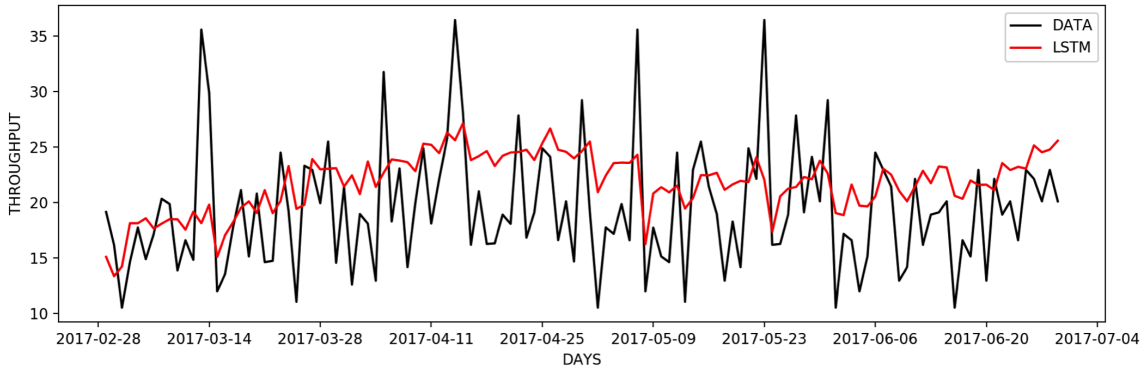
Figure 5-8: Evaluation results of a Neural Network adjusted with the optimal parameters for the testing data-set.

For our proposed Long Short-Term Memory Engine, well adjusted Neural Network parameters result to almost perfect match between trained and forecasted data-set.

Since fitting and testing data-set is the same, almost zero error during evaluation process indicates that the neural network is set-up correctly and ready to forecast in a real world scenario.

A close match is a sign of a well configured Neural Network, but still not for extremely accurate forecasting.

As stated before, over-fitting and under-fitting causes large deviation between the forecasted values and the training data-set during evaluation. This is a sign that the Neural Network, due to bad adjustment, is unable to learn or it memorizes instead of forecasting. Results also include large Mean Squared Error numbers and long fitting times.




```
[i] Report
 |- Fit Duration (s) :  125.95211434364319
 |- Prediction Duration (s) :  0.23458313941955566
 |- Total Duration (s) :  126.194748878479
 |- MSE: 31.739
```

Figure 5-9: Evaluation results of a Neural Network over-fitted to the training data-set.

The most profound way to avoid this effect, is empirically, through testing and evaluating performance.

After we complete the Neural Network performance evaluation and we are certain about using the optimal parameters, we can proceed to a real world forecasting scenario.

5.5.2 Perform Forecast on Unknown Data

Now, it is time to run a real world forecast, to draw some defining conclusions. Let's select a random cell from our data-set and forecast it's future throughout values.

For this forecast demonstration, we split 122 days of samples in half. The first 61 days will be used for training and the last 61 will be used for testing. That way, the Neural Network will forecast the later without any knowledge of their actual values. We also divide the 61 days in a specific number of data blocks, five in this specific case and select the one with the lowest Relative Standard Deviation, as stated in the previous subsection. That procedure ensure us that we train our model with the most condensed data, that are averaged around their mean value.

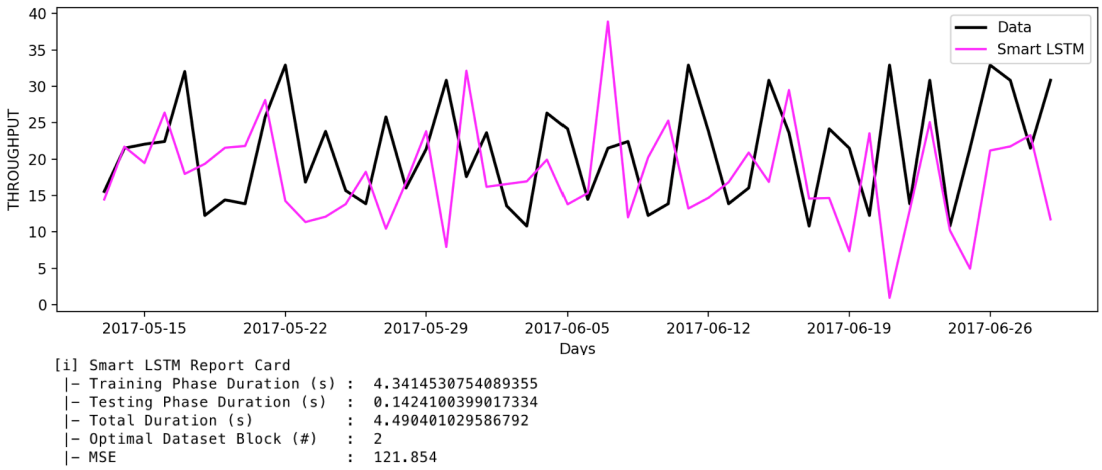
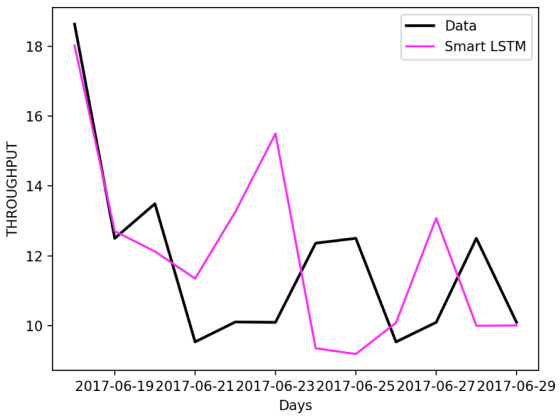


Figure 5-10: Real world forecast at a random base station sample data-set.

As we conclude from the results, the proposed forecasting mechanism continues the forecast with almost the same accuracy, even 60 days without any knowledge of new samples. Prediction duration is almost instant, especially compared to algorithms like Seasonal ARIMA or Support Vector Machine.

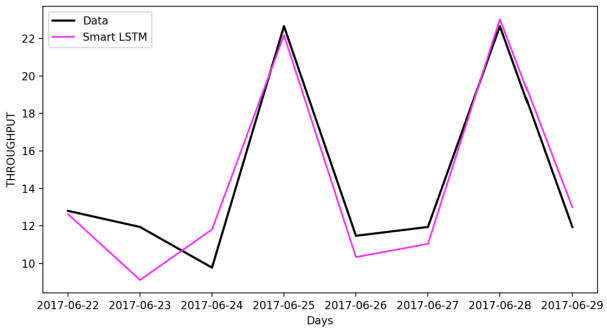
Let's focus on this specific result from another nearby base station. For this forecast we split 110 days into 6 even data blocks and chose the one with the lowest Relative Standard Deviation. With ease we can distinct that the results follows the flow of the real data and have low Mean Squared Error. It is almost impossible to predict the daily fluctuation in throughput of every base station with that data-set resolution, but we are confident that with additional samples our algorithm can reach that level of accuracy.

For this next result, we made a week long forecast of an other base station, by




```
[i] Smart LSTM Report Card
 |- Training Phase Duration (s) :  4.1948561668396
 |- Testing Phase Duration (s)  :  0.06031513214111328
 |- Total Duration (s)          :  4.259435176849365
 |- Optimal Dataset Block (#)   :  2
 |- MSE                         :  6.699
```

Figure 5-11: Brief, 11 day forecast.




```
[i] Smart LSTM Report Card
 |- Training Phase Duration (s) :  4.136817216873169
 |- Testing Phase Duration (s)  :  0.0596768856048584
 |- Total Duration (s)          :  4.20102596282959
 |- Optimal Dataset Block (#)   :  3
 |- MSE                         :  1.971
```

Figure 5-12: One week forecast.

splitting 114 days in 4 data blocks and training the neural network with the one that has the lowest Relative Standard Deviation. This forecast looks almost identical to the real data and the Mean Squared Error is minimal. As we stated in previous sections of this Thesis, using longer training data-set can return more accurate results.

Those last results are from two entirely different base stations, but yet share enough similarities. They are week long like the previous forecasts. Despite the fact that the left forecast belongs to a base station located in a more popular location than the right one, both have accurate results, but failed to predict a sudden transition. As previously mentioned, longer data-set and higher sample rate can train the neural network to predict more spikes, but there are more factors that compose this behavior.

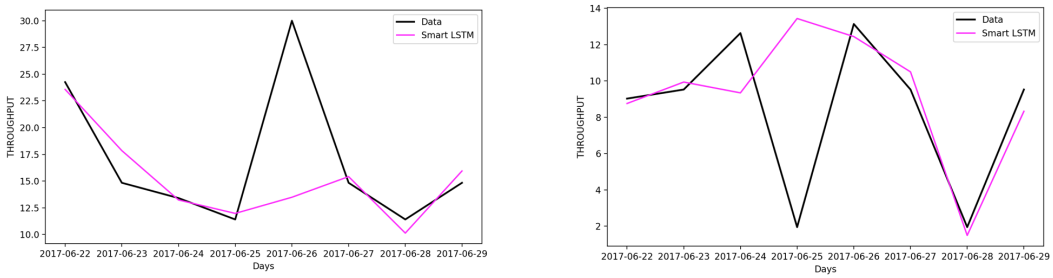


Figure 5-13: Similar forecast behavior, from different base stations.

Using an advanced spike filter or disabling it completely can help identifying upcoming unexpected transitions that appear periodically. Keep in mind that in some cases this can increase the Mean Squared Error of the forecast.

Improve Forecasting Accuracy

There are several ways to improve forecasting accuracy. The best way to drastically improve accuracy is to increase the number of samples in our data-set. Huge amount of training data can improve accuracy, but need more time to fit the neural network. Choosing an alternative pre-processing or filtering algorithm can also give better results. Training data-set pre-processing is important because gaps in samples or spikes in signal may confuse Neural Network weighting system. Another way, also mentioned before, is to use the Evaluation tool and re-adjust Neural Network parameters, to make sure that there is no over-fitting or under-fitting effects.

A more common scenario for forecasting application is just predicting the next value of our metrics. At this point, it is important to note that this functionality is build-in our proposed forecasting mechanism.

Benchmark Statistics

Since we use Mean Squared Error to calculate the deviation between forecasted and real values, we can compare the accuracy of all algorithms against the proposed.

The results show that the Seasonal ARIMA provide less accurate results in less time than ARIMA. Radial Basis Function of Support Vector Machine can give more

```
[i] Benchmark Report
 |- RBF     MSE: 10.1108263265 @ 7.22363805770874 (s)
 |- ARIMA   MSE: 15.2830136025 @ 21.44947600364685 (s)
 |- SARIMAX MSE: 22.796940058 @ 8.962857961654663 (s)
 |- LSTM    MSE: 5.72620053873 @ 0.23639702796936035 (s)
```

Figure 5-14: Benchmark of a real world forecasting scenario between LSTM and competing algorithms.

accurate results in less time than Seasonal and Linear ARIMA. Our proposed LSTM RNN architecture can easily outperform all of the competing forecasting algorithms in both accuracy and execution time.

Since forecasting can produce accurate guesses for long periods of time and data-set training can be performed in specific hardware in parallel, the mechanism proposed in this Thesis remains the best option.

5.6 Summary

We can clearly conclude this chapter by acknowledging the superiority of our proposed Long Short-Term Memory Recurrent Neural Network against similar forecasting techniques.

According the results, not only from this Thesis, but also from similar literature [15] [30] [23], most of the commonly used linear forecasting algorithms fail to adapt to the testing data-set. Linear time-series forecasting algorithms can mostly extract the possible trend from the given data-set, but cannot identify patterns and make educated guesses about future values.

Due to the rich multifractal behavior of the cellular subscriber Internet traffic activity [12], Long Short-Term Memory Recurrent Neural Networks can easily outperform linear algorithms. Depending their architecture they can be used to forecast time-series or signal values, given time lags of unknown size and duration.

A big advantage of the proposed Long-Short Term Memory Recurrent Neural Network offer in modern cellular network architectures, is that part of their computation can be offloaded to a centralized system. Since training can be performed by a centralized system, all base stations can perform predictions in simple embedded

hardware, allowing service providers the option to completely automate strategic resource planning in base stations themselves. That offers an energy, resource and cost efficient way for service providers to address the problem of resource planning and allocation.

Chapter 6

Cellular Network Architecture

In order to improve the resource management of cellular networks, traffic modeling and prediction has been focused in recent years [10].

6.1 Network Architecture Overview

In this section, we go through some basic concepts of the existing cellular network architecture. The typical architecture of a GSM/UMTS standard hybrid cellular network, consists of an Access Network[1] and a Core Network[2] [27]. Access Network is the edge part of a telecommunications network which connects subscribers to their service provider, and Core Network is the central part of the network that provides services to subscribers who are connected by Access Network.

In Access Network, subscriber uses a Mobile Station[3] to connect to Base Transceiver Station[4], which can be identified by a unique number, the cell ID. In Core Network, MSC[5] processes the voice and text demand in Circuit Switch[6] domain, as well as

[1] Access network is AN for short.
[2] CN also means core network.
[3] A Mobile Station, ST for short is considered any type of end user device with cellular connectivity.
[4] Also called BTS or NodeB. Usually mentioned as base stations.
[5] 'MSC' stands for 'Network Switching Subsystem'.
[6] Circuit Switch is pictured as CS.

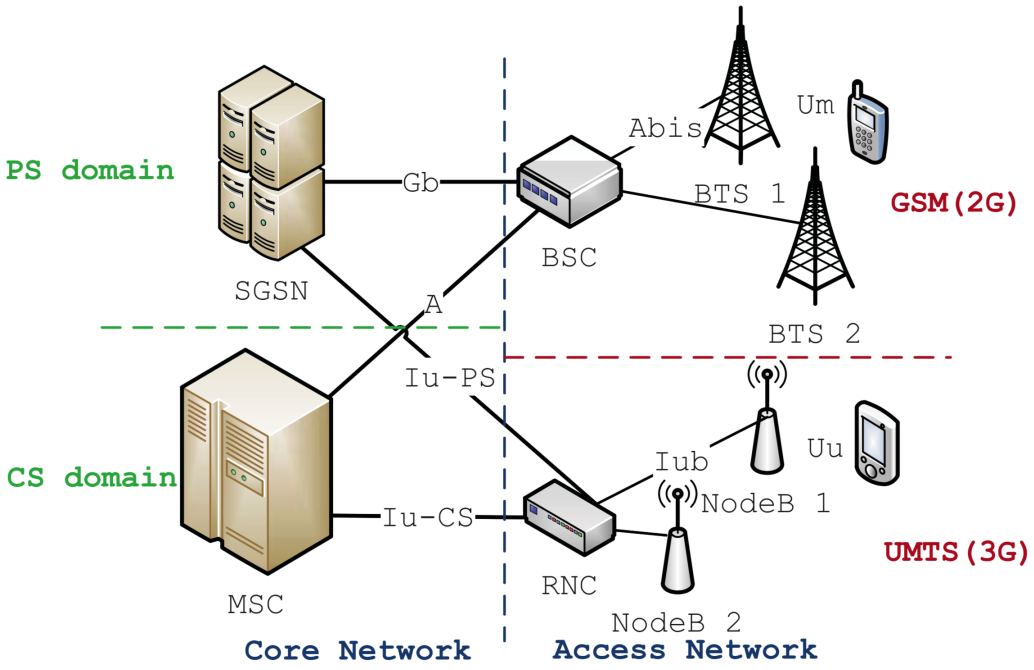


Figure 6-1: Typical architecture of a GSM/UMTS hybrid cellular network.

Serving GPRS Support Node[7] processes the data demand in Packet Switch[8] domain. Access Network has complex forms of connections and is adjusted frequently, therefore it is difficult to monitor every Base Station [27].

In this Thesis, we use Signaling Monitor System in Core Network, as well as most of the Base Station metrics, to get traffic information for forecasting.

6.2 Traffic-Aware Network Architecture

Despite the fact that research on traffic prediction is an established field, most existing works [11] have been carried out on traditional wired broadband networks and rarely shed light on cellular radio access networks[9].

However, with the explosively growing demand for radio access, there is an urgent need to design a traffic-aware energy-efficient network architecture [19] [17] [18].

[7]Serving GPRS Support Node is also called SGSN
[8]Packet Switch is also called PS for short.
[9]Cellular Radio Access Network is also known in literature as CRAN in similar literature.

The rebuilding of a traffic-aware energy-efficient architecture for cellular networks is becoming a trend [1] [7].

6.2.1 The TANGO Project

With traffic forecasting, it is possible for cellular networks to be configured and managed more efficiently [10]. In similar literature, Niu et al. [17], advocated establishing traffic-aware energy-efficient radio access networks, or the so-called TANGO[10]. One of the key principles in TANGO is to make the working status of network elements, mainly Base Stations, to be adaptively adjusted according to the traffic pattern. Specifically, some Base Station or elements of Base Stations can be tuned into sleeping mode to save energy when the predicted traffic is negligible [19], while other Base Stations expand their coverage in a coordinated manner [17].

6.2.2 Long-Term Solution

Owing to the flexibility of resource allocation and its considerable agility to meet explosively increasing traffic demands [27], Traffic-aware Networks would be the most potentially suitable future cellular architecture [17] [18], in which traffic prediction acts as one of the dominant factors for on-demand network management [10].

6.3 Centralized System for Prediction

The main concept behind using Neural Networks for subscriber generated traffic prediction, is a centralized overlying system over the current cellular network architecture.

Consider an Intelligent Agent located at the side of every Base Station, that is responsible for monitoring the Base Station environment and storing all the necessary data[11], which will be subsequently used for the training process of the Neural Network.

[10] TANGO stands for Traffic-aware Network Planning and Green Operation.
[11] All measured data should be time-stamped network traffic measurements

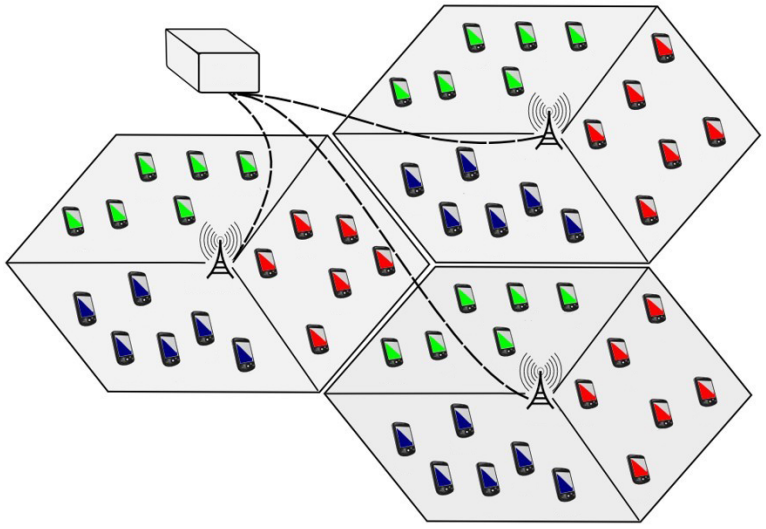


Figure 6-2: Overview of the proposed centralized system.

Then, based on the prediction model, the Intelligent Agent exchange sections of it's extracted measurements to a centralized system, a server, for the training phase and gets back the trained model. Intelligent Agent data exchange should take place only when the network is underutilized, most likely at dawn. At times where, due to congestion or malfunction, Intelligent Agent could perform training phase itself, but with the trade-off lower accuracy. That architecture allows for fast training and faster prediction, by offloading the resource demanding task of training phase to proper hardware and leaves forecasting phase to the energy efficient Intelligent Agent of the base station. Base stations can become completely autonomous as the Intelligent Agent estimates the forthcoming needs for resources and proactively requests their commitment from the back-haul network.

6.4 Implementation in Current Infrastructure

Since this proposed architecture is presented as an overlying technology, there is no need to alter the structure of the existing network in order to implement such system.

With just a simple set of hardware, this system can be set up and running. The hardware needed for the proposed centralized system, consists of the following:

1. Intelligent Agents are small network devices, that collect measurements and execute the testing phase[12] of neural network. They use the back-haul network to send collected values to the Service Provider for the training phase[13] and get back the trained model for instantaneous and energy efficient forecasting.

2. Centralized Intelligent System is a server, spatially located near Service Provider and is used solely for the training phase of Neural Network. Powerful and efficient, GPU or CPU, hardware can be used to fit the Neural Network of every Intelligent Agent quickly and transfer the trained model back to them for the next phase.

No further devices are required to implement this proposed architecture. In case of failure, Intelligent Agents are ready to take the situation in their hands and make decisions autonomously. Transition to that model is fast, efficient and most of all safe with respect to the current network, things that make it highly recommended.

[12]Testing phase and forecasting phase will be both be used in this Thesis.
[13]Training phase is also known as fitting phase.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 7

Concluding Comments

7.1 Future Work

It is desirable to investigate how to leverage the additional information to further optimize the proposed work and improve the prediction accuracy.

We intend to conduct another, but this time long term forecast, that the results will be from a period more than one year. Such an experiment, will give us an idea of the extreme exploitation of the proposed Long Short-Term Memory Recurrent Neural Network architecture.

Another significant task, is to take into account seasonality. Exploiting seasonality in extreme detail, may give an even more accurate insight in future traffic predictions. Complete seasonality model separation between workdays and weekends, can make it easier for neural network to distinct and learn human mobility patterns. The same concept should be applied for daily seasonality, in order to explore every possible combination that returns higher accuracy.

Understanding and forecasting cellular network traffic data, can have a dominant impact on the operation and Quality of Service[1] provided to the users every single day and this sets a direction for exploring further this issue.

[1] Quality of service, or QoS for short, is the description or measurement of the overall performance of a service.

7.2 Summary

In this work we focused on developing a Traffic-Aware, Subscriber Internet generated traffic forecasting mechanism, based on Long-Sort Term Memory Recurrent Neural Networks, that can be implemented on the current cellular network architecture.

The proposed Neural Network mechanism is capable to recognize typical patterns and the overall trend for each base station, by using embedded computing network devices called Intelligent Agents. Since it is a centralized system, it can offload the resource demanding task of training to proper hardware and leaves the forecasting to the energy efficient Intelligent Agents of the base stations.

The idea is to implement this scheme in order to give the ability to the network to configure itself and to propose intelligent solutions based on future traffic demands.

We believe that the contributions given on this work can be used to offer a solution for service providers to enhance cellular network performance, by utilizing effectively all available resources with smart strategic planning, that uses predictions by the proposed mechanism of this Thesis.

Bibliography

[1] Abdelnaser Adas. Traffic models in broadband networks. 35:82 – 89, 08 1997.

[2] Vicente Alarcon-Aquino and Javier A. Barria. Multiresolution fir neural-network-based learning algorithm applied to network traffic prediction. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICSPART C*, 36(2):13, March 2006.

[3] N. C. Anand, C. Scoglio, and B. Natarajan. Garch x2014; non-linear time series model for traffic modeling and prediction. In *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*, pages 694–697, April 2008.

[4] M. Chuah and W. Luo. Impacts of inactivity timer values on umts system capacity. *IEEE Wireless Communications and Networking Conference*, page 897903, 2002.

[5] J. Bolot D. Willkomm, S. Machiraju and A. Wolisz. Primary users in cellular networks: A large-scale measurement study. *IEEE DySPAN*, 2008.

[6] Xiaofeng Zhong Zhisheng Niu Xuan Zhou Honggang Zhang Dongheon Lee, Sheng Zhou. Spatial modeling of the traffic density in cellular networks. *IEEE Wireless Communications*, 21(1):88, February 2014.

[7] Y. Zhang et al. Spatio-temporal compressive sensing and internet traffic matrices. *Proc. ACM SIGCOMM*, August 2008. Barcelona, Catalonia.

[8] Emir Halepovic and Carey L. Williamson. Characterizing and modeling user mobility in a cellular data network, 01 2005.

[9] Manish Joshi and TheyaznTheyazn Aldhayni. A review of network traffic analysis and prediction techniques. 07 2015.

[10] Hyojoon Kim and Nick Feamster. Improving network management with software defined networking. 51:114–119, 02 2013.

[11] Rongpeng Li, Zhao Zhifeng, Xuan Zhou, Jacques Palicot, and Honggang Zhang. The prediction analysis of cellular radio access network traffic: From entropy theory to networking practice. 52:234–240, 06 2014.

[12] Jian Liu. *Fractal Network Traffic Analysis with Applications*. PhD thesis, School of Electrical and Computer Engineering at Georgia Institute of Technology, 2006.

[13] Ioannis Loumiotis, Evgenia Adamopoulou, Konstantinos Demestichas, Theodora Stamatiadi, and M Theologou. On the predictability of next generation mobile network traffic using artificial neural networks. 28, 12 2013.

[14] C. A. Hidalgo M. C. Gonzalez and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453:779782, 2008.

[15] Andrei B. Rus Virgil Dobrota Melinda Barabas, Georgeta Boanea. Evaluation of network traffic prediction based on neural networks with multi-task learning and multiresolution decomposition.

[16] Rtsch B. Schlkopf J. Kolmorgen Mller, A. J. Smola and V. Vapnik. Using support vector machines for time series prediction.

[17] Z. Niu. Tango: Traffic-aware network planning and green operation. *IEEE Wireless Communications*, 18(5):25, October 2011.

[18] Eunsung Oh and Bhaskar Krishnamachari. Energy savings through dynamic base station switching in cellular wireless access networks, 12 2010.

[19] Eunsung Oh, Bhaskar Krishnamachari, Xin Liu, and Zhisheng Niu. Toward dynamic energy-efficient operation of cellular network infrastructure. 49:56–61, 06 2011.

[20] Pubudu Pathirana, Andrey Savkin, and S Jha. Location estimation and trajectory prediction for cellular networks with mobile base stations. 53:1903 – 1913, 12 2004.

[21] Utpal Paul. Understanding traffic dynamics in cellular data networks.

[22] Meng Qing-Fang, Chen Yue-Hui, and Peng Yu-Hua. Small-time scale network traffic prediction based on a local support vector machine regression model. 18:2194, 06 2009.

[23] A S. Lapedes and Robert Farber. Nonlinear signal processing using neural networks: Prediction and system modelling, 06 1987.

[24] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-Laszlo Barabasi. Limits of predictability in human mobility. 327:1018–21, 02 2010.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. 15:1929–1958, 06 2014.

[26] G Thimm, P Moerland, and E Fiesler. The interchangeability of learning rate and gain in backpropagation neural networks. 8:451–60, 03 1996.

[27] Rongpeng Li Yifan Zhou Honggang Zhang Xuan Zhou, Zhifeng Zhao. The predictability of cellular networks traffic. *2012 International Symposium on Communications and Information Technologies (ISCIT)*, page 6, 2012.

[28] Lianfang Zhang Lei Wang Yentai Shu, Zhigang Jin. Traffic prediction using farima models.

[29] Y. Yu, M. Song, Y. Fu, and J. Song. Traffic prediction in 3g mobile networks based on multifractal exploration. *Tsinghua Science and Technology*, 18(4):398–405, August 2013.

[30] Jia-Shu Zhang and Xian-Ci Xiao. Predicting chaotic time series using recurrent neural network. 17:88, 08 2008.

[31] Bo Zhou, Z Sun, and Wee Hock Ng. Network traffic modeling and prediction with arima/garch. 07 2008.

[32] M Zubair Shafiq, Lusheng Ji, Alex X. Liu, Jeffrey Pang, and Jia Wang. Characterizing geospatial dynamics of application usage in a 3g cellular data network. pages 1341–1349, 03 2012.