Tampere University

Joonas Toimela

Spatio-Temporal Prediction for Mobile Network Quality Analysis

Faculty of Information Technology and Communication Sciences (ITC)
Master's thesis
October 27, 2020

Abstract

Joonas Toimela: Spatio-Temporal Prediction for Mobile Network Quality Analysis
Master's thesis
Tampere University
Master's Degree Programme in Computational Big Data Analytics
October 27, 2020

Mobile networks are under constant change and require frequent optimization and replanning. Towards this goal there are multiple tools however, the data coming from the user equipment is underutilized for this purpose. In this thesis the measurements collected from user devices are exploited to create spatio-temporal maps, following ideas coming from radio environment maps literature, that can be used for optimization and planning. The thesis studies different ways of creating such maps and predictions and extends them to forecasting in the temporal domain. The prediction results are visualized so that network operators can use them for optimization and planning. Locally approximate Gaussian process regression is a novel approach in this context and it is studied along with k-nearest neighbour interpolation, fixed rank kriging and a neural network based solution. The novel approach is among the best performing overall.

Keywords: Radio environment map, mobile network, big data.

Tiivistelmä

Joonas Toimela: Ajallinen ja tilallinen ennustaminen langattomien tietoverkkojen laadun analysoimiseksi
Pro gradu -tutkielma
Tampereen yliopisto
Master's Degree Programme in Computational Big Data Analytics
27. lokakuuta 2020

Mobiiliverkojen ympäristö on jatkuvassa muutoksessa ja mobiiliverkot vaativat jatkuvaa optimointia ja uudelleen suunnittelua. Vaikka tähän tarkoitukseen on olemassa monia työkaluja on käyttäjien laittesta saatava tieto edelleen heikosti hyödynnettyä. Tässä tutkielmassa käyttäjien laitteista kerättyä tietoa käytetään ajallisten ja tilallisten ennusteiden luomiseen, tavoitteena on luoda karttoja joita voitaisiin hyödyntää langattomien verkkojen optimoinnissa ja suunnittelussa. Tässä tutkielmassa tutkitaan erillaisia tapoja luoda kyseisenlaisia karttoja seuraten radioympäristökartta kirjallisuutta. Ennustuksen lopputulos visualisoidaan niin että tuloksia on mahdollista käyttää verkon uudistamiseen. Paikallisesti approksimoituun Gaussiseen prosessiin perustuva malli on tähän käyttötarkoitukseen uusi ja sitä testataan k:n lähimmän naapurin interpolaation, fixed rank kriging menetelmän sekä neuroverkkoihin perustuvan menetelmän ohella. Uuden menetelmän tuottamat tulokset ovat parhaimpien joukossa kaikissa testatuissa tapauksissa.

Keywords: radioympäristökartta, langattomat verkot, suuret tietoaineistot.

Contents

1 Introduction

The behaviour of a mobile network is in constant change, requiring the network operator to continuously do changes to network parameters and sometimes even to the structure of the network. Network operators have a wide variety of different tools at their disposal to assist them in doing informed decision about the best possible changes to the network.[48, Chapter 2]. The aim of this thesis is to find new ways to analyse the network quality by leveraging signal quality measurement data collected from the user equipment (UE).

The behaviour of the mobile network can change due to various reasons. Some of the reasons are permanent such as a new building that obstructs the signals, which is a very common reason in urban environments. Some other reasons for change are more temporal. For example, a concert that gathers large crowd on a small area or heavy rainfall that changes the signal propagation on a large area by a small amount. All the phenomena to consider are not external to the network either, sometimes a drastic change in signal quality or coverage can be caused by a base station (eNodeB) that is malfunctioning for one reason or another. All of these changes should be detectable and identifiable by the operator so that the operator can take appropriate action.

When a problem or poorly performing area in the network is detected the operator has many ways to mitigate the effect on end-users. The fastest mitigation actions to deploy are changing parameters for the eNodeB, such as electrical down tilt and transmit power. However, changing tilt or power will also affect interference in neighbouring cells so even the fastest actions might not be straightforward. For more permanent problems the affected area can be re-planned, which might mean adding new eNodeBs, changing the location of old ones, deploying small cells or just more parameter tuning.[48, Chapter 3]. The analysis done on the network should help in choosing the optimal parameters to change and make it easy to detect areas that require re-planning.

If the mobile network is left unoptimized or problems are not acted upon fast enough, the connectivity of the users will deteriorate. Some common symptoms for the user are; dropped connection, low bandwidth, high latency and shorter battery life. In the 5G era, acting fast to solve network problems only becomes more important as many parts of society's critical infrastructure are proposed to become dependant on wireless connectivity. But it is not just angry customers that will affect the communication service provider (CSP), badly optimized mobile network will also use more power, increasing operational expenses. Furthermore, the available spectrum is a limited resource and a badly optimized network is wasting a

lot of spectrum that could otherwise be used more productively. Thus, detecting network problems and affected areas in a timely manner is a matter of user satisfaction, operational costs and efficiency and it is becoming also a matter of public safety. Optimally network degradation could be predicted and the operator could act proactively.

Thus far, I have talked about operators actions as if it was a person making the actions, but increasingly that is not the case. As CSPs try to reduce the expenses for managing the network and the network becoming more and more complex at the same time, an automated solution is required. Thus the idea of self-organizing networks (SON) is born to help manage LTE networks. SON can automatically configure newly added eNodeBs (self-configuration), find optimal parameters for the current situation (self-optimization) and mitigate the effect of eNodeB malfunction (self-healing)[12]. For this study, especially the self-optimization and self-healing functionality are of interest. Many machine-learning approaches have been suggested towards self-optimization and self-healing[4, 3], but many of these approaches are geared towards acting fast to react to ongoing change and would be of limited value in characterizing the network for example in re-planning context. For the analysis to be done in this thesis it would be beneficial if its output could be used by both human operators and automated processes.

Things introduced in past four paragraphs require a lot more background information to create a proper base for building this thesis on and consequently they are discussed more thoroughly in section 2.1. The rest of the introduction focuses more on what is required from the analysis methods that are to be used, with the initial starting point discussed in depth in section 2.2.

Combining the requirements from earlier, detecting changes in network behaviour in different time scales as early as possible and usable by human and machine operators, there is one approach in the literature that rises above others: radio environment maps. While radio environment maps (REM) incorporate many different aspects that affect operating a radio, such as regulation and geography, the most interesting for this thesis are the spectrum sensing, spectrum situation generation and visualization aspects of it[21]. While REM is not a new idea it has gained more attention recently in the context of cellular networks as getting a large number of spectrum measurements from mobile networks has become easier [44]. There is plenty of literature of constructing REMs from different kind of radio frequency measurements, but as far as I can tell none of them is directly applicable to the proposed use case.

As mentioned in the previous chapter acquiring spectrum measurement has become easier recently with a new mobile network feature called minimization of drive testing (MDT) [44]. MDT allows any user equipment to work as a measurement de-

vice that sends the measurements back to the network every few seconds. It is easy to see that even in a medium-sized city (population between 100 000 and 0.5 million) using the full capability of MDT would give the CSP billions of measurements each day. This capability is limited by the fact that not all user devices support MDT and GPS, the estimated percentage of devices supporting both together being around 3% in 2019 [36]. Further reduction in incoming data can be achieved by limiting the measurement area or frequency. Even with these limitations in place, it is easily possible to gather millions of measurements each day. This is necessary for building high-quality radio environment maps as discussed in an earlier chapter, however, large amounts of data also pose new challenges especially in the form of increased computational time. Many classical statistical tools start to become unusable when applied on millions of measurements, not to mention billions. For the analysis used in this thesis to be also usable in future, it must have very low computational complexity or very good scalability to take advantage of increasing computing resources. For this reason, computational complexity and scalability are discussed constantly along with the techniques to be used.

The above considerations regarding network operators workflow, radio environment maps and problems of big data lead to three research questions for this thesis:

1. How to create accurate spatial-temporal maps of network quality measurements based on large amounts of past observations.

2. How these maps can be extended to forecasting in the temporal domain to facilitate proactive optimization, and

3. What kind of visualizations and other methods make detecting network degradation easier for human and machine operators alike.

2 Theoretical Background

This chapter is devoted to making a more in-depth introduction of mobile networks, the methods to be used and literature that have a similar aim or is otherwise relevant for the thesis.

This chapter starts with an introduction to modern mobile networks to give the reader a better idea of where the measurements are coming from, what exactly are we measuring and why this is a worthwhile topic to study. Additionally, I want to introduce many terms and abbreviations that I am going to use throughout the thesis.

After a short lecture on mobile networks, the chapter continues to discuss radio environment maps that were established as a relevant concept already in the introduction chapter. In this section I will present different REM construction methods from previous literature, the section will also motivate a secondary use case for the obtained results.

Finally, there is a section that describes individual prediction methods used to obtain the thesis results. Many of the methods or their variant are already motivated in the earlier section so this section focuses on highlighting the differences to the methods already used in the literature and presenting the methods in a very detailed manner.

2.1 Mobile Networks

Mobile networks or cellular networks were first launched about 40 years ago, since then the networks have seen three new generations of network standards (2G, 3G, 4G/LTE) and are about to see the fourth (5G). During these generations, the mobile networks moved from analogue signals to digital ones, from circuit switching to packet switching and from 2.4 kilobytes per second to multiple gigabits per second maximum data rates. Each new generation has also made the networks more complex by adding new parts that are required for running the network. For the purpose of this thesis, long term evolution (LTE) and 5G networks are similar enough that I do not need to make a distinction between them but the structure subsection is based on the LTE network since that is the more mature technology of the two.

The first subsection is going to discuss the structure of the network to the minimum extent that is required to understand the thesis and to introduce the terms and abbreviations used elsewhere in the thesis. The second subsection introduces key performance indicators that are used to monitor the state and performance of the network, also some indicators are presented that are not directly used in the thesis

but that are required to discuss the possibilities and limitations of the proposed method. In the final subsection the usual workflow of operators (human and machine) for troubleshooting network problems is outlined so that the possible impact of the thesis can be discussed.

2.1.1 Structure

Mobile network architecture is commonly split to two main parts namely RAN and core, where RAN stands for radio access network and consists of the antennas, base stations and radio network controller and core consist of ”everything else” that is required for the network to function such as mobility management, user authentication and packet switching. For the thesis, only the RAN part is of interest.

For the planning of the mobile network, the area is divided into a mesh of cells. While the cells in the mesh can have different shapes and sizes, the simplest case is equally sized cells in a hexagonal grid as shown in figure 2.1. Valid parameters are then found for each neighbourhood of cells at a time. When choosing the parameters the operator needs to ensure that the whole cell is covered by the signal but the signal does not interfere with the neighbouring cells. The interference is especially problematic in LTE networks as each cell can (and usually does) use the same frequency unlike in earlier generations. The cell sizes can also change during operation of the network to offload some traffic from the congested cell or to compensate for a failed node such as in self-healing procedure of self-organizing networks. When the cell size changes during operation the same considerations regarding coverage and interference need to be made. The cell size affects how dense the network is and ultimately how many users it can serve. To not make the planning too easy, there can also be smaller cells (small cell) inside larger cells (macro cell), creating a layered structure referred to as heterogeneous network (HetNet)[29, Section 2.6.3.8].

In LTE networks the radio network controller is built into the base station and together it is called evolved nodeB or eNodeB or eNB for short [29, section 3.3.3]. One eNodeB can have multiple antennas and thus it can serve multiple cells, the technical limit being 255 cells but 3, 6 and 9 being the common ones. For the hexagonal cell grid structure shown in figure 2.1 the usual solution would be three cells for each eNodeB, so the eNodeB would be placed in the corner of the hexagon and it would serve the three neighbouring cells.

Different frequency bands do not interfere, so each frequency band is optimized separately even if the coverage areas overlap, it also means that each frequency band can use a different grid of cells for the planning. Different frequency bands are also affected differently by obstacles, the lower end of the spectrum usually penetrating walls better and higher end worse, highest frequencies used in 5G (mmWave) being effectively blocked by even light obstacles. While this means that more antennas are

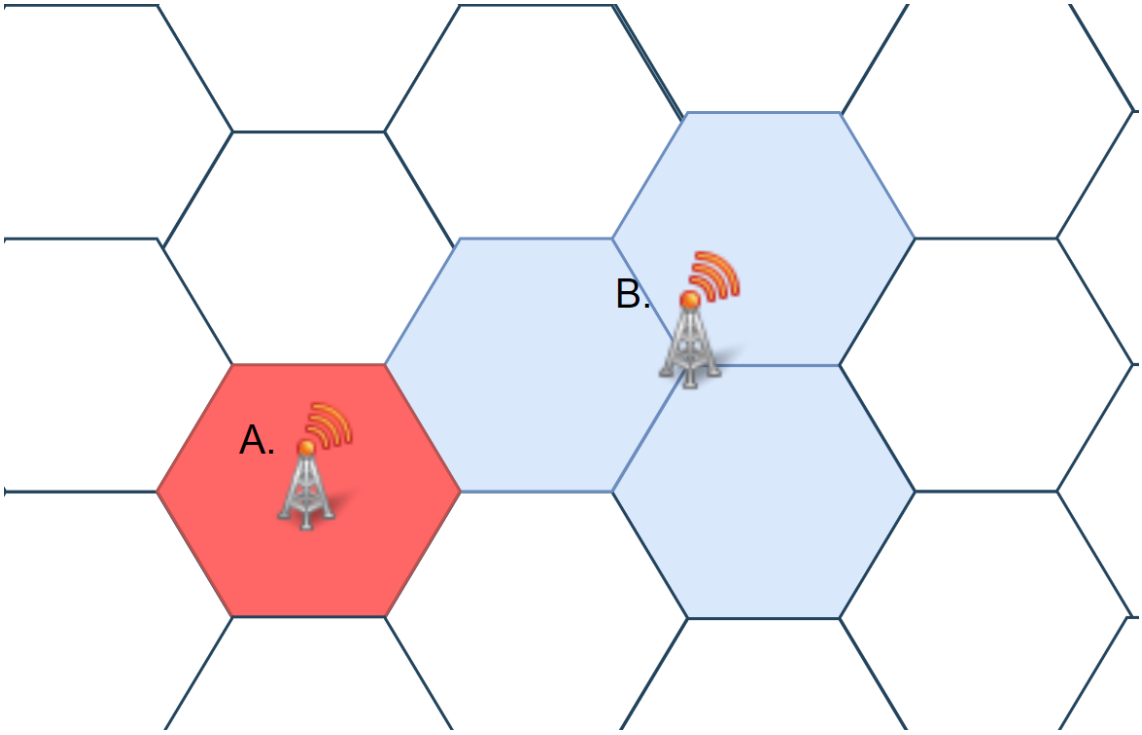

Figure 2.1 *Shows hexagonal grid of equally sized cells. The cell denoted by a red background and a letter A corresponds to the case where the eNodeB is placed in the middle of the cell and the eNodeB is serving only one cell. The cells with a blue background and a letter B corresponds to the case where eNodeB is placed in the corner of three cells and it is serving them all.*

needed for the same coverage and that the coverage can be severely affected even by small changes in the environment, it also means that each room in a building could be a different cell with its own antenna without massive interference problems allowing very high density for the network.[35] This kind of planning would require very high resolution knowledge of the signal propagation to make sure that no interfering areas are left. This knowledge is commonly obtained with simulators in the planning phase and drive testing in the operation phase.

When moving around in the network the user equipment (UE) needs to know when to change from listening one antenna to listening to another one, so effectively when to move from one cell to another. For this purpose, each cell has its own reference signal, a signal which is different for each cell (each physical cell id) and has known content. From this reference signal, the UE can calculate the power of each cell around it and pick the one with the best connectivity. As the reference signal content is pre-defined and known by the receiver, it is also possible to calculate effects such as noise and phase-shift from the reference signal.

To align the message timings correctly in eNodeB from multiple UEs the eNodeB and UEs need to have a rough idea of how long the message will be in the air when doing the transmission, this time is called timing advance and it can be used

to calculate how far away the UE is from the eNodeB. However, sometimes the signal can bounce from walls/water (called multipath propagation) and the distance observed by calculating from timing advance does not agree with distance computed from GPS. When the signal bounces it can cause destructive interference in the signal at the receiving end, at the same time modern multiple-input and multiple-output (MIMO) antennas are designed to take advantage of multipath propagation by sending and receiving multiple signals at once. It would be beneficial to be able to find areas that are problematic for older technology and beneficial for MIMO antennas, some of these areas could be identified by comparing timing advance distances with the distance given by GPS.

2.1.2 Key Performance Indicators

Reference signal received strength, RSRP for short, is arguably the most used metric when studying mobile network coverage. There are a plethora of studies [2, 41] only focused on predicting and characterizing the distribution of RSRP without even considering other coverage measures. RSRP can be measured by the UE and it is common to obtain these measurements with drive testing. RSRP is defined to be the average power of resource elements that carry the reference signal [11]. The observable range of values is from -44 dBm to -140 dBm, more than -80 dBm is considered very good and less than -100 dBm is considered bad. Originally RSRP values are used to determine when to do handover of the UE from one cell to another[1]. The value at a certain point is not expected to change depending on the number of users, but obstacles like walls that obstruct the signal do affect it.

RSSI or received signal strength indicator is used to compute RSRQ. Unlike RSRP which counts the average power for one cell, RSSI counts average received power including interference from other cells and channels. RSSI is not commonly used as is but through RSRQ instead. RSRQ or reference signal received quality is used for handover decision when RSRP alone is not enough to do the decision[1]. RSRQ is calculated as $N * RSRP/RSSI$ where N is the number of resource blocks used to determine RSSI. RSRQ is a metric that combines signal strength and interference. As interference is also included the RSRQ is affected by network usage of other users in the area as well as poor optimization of neighbouring cells.

Signal to noise + interference ratio, SINR, tells how well the signal from the current cell can be distinguished from the interference caused by other cells and random noise. SINR is sent back to the eNodeB by the UE where it is used to determine optimal modulation and coding scheme, as such it strongly determines the throughput available to the UE.[1].

UE power headroom measurement tells how much more power the UE can spend to do the transmission if the cell allows, the value is calculated as $headroom =$

$UE\_nominal\_max\_power - Estimated\_power\_needed$. If the estimated power exceeds the nominal power of the UE, the reported number can also be negative. The power headroom value can be used along with the SINR to decide on optimal modulation and coding scheme and to select how many resource blocks get allocated to the UE.[29, chapter 5.9.3.3]

The KPIs above are based on measuring the power and quality of the radio frequencies, however, these are an only small part of KPIs that the operator has access to. Next, I am going to introduce some KPIs that are commonly used, but which are only measurable from network side not from individual UEs. These metrics can be used along with the UE measurements above to provide the operator with a more complete picture of the network state.

Handovers from one cell to another can be non-optimal in more than one way, the handover can fail altogether or the handover can result in ping-pong handovers where the UE is connected back and forth between two cells. The failed handovers are being watched by handover success rate KPI, which can be aggregated on a cell or eNodeB level while the ping-pong handovers can be detected as an unusually high number of handovers. On top of these two ways the handover can happen too early or too late causing the UE to be at the cell edge which decreases throughput and increases UE power usage.[48, Chapter 8].

Another commonly used KPI is call drop rate. Dropped calls are commonly caused by bad coverage, high interference or network error. Call drop rates are being actively followed and they can be aggregated in a similar manner as handover failure rate.[48, Chapter 6].

2.1.3 Workflow of the Operators

The role of the operators is to monitor the performance of the network. Usually, the monitoring is done based on the KPIs available from the network side and the UE measurements are only used for planning new changes. It is common for the network operator to choose ten worst cells based on some KPI, like the call drop rate introduced in the previous chapter, and then take a closer look to all the KPIs of the worst cells. Then combining the different KPIs and experience to detect what could be causing that cell to have worse performance than the others. Once a possible reason is found the cell configuration can be changed, a technician is ordered, or the cell can be marked for re-planning. [29, Section 10.3.2].

Maybe more commonly the operator does not have the option to pick the KPI to optimize but instead, the KPI and the problematic cell is given as an alert. An alert is created when some cell's performance is worse than a predefined limit on some KPI. Operators can get thousands of such alerts daily, so having time to optimize the network before it starts to create alerts may not always be feasible. After receiving

alert the troubleshooting is pretty similar than in the proactive optimization case.

Even more alerts could be sent by lowering the limits of the alert sending, and arguably this should keep the network in better condition, however at some point there just are not enough operators to deal with all the alerts. But some alerts do not need a very thorough investigation to find the reason, in these cases even an automatic procedure could do the job. Thus, along with LTE networks a new feature that could assist the operators in basic network management tasks was introduced, the feature was named self-organizing networks (SON). The very basic use cases for SON include automatic configuration of newly added eNodeB, optimizing some of the configuration parameters with respect to neighbouring cells, and automatic reaction to problems in neighbouring cells. Since then it has been proposed that SON should take more and more of the network operation workload. Just as in other industrial automation cases the plan for the automated solution has gone from ”nice additional help” to ”automate everything”, which seems only natural considering the progress made in machine learning in past ten years, yet replacing human operators is still far away.[29, Section 10.5].

In addition to the cell level aggregations of the KPIs, it would be helpful to be able to determine if it is some well-defined area within the cell that is having problems, but the cell-based KPIs can only help to identify the problem, not localize it. As mentioned in the introduction, some of the problems can be very localized, like concerts, in which case more granular information than cell level knowledge might be needed for easy troubleshooting.

How about planning then, how to quantify if some parameters are good or not before the network is even operational? Section 2.1.1 tells pretty clearly how the planning is done and it is also mentioned that the signal propagation KPIs can be computed with simulators. While simulators can be used to estimate KPIs on an area without any observations from UEs there are two problems with simulated data, firstly simulators may not be accurate and the results need to be verified (or the simulators need to be calibrated) with drive testing, and secondly, when using highly accurate simulation techniques like ray-tracing the computational cost can be massive for complex or large environments. Typical raster sizes for simulation is $25m \times 25m$ to $500m \times 500m$ [29, section 9.3.3]. Besides simulation, when upgrading network from older generation it is also possible to use measurements done in the older network to get a baseline for the new network.

2.2 Radio Environment Map

Radio environment maps were originally created for the needs of cognitive radio and much of the research is centre around the needs of cognitive radio, so few words about cognitive radio at first. The concept of cognitive radio was first introduced

by Mitola and Maguire in 1999 [26]. The basic idea is that the radio (cell phone) is aware of everything and can make predictions based on its user's actions but also based on the information given by other radios to use the radio resources more efficiently. From a radio that is aware of everything around it, as envisioned by Mitola, it was developed towards a radio that is aware of the available and usable spectrum around it. The first actual use case for cognitive radio was using spectrum allocated for television broadcast to do other transmissions without interfering the primary users of the spectrum. To enable even higher efficiency in spectrum usage many methods have been developed such as sensing-based spectrum sharing [19] and database-based spectrum sharing [46]. Where the first method is based on measurements made by the radio itself and the second one is a map of the primary user spectrum usage based on simulations and topology data. Combining these two ideas we have a map that is constructed using measurements from all around the mapped area instead of simulation, this map is called radio environment map (REM) or radio frequency layer of radio environment map if one wants to make a distinction between the radio frequency map and other features often found from REMs [30]. In this thesis, REM refers only to the radio frequency map part.

Constructing REMs can be roughly divided into direct and indirect methods. Indirect methods involve knowing some parameters beforehand and then simulating to obtain the map similarly as was done by Villardi et al. for database-based spectrum sharing[46]. Direct methods use available measurements to create interpolation that tries to predict the value of signal measurement at an unobserved location, as this is very close to what is needed in the thesis the rest of the chapter will focus on these direct methods.

Surveys [30, 21] mention multiple interpolation methods that have been used for creating REMs from measurement data. Especially interesting methods are nearest neighbour interpolation as it has lowest computational complexity out of the methods compared, also the Kriging method is of interest as it is claimed to be the most accurate, both of these methods are mentioned in multiple surveys. Inverse distance weighted interpolators were studied in detail by Denkovski et al. [10], IDW interpolation is also interesting as it can be modelled as a k-nearest neighbours problem with some restrictions, but it should provide better accuracy than using k-NN interpolation without weights. The flavour of k-NN used in the thesis will be discussed more in 2.3.1 and the evolution of Kriging applied to larger and larger data sets is outlined in 2.3.2.

A novel and less studied approach to the map generation from sparse observations is given by tensor completion. A particularly interesting and detailed paper about the subject is given in by Teganya and Romero [43], who modelled the problem as a deep learning problem that is then solved by an encoder-decoder network. The

results given by the encoder-decoder solution look promising at least in the simplistic case modelled in the paper. Very recent work in the field is done by Liu et al. [22], who give a very nice introduction to the problem but do not provide details on the encoder-decoder construction as the paper is more focused on efficient data collection through crowdsourcing.

Some problems of highly varying or outright erroneous measurements in the context of REMs are studied by Farnham [13]. The most notable recommendation is to smooth the data, for example by making sure that each cell in the grid used to make the map is calculated by using more than one observation (three in the paper).

Given how close REMs are to network coverage analysis for LTE (and other) networks, it is no surprise that REMs have already been used for coverage analysis in LTE with the help of MDT measurements. Galindo-Serrano et al. use REMs for coverage hole detection [16]. However, while the method is proposed to work with MDT data the tests are done on simulated data. Furthermore, the paper only considers coverage holes detectable with RSRP measurements.

What is inherently missing from sources working with REMs cited earlier is forecasting the radio conditions. In many REM articles, the REM only changes when significant change is detected. Being able to forecast how the signal conditions will change in upcoming hours for example would be very beneficial for mobile network operators, also it would be closer to Mitola's original idea of cognitive radios. Some work for predicting the measurements in time after aggregating over some spatial area has been done by Mureithi et al. [28], however, the predictions have not been applied to a map. In the other hand Rahman et al. consider the temporal aspect by constructing a map for each state of the transmitters [33], but adapting this to LTE networks would be harder as the state change can be more granular than in the paper and thus the number of states would be larger.

Recently, REMs have also gathered interest in military communications and electronic warfare use cases, most notably as part of NATO research project for electromagnetic environment situational awareness cited for example in a study by Suchanski et al. [42]. Now, because REMs are used as part of cognitive radio and military communications research, any progress or results I gain trying to improve the workflow of mobile network operators will also advance these other fields of research as well.

All the studies sampled in this chapter have taken a similar approach to visualize the REM, sample the function on some regularly spaced grid and create a raster image from those results. The "pixels" in the image are squares of some area with colour based on the value given by the REM construction method. The size of the "pixels" or grid cells are from two meters [41] to 25 meters [16] while the aggregation

of real data (but not used with any prediction methods) is done at a maximum resolution of 10 by 10 meters [37]. The size of the whole area that is mapped varies between 100m x 100m [43] and 2,7km x 2.7km [41]. Sadly, not all studies report these values so getting a good view of what is a typical approach is hard.

2.3 Prediction Methods

The prediction problem that this thesis tries to solve can be formulated as: Given observed values y at locations x, what are the expected values y_* at unobserved locations x_*. Adapting this to the problem at hand, the y would be the observed RSRP or another measurement at location x which is given by GPS coordinates and a timestamp and the problem is to find probable value for the measurement at an arbitrary location. This problem definition also fits interpolation, which is the word commonly used for this procedure in the REM literature. Also, as stated in section 2.2, it is common that the predictive function/interpolation is evaluated at locations forming a regular grid so especially time complexity for generating such a grid is of interest. Time complexity often depends also on the dimensionality D of x, in this thesis D is either 2 (spatial interpolation) or 3 (spatio-temporal interpolation) so the dimensionality does not have a very large effect. In this thesis the spatial area is fully contained in the training set of data thus matching the definition of interpolation, however, the time dimension is being forecast so interpolation may not be a fitting term for the three-dimensional case.

Next, I will introduce the three prediction methods used in this study. First, the nearest neighbour interpolation as it is arguably the simplest of the three, the chapter will also include inverse distance weighting interpolation as an extension for k-nearest neighbours interpolation. The second one will be the Gaussian process regression, which has seen quite many modifications trough different approximation techniques to make it feasible on larger datasets. In the Gaussian process section, I want to introduce some of these approximation techniques to illustrate how the version used in the thesis is different from the others and what we may be losing when using this approximation over others. The basic explanation follows the book by Rasmussen and Williams [34]. Finally, auto-encoder neural networks are investigated based on the blueprint provided by Teganya et al. [43].

The prediction techniques cannot be always compared directly as some can be evaluated in arbitrary locations while others require regular grid for the inference. I will provide detailed explanation how the methods have been compared to each other in Chapter 3, but I will also highlight the differences in this chapter as they can be important when choosing a suitable method.

2.3.1 Nearest Neighbours Regression

The simplest possible prediction method would be associating each point with the nearest measurement in the data, this approach is called nearest neighbour interpolation. Calculating the nearest neighbour to all points in the mapped area is same as calculating a Voronoi diagram for the area, which can be done in $\mathcal{O}(n\ \log(n))$ time[15]. For a single location, the nearest neighbour can be found in approximately $\mathcal{O}(log(n))$ time using kd-trees [5] or cover-trees [7], however using these advanced structures for speeding up the search makes exact time complexity calculations hard and add quite large overhead for creating the tree in the first place. For both of the tree structures, the exact time complexity also depends on the dimensionality of x and distribution of the points in space. Finding the interpolation value at multiple locations, for example when creating a grid of values, scales linearly with the number of locations m and thus the complexity is approximately $\mathcal{O}(m \log(n))$. The downside of simple nearest neighbour interpolation is the high variability of the output as each (possibly noisy) measurement is taken as true value over some area.

To reduce the variability, multiple nearby points can be used together to produce the output value. In regression setting, the simplest version of this is choosing k points in the data that are nearest to the one to be estimated and taking the average of those points. If calculating exact time complexity for 1-nearest neighbour search is hard, calculating it for k-nearest neighbours search is not any easier, but we can assume that the time complexity is somewhere between $\mathcal{O}(k\ \log(n))$ and $\mathcal{O}(k^2\ \log(n))$for single location and $\mathcal{O}(mk\ \log(n))$ and $\mathcal{O}(mk^2\ \log(n))$ for m locations.

In inverse distance weighted interpolators, each point in the training data is weighted based on how far it is from the location we are trying to interpolate, closer points getting higher weight than far away ones and then the interpolation value is calculated from the weighted points. We can borrow from this idea and apply it to a local neighbourhood of k-nearest points instead of all the points in the data because the points that are further than the k-nearest points have lower weight making them gradually less important as k increases [39]. Furthermore, the paper from Sheppard [39] introduces more ways to improve the interpolation result by selecting a maximum distance in conjunction with the k parameter and by incorporating direction to the weighting function. These approaches have been evaluated for usage in REMs by Denkovski [10].

Based on the above chapter we get four different parameters that need to be optimized for optimal results: number of neighbours to consider, a maximum distance of a point, a minimum number of points chosen regardless of the maximum distance and inverse distance weight parameter. The extension to time dimension

can be done just by considering time as a third feature used to calculate the distance, however as time and spatial distance do not use same units there needs to be some scaling done to get sensible results. The correct scaling value from seconds to meters as well as the other parameter values have to be learned from the training data.

2.3.2 Gaussian Process Regression

Gaussian process (GP) regression is commonly known as Kriging in geostatistical context, named after Daniel G. Kriege, even though there are subtle differences between the terms they are used interchangeably in the thesis. The explanation of the Gaussian process regression follows book from Rasmussen and Williams [34] and prominent approximation techniques are discussed based on other sources as well.

Bayesian Linear Regression Model

The book starts the derivation of the Gaussian process regression from the standard linear regression model with Gaussian noise

$$f(x) = x^T w, \quad y = f(x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2). \tag{2.1}$$

A Gaussian prior with zero mean and covariance matrix $\boldsymbol{\Sigma}_p$

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_p) \tag{2.2}$$

is applied to the weights of the linear model in (2.1). Given the model, noise distribution and the prior distribution, a posterior distribution for the weights can be derived. Because the noise and prior were Gaussian also the posterior will be Gaussian. The posterior is given by

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) \sim \mathcal{N}(\sigma_n^{-2}(\sigma_n^{-2}\boldsymbol{X}\boldsymbol{X}^T + \boldsymbol{\Sigma}_p^{-1})^{-1}\boldsymbol{X}\boldsymbol{y},\ (\sigma_n^{-2}\boldsymbol{X}\boldsymbol{X}^T + \boldsymbol{\Sigma}_p^{-1})^{-1}),$$

where $\boldsymbol{X}$ is now $D \times n$ design matrix consisting of the inputs x_i. The equation can be shorthanded to

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) \sim \mathcal{N}(\sigma_n^{-2}\boldsymbol{A}^{-1}\boldsymbol{X}\boldsymbol{y},\ \boldsymbol{A}^{-1}), \quad \boldsymbol{A} = \sigma_n^{-2}\boldsymbol{X}\boldsymbol{X}^T + \boldsymbol{\Sigma}_p^{-1}. \tag{2.3}$$

From (2.3) it can be seen that obtaining the weight parameters $\boldsymbol{w}$ requires an inversion of matrix $\boldsymbol{A}$ that is a $D \times D$ matrix.[34, p. 8-9].

Obtaining the predictive distribution f_* (and thus also the distribution of y_*) at locations x_* can be done by averaging the output of all possible linear models with

respect to the posterior (2.3).

$$\begin{aligned} p(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) &= \int p(f_*|\boldsymbol{x}_*, \boldsymbol{w}) p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})\, d\boldsymbol{w} \\ &= \mathcal{N}(\sigma_n^{-2} \boldsymbol{x}_*^T \boldsymbol{A}^{-1} \boldsymbol{X}\boldsymbol{y},\ \boldsymbol{x}_*^T \boldsymbol{A}^{-1} \boldsymbol{x}_*) \end{aligned} \tag{2.4}$$

Thus far we have considered the standard linear model in Bayesian framework, however, this kind of model is too rigid for our goal of spatial prediction.[34, p. 11].

Feature Expansion

To make the model more flexible we can project the inputs x to feature space of dimensionality N using a mapping function $\phi(x)$. Choosing the basis functions to use for the mapping is an important factor in accuracy and computational performance of the Gaussian process. One example of such mapping function could be $\phi(x) = (1, x, x^2, x^3, ...)$. Projection like this would make the model more flexible but keep it linear with respect to the weights $\boldsymbol{w}$, and thus analytically tractable.[34, p. 11]

Extending the model to include this kind of projection of inputs is simple, as all the equations still hold when individual input location x and model matrix $\boldsymbol{X}$ are replaced by $\phi(x)$ and $\boldsymbol{\Phi(X)}$ respectively. Notably the lower part of equation (2.4) becomes

$$\begin{aligned} p(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) &= \mathcal{N}(\sigma_n^{-2} \phi(\boldsymbol{x}_*)^T \boldsymbol{A}^{-1} \boldsymbol{\Phi(X)} \boldsymbol{y},\ \phi(\boldsymbol{x}_*)^T \boldsymbol{A}^{-1} \phi(\boldsymbol{x}_*)), \\ \boldsymbol{A} &= \sigma_n^{-2} \boldsymbol{\Phi(X)} \boldsymbol{\Phi(X)}^T + \boldsymbol{\Sigma}_p^{-1}. \end{aligned} \tag{2.5}$$

Computing f_* from (2.5) now requires inversion of $N \times N$ matrix $\boldsymbol{A}$, which may not be computationally feasible if N is large e.g. infinite. Luckily for situations like this, the equation can be rewritten as

$$\begin{aligned} p(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}(&\phi(\boldsymbol{x}_*)^T \boldsymbol{\Sigma}_p \boldsymbol{\Phi(X)} (\boldsymbol{K} + \sigma_n^2 I)^{-1} \boldsymbol{y}, \\ &\phi(\boldsymbol{x}_*)^T \boldsymbol{\Sigma}_p \phi(\boldsymbol{x}_*) - \phi(\boldsymbol{x}_*)^T \boldsymbol{\Sigma}_p \boldsymbol{\Phi(X)} (\boldsymbol{K} + \sigma_n^2 I)^{-1} \boldsymbol{\Phi(X)}^T \boldsymbol{\Sigma}_p \phi(\boldsymbol{x}_*)), \end{aligned} \tag{2.6}$$

where $\boldsymbol{K} = \boldsymbol{\Phi(X)}^T \boldsymbol{\Sigma}_p \boldsymbol{\Phi(X)}$. Now the computation requires inverting $n \times n$ matrix, where n is the number of observations. As the number of observations is always less than infinity this can make the computation possible in the first place. [34, p. 12].

The time complexity of calculating a matrix inverse is $\mathcal{O}(n^3)$ so being able to technically do the computation seems like cold comfort in the context of this thesis where the number of observations available in the data is expected to be very high. Furthermore, storing the $n \times n$ matrix $\boldsymbol{K}$ requires $\mathcal{O}(n^2)$ memory, which might also be problematic. What makes Gaussian process regressions special compared to

other interpolation methods introduced and still worth studying further is the form of the output. As seen in the equations (2.5) and (2.6), the output is a Gaussian distribution with some mean and covariance. This allows very powerful interference on the results because besides the mean we also get the variance and well-defined confidence regions. As with k-nearest neighbours interpolation, we want to sample the mean and now also the variance at m locations, which can be obtained with the complexity of $\mathcal{O}(mn)$ for mean and $\mathcal{O}(mn^2)$ for the variance after calculating the matrix inverse. So let us continue with some more theory before going into the approximation techniques used to decrease the time and memory complexity.

Kernel

The inputs projected to the feature space in equation (2.5) are always in the form $\phi(\boldsymbol{x})^T \boldsymbol{\Sigma}_p \phi(\boldsymbol{x'})$, where $\boldsymbol{x}$ and $\boldsymbol{x'}$ are either in the training or test set. From this, we can see that the prior from eq. (2.2) applied to weights has a profound effect on how the Gaussian process is formed. Let us define

$$k(x, x') = \phi(x)^T \boldsymbol{\Sigma}_p \phi(x'), \tag{2.7}$$

and call $k(\cdot, \cdot)$ a kernel or covariance function. Furthermore, by decomposing the covariance matrix of the prior $\boldsymbol{\Sigma}_p$ from equation (2.2), the kernel can be expressed as a dot product

$$k(x, x') = \boldsymbol{\Sigma}_p^{1/2} \phi(x) \cdot \boldsymbol{\Sigma}_p^{1/2} \phi(x'). \tag{2.8}$$

By defining $\psi(x) = \boldsymbol{\Sigma}_p^{1/2} \phi(x)$ the dot product can be rewritten to be $k(x, x') = \psi(x) \cdot \psi(x')$, making it easy to see that the kernel can be defined as dot product of the input space. This allows replacing the projections from input space to feature space with the kernel, avoiding the computation of the feature vectors and instead computing the kernel in the cases where computing the feature vectors would be costly. It also allows us to discuss the properties of the kernel instead of the feature space it corresponds to. This has also caused much of the Gaussian process research to centre around finding suitable kernels for good accuracy or faster approximation. Each kernel also corresponds to some covariance matrix of the prior, so discussing the form of the prior is in some sense same as discussing the form of the kernel[34, p. 12].

The matrix $\boldsymbol{K}$ that collects all the kernels is called Gram matrix; a Gram matrix is defined to be positive semidefinite. The definition limits the space of functions that can be considered as valid kernels, but it also causes that the valid kernels can be seen as a similarity measure between two locations. An example of a kernel that is commonly used but rather simple is the squared exponential (also known as

isotropic Gaussian):

$$k(\boldsymbol{x}, \boldsymbol{x}') = exp(-\sum_{d=1}^{D}(x_d - x'_d)^2/\theta), \tag{2.9}$$

where θ is a hyperparameter that controls the length-scale of the process. From the equation (2.9) we can see that the kernel is isotropic; it only depends on the squared distance between points, not the direction of the distance. The kernel is also stationary because the value does not depend on the location of x or x' i.e. in one dimension $k(x + h, x' + h) = k(x, x')$ for any scalar h.

Hyperparameters

Gaussian processes can have many hyperparameters that need to be optimized as part of the training. The equation (2.9) above has one hyperparameter θ that controls how flexible the prediction is or in different terms; how much the distance affects the correlation between two locations. Changing length-scale is equal to rescaling dimensions of x. It is common to see the squared exponential kernel written with two more parameters as

$$k(x_i, x'_j) = \sigma_f^2 exp(-\sum_{d=1}^{D}(x_d - x'_d)^2/\theta) + \delta_{ij}\sigma_n^2, \tag{2.10}$$

where σ_f is the signal variance, σ_n is the noise variance and δ_{ij} is the Kronecker delta function. Signal variance σ_f affects how much the function can vary from the mean, changing signal variance is equal to rescaling the response y of the training data. Noise variance (also known as nugget effect in kriging context) affects how closely the function follows the observed locations and affects the prediction variance as allowing more room for movement at the training locations allows more wiggly functions. The noise variance σ_n is the portion of the variance that is independent of the other observations. While this kernel only has 3 hyperparameters there could be many more in some other kernel, for example in the next section we deal with relevance vector machine approximation which has a hyperparameter for each basis function.

Optimal values for these parameters can be obtained by calculating the marginal likelihood over the parameters. In general, marginal likelihoods require calculating (or rather approximating) complex integrals over parameter space, which can be done for example with Markov chain Monte Carlo (MCMC) methods, at a very high computational cost. Fortunately, there exists a subset of problems where the integrals have analytical solutions, as is the case with Gaussian processes with Gaussian noise [34, Chapter 5].

Approximations

Observing the equations (2.5) and (2.6) it can be seen that it is possible to invert the $N \times N$ matrix or $n \times n$ matrix and still end up with same predictive distribution. This leads to two routes for creating faster approximation: reducing the number basis functions N or reducing the number of observations n to consider. Both of these routes shift the problem from computing the matrix inverse to choosing either good basis functions or good observations to include. Rasmussen et al. [34, chapter 8] outlines some of these approaches, especially choosing k observations from the n total observations and approximating the Gram matrix based on those. A more comprehensive and recent overview of different approximation and computational speedup methods is provided by Liu et al[23].

Based on the results provided in [34, p. 182], approximation method that provides the best balance between accuracy and runtime is one called "subset of regressors". The subset of regressors method is based on approximating the Gram matrix with a subset of size m taken from the n training observations. The optimal approximation of the Gram matrix $\boldsymbol{K}$ with respect to Frobenius norm is $\boldsymbol{K} = \boldsymbol{U}_m \boldsymbol{\Lambda}_m \boldsymbol{U}_m^T$, where matrix $\boldsymbol{\Lambda}_m$ contains the top m eigenvalues of $\boldsymbol{K}$ and $\boldsymbol{U}_m$ contains the matching eigenvectors[34]. However, computing a full eigendecomposition is an $\mathcal{O}(n^3)$ operation, so a faster way of obtaining the eigenvectors and eigenvalues is required. One way to obtain approximate eigendecomposition is given by the Nyström method which leads to the Gram matrix being approximated by $\tilde{\boldsymbol{K}} = \boldsymbol{K}_{nm} \boldsymbol{K}_{mm}^{-1} \boldsymbol{K}_{mn}$, where $\boldsymbol{K}_{mm}$ is a Gram matrix collected from the kernels of the m datapoints that are included in the calculation, $\boldsymbol{K}_{nm}$ is matrix collecting kernels between m datapoints and all n points, $\boldsymbol{K}_{mn}$ being its transpose. The matching covariance function is $\tilde{k}(x, x') = \boldsymbol{k}(\boldsymbol{x})^T \boldsymbol{K}_{mm}^{-1} \boldsymbol{k}(\boldsymbol{x'})$, which is induced by the prior (see eq. (2.2)) $\mathcal{N}(0, \boldsymbol{K}_{mm}^{-1})$. Creating the approximation requires $\mathcal{O}(nm^2)$ time, mean and variance prediction at new location requires $\mathcal{O}(m)$ and $\mathcal{O}(m^2)$ time respectively, making the method suitable for even quite large datasets.

As mentioned earlier this kind of approximation depends on choosing a good subset of datapoints to include for the estimation of $\boldsymbol{K}$. Finding the optimal subset is hardly possible as it would require evaluating all possible m subsets of n which quickly becomes impossible. Rather a forward selection strategy is suggested by Rasmussen and Williams [34]; some selection criterion is evaluated for each datapoint and the best datapoint is selected until k datapoints have been selected. Instead of calculating the criterion for all of the datapoints, a randomly selected subset of the datapoints can be considered at each step, this allows computationally heavier selection criteria to be used. One example of a simple selection criterion to use is to calculate the residual sum of squares for each datapoint and choose the one that minimizes the residual error.

In the other vein of approximations, the number of basis functions N is somehow limited so that $N \ll n$, meaning that the covariance function is finite-dimensional. Rasmussen et al. also describe a procedure suitable for this kind of reduction by relating a Gaussian process to a relevance vector machine (RVM). In RVM case the kernel $k(\cdot, \cdot)$ depends on the training data, which means that the prior of the process shown in equation (2.2) depends on the data. The covariance function for RVM is $k(x, x') = \sum_{j=1}^{N} \frac{1}{\alpha_j} \phi_j(x)\phi_j(x')$, where the α_j are parameters estimated from the data. If we want to relate this to the form used by the subset of regressors method, it makes sense to write it in another form instead: $k(x, x') = \phi(x)^T \boldsymbol{A} \phi(x')$, where $\boldsymbol{A}$ is a diagonal matrix containing the α_j^{-1}'s. Optimizing this kind of Gaussian process often leads to some of the α_j parameters tending towards infinity, effectively removing some of the basis functions from the equation. Removing the basis functions leads to sparsity very much the same way as reducing the number of datapoints used for the approximation.[34, Chapter 6].

A very similar approach to reducing the computational complexity is given by Cressie and Johannesson [9], who describe the fixed rank kriging (FRK) procedure. FRK has since then attracted a lot of interest also in the radio environment map literature, especially in the context of mobile networks and MDT measurements [8]. In the FRK approach there is a fixed number of basis functions $r \ll N \leq n$. And very similarly to subset of regressors and RVM approaches the covariance function can be written as $k(x, x') = \phi_r(x)^T \boldsymbol{K}_{rr} \phi_r(x')$. Now the $\phi_r(\cdot)$ gives the mapping from input space to r dimensional feature space and $\boldsymbol{K}_{rr}$ is a positive definite $r \times r$ matrix that is estimated. While the idea is similar to those shown earlier, it allows more flexible specification of the basis functions as no orthogonality is assumed. Cressie and Johanneson offer some choices for the class of basis functions to use. Furthermore, FRK describes "binning" the data to m spatial bins (called basic aerial unit, BAU) to get the method of moments estimator for the covariance matrix of the process. Aggregating the data is optional, but aggregating to $m < n$ bins further reduces the time complexity. Using m to denote the number of bins is done deliberately to highlight the similarity between bins and prediction on a grid, bins and the prediction locations do not need to be the same, however, the prediction area cannot be smaller than the BAU. The computational cost for FRK is the same as for the subset of regressors (substitute k with r), but the estimation method is somewhat different. For FRK the question is how many basis functions is required to allow enough variability in the model.

Locally Approximate Gaussian Process

All of the approximations above generate a global estimate for the Gaussian process functions by modifying the prior, which ensures that the function is smooth (at

least to some degree) and stationary. However, for some processes the stationarity is not a good assumption or smoothness is not desirable feature at all locations and further computational and accuracy gains could be harvested by considering Gaussian process construction that is local to the area to be predicted. Next, I am going to introduce the locally approximated version of Gaussian process that is developed by B. Gramacy and implemented in the *laGP* R package [17]. There are multiple other approaches to local Gaussian process estimation described by Liu et al. [23], but the chosen flavour has answers to many problems found in other similar approaches as well as having (mostly) stable implementation.

The equations to obtain local Gaussian process prediction are the same as in the global version, but now the predictive location x_* is considered already in the construction step. As can be seen from the squared exponential kernel (2.9), as the distance between x and x' increases the kernel value decreases making the far away points irrelevant. Thus in the local Gaussian process construction datapoints far away from x_* can be ignored, meaning that the prediction for x_* can be approximated very well with $k \ll n$. Furthermore, when considering multiple predictive locations the local Gaussian process leads to simple scalability as each location can be predicted without considering the others, unlike in global Gaussian process where the bottleneck is already in the construction step. Common problems for the local constructs are choosing the nearby points to include and finding hyperparameters for the local construct.

In practice, the *laGP* package uses a Student's t process instead of a Gaussian one to allow for more flexibility. In literature the t process has been received quite variably, Rasmussen and Williams being reserved about the analytical tractability [34, Chapter 9] of the t process while Shah et al. believe the t process to be superior to Gaussian process in almost every way and suggest replacing Gaussian process with t process almost everywhere [38]. However, the model construction or inference is not much affected whether we are using Gaussian or t process and in this thesis we will continue speaking about Gaussian process despite the underlying process actually being a t process.

In local context choosing a subset of all datapoints to estimate the function value seems more intuitive than in global context, after all now it is easy to see that datapoints that are far away from the predictive location should not be considered. The naive takeaway from this would be to implement Gaussian process for k-nearest neighbours of the predictive location (very much like the IDW implementation discussed in 2.3.1); however, it has been shown that naively choosing nearest neighbours produces suboptimal results as it is beneficial to have some spread in chosen datapoints to estimate the hyperparameters [17]. The idea of selection criteria is similar in the local context as it was in the global subset of regressors case, but

now we have the location x_* available and it is possible to estimate the error at that exact location, possibly giving a lot better estimate than in the global case where trade-off resulting in overall best fit had to be considered. Specifically, Gramacy suggest selecting the observation that reduces the variance at the predictive location the most. Gramacy introduces a criterion that does that named ALC (active learning Cohn), which is a simplification of mean square prediction error criterion, and a faster approximation to it named ALC-ray. In the global context, the cost of computing the selection criterion was reduced by considering only a randomly selected subset of datapoints at each step, in local context we can do better by choosing nearest neighbours instead of a totally random subset. Let's use j for the size of this subset for time complexity calculations so that $k \ll j \ll n$. The ALC-ray approximation is based on the empirical findings that, at least in some datasets, the ALC criterion tends to select locations following "rays" emitting outward from the predictive location. ALC-ray provides the same behaviour without calculating the selection criterion at all the locations and thus it can use larger j value. The differences between nearest neighbour, mean square prediction error, ALC and ALC-ray selection criteria can be seen in figure 2.2.

After the datapoints to include has been selected we still have to estimate the parameters of the model; length-scale and noise variance. These parameters will be estimated separately for each predictive location, resulting in non-stationary *global* model despite the *local* models being stationary. The *laGP* package allows estimating the parameters using maximum likelihood estimation, based on Newton-like iterative scheme, either separately or jointly, based on some limits or with maximum a posteriori probability estimation if a gamma prior is given.

The computational complexity of this kind of locally approximate Gaussian process has many more variables than the global Gaussian processes shown before, but let us try to get something that is comparable with the other methods. To begin with, calculating the posterior requires inverting $k \times k$ matrix at $\mathcal{O}(k^3)$ time, note that k used in the local context is generally a lot smaller than k used in the global context. Selecting the k datapoints from j closest ones to include requires evaluating the selection criterion almost kj times, as the selected locations are removed from the pool of available ones but $k \ll j$ is assumed. For example, the ALC criterion's time complexity depends on the number of already selected datapoints k_s as $\mathcal{O}(k_s^2)$ for each candidate left, which means that the time complexity of this step would be bounded from above by $\mathcal{O}(k^3 j)$. Maximum likelihood estimations exact time complexity depends on how many iterations are needed for convergence, however, this cost is not expected to be higher than finding the datapoints for the local design. Thus the overall time complexity for a single location is $\mathcal{O}(k^3)$, which hides a rather large constant. And $\mathcal{O}(mk^3)$ for m locations, for modern computers it is noteworthy

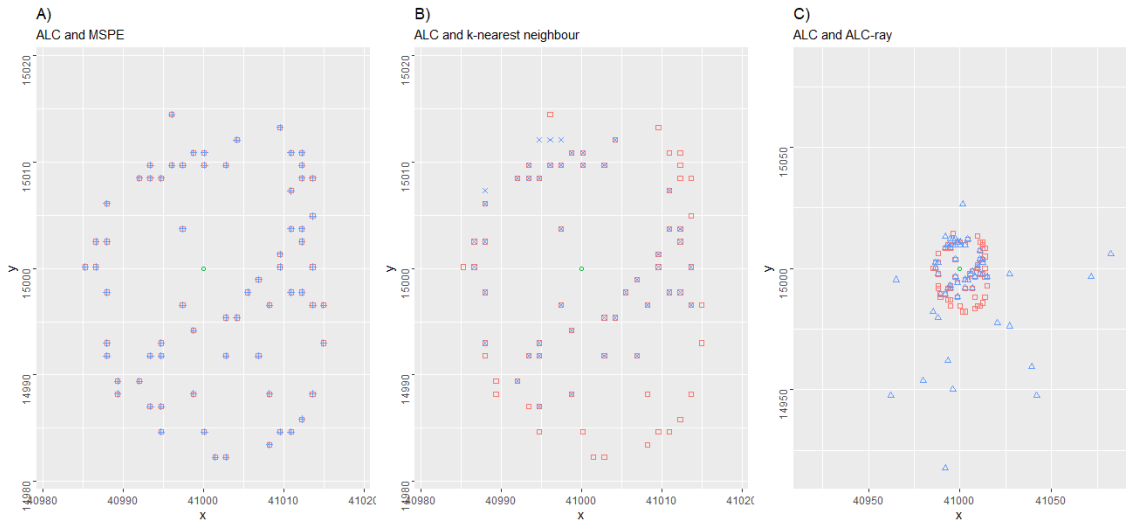

Figure 2.2 *Shows comparison between ALC selection criterion with mean squared prediction error (MSPE), nearest neighbours and ALC-ray when selecting 100 locations to include for one predictive location (green circle), the methods are applied to dataset introduced in section 3.1. Plot A shows a comparison between ALC (red square) and MSPE (blue cross) criteria, it can be seen that the ALC and MSPE select exactly the same locations despite ALC being considerably faster of the two. Plot B compares ALC (red square) and nearest neighbour (blue x) methods. Clearly, ALC and nearest neighbours do not choose same locations, furthermore, it looks like nearest neighbours is choosing multiple measurements that are at the exact same location, which the ALC is not doing, allowing ALC to cover a much larger area. C is comparing ALC (red squares) and ALC-ray (blue triangles) methods, it can be seen that in this dataset the ALC method is not emitting rays as expected and thus ALC-ray fails to find similar structure as ALC.*

that the computation can be divided to m threads without much overhead. Furthermore, the ALC and ALC-ray computations can also be done massively parallel for example with graphical processing unit (GPU) by computing the criterion for each possible datapoint $j - k_s$ in parallel. The *laGP* leaves the selection of exact values for j and k as well the MLE limits and priors to the user with some reasonable defaults, the optimal values are dataset specific and could be estimated from the training data as will be shown in 3.3.1.

2.3.3 Artificial Neural Networks

Artificial neural networks (ANN) refer to computing systems that are inspired by biological neural networks found in animal brains, like its biological predecessor, ANNs can learn based on examples without being explicitly programmed to do the task. The earliest work on ANNs can be found from the 1940s when McCulloch and Pitts laid basic mathematical foundations required for ANNs [25], since then a lot of the connections to biology have been abandoned and the research has been driven forward by computer science. From a computer science perspective the ANN is seen

as a graph where each node is a neuron associated with some activation function and each edge is a connection between neurons having some weight based on how strong the connection is. Given a large enough network, the ANN can approximate any continuous function, learnability of the parameters or feasibility of such "large enough network" is another matter.

Nowadays by far the most common way to portray ANNs is a structure where the nodes are arranged in layers that get sequentially applied to the input data to produce an output. The backpropagation algorithm is a common way to train this kind of layered ANNs. In backpropagation, the error between the generated output and ground truth sample is propagated backwards through the network and the weights of the network are changed based on the gradient of the error function so as to find the minimal error. The language used in ANN literature is not quite standard on how to define many of the concepts used in ANNs, for example sometimes the activation function is seen as part of some layer and sometimes as its own separate layer, the wording used in this thesis tries to agree with "sequential model" of the Keras deep learning library.

This section will give a simplified introduction to ANNs so that the reader can understand the network shown in [43]. The introduction will begin with the general structure of auto-encoder networks and the type of layers used in this specific network. Some more emphasis is given to convolutional layers due to their common usage in a variety of applications.

Autoencoder Network

Autoencoder refers to any two functions F, the encoder function, and G, the decoder function, that satisfy $F : \mathbb{R}^n \to \mathbb{R}^d$, $G : \mathbb{R}^d \to \mathbb{R}^n$ and $F \circ G : \mathbb{R}^n \to \mathbb{R}^n$, where $d < n$. Same in English would mean any two functions where the first one maps n-dimensional input vector to d-dimensional output vector where d is less than n, and a second function that maps the lower-dimensional vector back to a higher dimension. The idea is to minimize the difference between the n-dimensional input vector and n-dimensional output vector given by the second function, achieving small error between the input and output would indicate that the n-dimensional vector can be well presented by a d-dimensional vector that was in between, in the other hand if the d-dimensions are not enough to present the data the output can become too smooth and not present any features of interest. Since ANN is capable of learning any continuous function it is sensible to model F and G as ANNs, thus autoencoders have become almost synonymous with autoencoder networks even though other functions than ANNs could be used as well.

Using the layered presentation for ANN, an autoencoder network in its simplest would have three layers: an input layer, a hidden layer and an output layer, the input

and output layers would have n nodes and the hidden layer would have d nodes. Of course, any number of layers could be used as long as one of the layers has fewer nodes than the input or output so that the autoencoder is clearly formed from an encoder and a decoder functions. In the paper by Teganya et al. [43] the aim is to find encoder and decoder functions that are able to find the correct output image when some parts of the input image are masked or missing, image reconstruction in a sense. This kind of task is called "denoising" and consequently, the autoencoder is called denoising autoencoder (DAE). Autoencoders are considered suitable for this task because it is expected that the encoder can find a good approximation for the lower-dimensional presentation even with partial input data and after the lower dimensional presentation is generated the missing or incorrect data in the input should not matter for the decoder.

The input to the neural network used by Teganya et al. [43] is a matrix that contains observations and missing values (zeros) at different locations on the uniform grid. Additionally, the input is enhanced by merging it with a second matrix that contains the locations of missing values. Let's assume that the uniform grid used to aggregate the observations is a square and denote the observation matrix as $O \in \mathbb{R}^{n \times n}$ and the missingness indicator matrix as $M \in \{0, 1\}^{n \times n}$. Together they create a tensor with dimensions $n \times n \times 2$. Another way to see these uniform grids is as an image where each pixel is an aggregated observation value from the grid, now that we have "glued" two such images together it would match a normal image with only two colour channels.

The layers of a neural network are defined by the activation function, other operations they might apply to the input and their hyper-parameters. Arguably the simplest layer is a fully connected layer where each node is connected to all the nodes in the previous layer. So a single node has an edge, and thus also a weight value, associated with all the nodes in the previous layer. The outputs of the previous layer get multiplied by the weight values and added together, then bias is added to the sum and finally, an activation function is applied on the sum before passing it to the next layer (note that the activation function can be identity, in which case the value does not change). However, fully connected layers can cause a lot of (unwanted) computation, for example having fully connected input layer in the case discussed previously would lead to $2n^2$ weight values for each node in the input layer. For this reason, multiple ways to reduce the number of connections have been created, most notable one for this thesis and the paper of Teganya et al. is the convolutional layer based on convolutions that will be discussed more in-depth later. The hyper-parameters for fully connected (and many other kinds of) layers include; the number of nodes, parameters for regularization that can be applied to the weights and biases and initialization values. These are not found

with backpropagation and they require some other way to optimise (often sensible defaults).

The reduction in the size required by the encoder function is obtained with *pooling* layers. Pooling layers are characterized by the pooling function, size of the pooling window and the stride between separate windows. The parameters suggested by Teganya et al. is to use pool of size 2×2 with the stride of 2 and calculate the average as the pooling function. What this means is that each value in 2×2 area are averaged to produce one output value, then the area is moved by two spaces (stride = 2) and the averaging is repeated. The averaging is repeated until all non-overlapping 2×2 areas are averaged, as the pooling layer with these parameters averages four input values into one output, the output is four times smaller than the input. The operation is performed separately for O and M so the output dimensions of one such layer is $\frac{n}{2} \times \frac{n}{2} \times 2$. For the decoder there exist a layer with a similar operation performed the other way around meaning that one input is split into four outputs, the layer is called up-sampling layer and the splitting can mean simply repeating the same value multiple times or interpolating the values from other nearby values. The interpolating version is used in the paper.

The activation function is the part of the neural network that allows it to learn non-linear features. A lot of work has been put into finding optimal activation functions, but very few seem to be able to beat some of the simplest functions imaginable outside some specific datasets. Probably the most commonly used activation function is *rectified linear unit* (ReLU), awfully complex name for something that is essentially a max function between input value and a zero, or more formally $max(x, 0)$. Despite the simplicity, it has some features that make it hard to beat, its gradient is very simple to calculate and it often causes about half of the outputs to be zero making it very efficient to calculate. Additionally, the gradient being either zero or one helps with the vanishing gradient problem. The largest problems with the ReLU is that it is unbounded, allowing the outputs from layer to be arbitrarily high and the fact that nodes having ReLU activation can be accidentally be trained so that the gradient is zero on practically all inputs, making the node untrainable. The untrainability problem is fixed a variant called leaky ReLU that assigns a small gradient to negative values instead of zero and the unboundedness can be mitigated by scaling the values between layers. Teganya et al. use parametric leaky ReLU, which allows the leakiness (the gradient of the negative values) to be trainable, it is unlikely that the exact choice between ReLU, leaky ReLU or parametric ReLU makes a large difference. A place where the activation function does matter is the output layer and ReLU is rarely the right choice for output, the output activation function of the paper from Teganya et al. is not specified but I assume it to be linear/identity function.

In the paper by Teganya et al. the data is obtained with simulation and the simulation result is taken to be the ground truth value for the training. To allow the model to learn, some of the values from the ground truth are masked with zeros and this masked data is given as an input to the neural network. The output from the network is a matrix with same dimensions as the input but without masked values, now error between the ground truth and the predictions given by the network can be used for training the network.

Convolutional Layer

The idea of a convolutional layer was first popularized by LeCun in 1998 [20], where convolutional layer was used in a network trained by backpropagation and the network was used to classify handwritten digits. Since then convolutional layers have been proposed to work in a multitude of different pattern recognition tasks, especially on image inputs. Convolutional layers have a few useful features when applied to images. First of all, they are not fully connected and thus have significantly fewer weights than fully connected layers, which is especially important with image inputs containing easily thousands of dimensions. Secondly, convolutional layers are shift-invariant, meaning that it does not matter where in the image the pattern of interest is found.

A convolutional layer works by extracting feature maps from the image, the method used to extract the feature maps corresponds to convolution operation applied to the input, hence the name *convolutional* layer. In practice each feature map has its own weight matrix that matches to some input pattern of interest, size of the matrix determines how large area of the input is considered at once. The feature map is extracted by centring the weight matrix on each input unit (note that units at the boundary of the input require some padding) and multiplying the input values by the weight in the corresponding location in weight matrix and finally summing the weighted inputs. After the weighting and summing has been applied to all input units and the results have been collected to a matrix, we have a matrix with same dimensions as the input where each value shows how well the input location matches the weight matrix. To find multiple patterns, the same procedure is applied multiple times with different weight matrices. Finally, an activation function can be applied before sending the output to the next layer. The final dimension of convolutional layers output is $n \times n \times k$, where k is the number of feature maps.

Each value in the output of the convolutional layer is typically based on a rather small area in its input, to allow a larger *receptive field* convolutional layers are often applied subsequently and combined with the pooling layers. Having a pooling layer between two convolutional layers allows a multiple times larger receptive field in the original input.

Forecasting With Neural Networks

Convolutional neural networks are generally rather strict about their input and output, it has to be a matrix (or a tensor), so the predictions cannot be sampled at an arbitrary location and inputs have to be aggregated onto a uniform grid. But when it comes to forecasting the price that has to be paid in input/output flexibility may be worth it. While Tegenya et al. trained the autoencoder with masked input locations from a single timepoint and tried to predict the masked values while keeping the non-masked values the same, there is no actual reason to limit to predicting the current timepoint. The autoencoder model can be used for forecasting the next timepoint by just changing the training targets to be the next timepoint. Of course, this will require a lot more learning capability from the model, but given large enough model and enough training samples, it should be doable. Also, on the input side, there is no reason why the network could not be given four past timepoints, for example, this would allow the network to base its prediction on more data than just one timepoint. To this end the convolutional layer easily generalizes to three input dimensions, allowing the weight matrix to be applied on inputs that are close both in space and time.

3 Implementation

This chapter answers the research questions by implementing the theories introduced in chapter 2. Specifically, k-nearest neighbours interpolation (k-NN), fixed rank kriging (FRK), locally approximate Gaussian process regression (laGP) and artificial neural networks (ANN) will be considered and problems related to implementing them will be discussed.

The chapter begins by introducing the data that has been acquired to do the analysis by giving some basic metrics that are then compared to other datasets found in the literature. Also, some special considerations and limitations posed by the data are highlighted.

The data processing as well as some other problems require attention before the analysis can be conducted, these include: what tools to use and how to calculate accuracy in this context.

After those problems have been adequately solved we can fit the models. Fitting includes some form of parameter search for all the models. On one end we have k-NN which has a small set of hyper-parameters and on the other end, we have ANNs that have thousands of parameters and tens of hyper-parameters.

When we have predictive models we can do inference using them in different kind of predictive situations. It is also possible to experiment with different kind of visualizations and ways of detecting network performance degradation.

3.1 Dataset

As one can imagine constructing a map of the current situation of radio environment requires a large number of measurements about the quality of the received signal at known locations, and for temporal analysis also at known timepoints. For the purpose of this thesis, I have acquired a dataset consisting of more than 60 million records collected from user devices in the LTE network of a large city over the span of about 3 weeks. The data is recorded through the MDT feature. The measurements include downlink measurements such as RSRP and RSRQ (see 2.1.2 for details), uplink measurements like UEs power headroom and other information including GPS location and time. All the features used in the thesis are shown in table 3.1. Sadly many of the observations have missing or unreliable data in features that are required for the study, such as GPS coordinates. The observed area also varies with time and the three week time period contains some unobserved periods as well.

The recorded area is 2071 km^2, this leaves us with average observation density of 0.0302 obs/m^2. The closest data set that is somewhat comparable in the observed

area can be found in a paper from Alimpertis et al. [2], where the biggest data set they are working with covers 1600 km^2 of LA metropolitan area with 6.7 million observations, meaning they have observation density of 0.0042 obs/m^2. Comparing to other studies that are constructing REMs, 2071 km^2 is way beyond the scale of anything reported in papers sampled for section 2.2. On the other hand, in the paper from Scaloni et al. [37] they are observing a small area of 9 km^2 and have 6.9 million observations, which would mean 0.767 obs/m^2. In the data used in the thesis, most of the observations are clustered in the city centre, leaving plenty of areas without any observations while the busiest areas can be over the 0.770 obs/m^2 that is found in the paper of Scaloni et al.

Because the data is not collected as part of a specialized drive test campaign but instead from user devices it is obviously less accurate. While the measurement devices used in drive testing have a uniform design and high-quality components meant for precise measurements or at least same device is used for all the measurements, the user devices come in many shapes and forms. There is a wide variance in quality and radio characteristics between different mobile devices, and that is before considering that sometimes the device is held in hand, sometimes in a pocket and sometimes inside a car[14]. Due to these reasons, the measurements are expected to have higher variance and less reliable results than specialized drive testing. Furthermore, as the location is received from the GPS of the UE, there is also innate variance in the locations: While cell phone GPS can get an accuracy of about 2 meters in an open field, actual accuracy in an urban environment or inside buildings is not nearly as good[27]. Additionally, as only observations with available GPS location are used it can create a sampling bias as it is quite easy to lose GPS connection inside buildings. Instead, the higher variance is compensated by the sheer number of observations and massive density at locations where there are the most devices (and thus where the accuracy arguably matters the most).

3.2 Prior Considerations

This section deals with practical problems that need to be solved before any model can be fitted or results obtained. This section also includes a discussion about the behaviour of the data that leads to assumptions that might have a large impact on the outcome.

The section starts with a quick list of tools that were used in the implementation, including the programming languages and packages of the main predictive models.

After the tools are introduced we'll take a deeper look at what can be done to the data in order to get good and reliable results. Multiple preprocessing steps will be used to reduce the data heterogeneity and to facilitate efficient machine learning.

Finally, it has to be decided how to tell apart a "good" result from "bad" one.

Measurement	Explanation	Notes
RSRP	Reference signal received power of current cell (dB)	No missing values
RSRQ	Reference signal received quality of current cell (dB)	No missing values
UE power headroom	How much more power the UE could use for transmission (dBm)	Only observed on part of the network
UE uplink SINR	Uplink Signal to Noise + interference ratio observed by the UE (dBm)	Only observed on part of the network
Latitude	Latitude reported by GPS (degrees)	No missing values
Longitude	Longitude reported by GPS (degrees)	No missing values
Timestamp	When was the report received (ms)	No missing values
Distance	Distance from the serving cell to UE, calculated from GPS location and network topology (meters)	No missing values
GPS confidence	Some value if GPS location is corrupted, missing otherwise	All non missing are filtered out
Timing advanced	Timing advance value transformed to distance (meters)	Distance calculation that is not based on GPS
EARFCN	Downlink EARFCN of the used frequency	Used to distinguish between different frequency channels

Table 3.1 *Different measurements available in the data with basic explanation and some notes related to missingness or use case of the measurement*

This is not quite simple once the requirements from the different predictive models and computational feasibility is included. Thus the considerations lead to a rather long discussion about how the result could be made better by optimizing prediction locations, which is separated as its own section.

3.2.1 Tools

The implementations are mainly done in R [32] and some parts with Python [31]. Specifically the nearest neighbour search is performed with the *RANN2* [18], locally approximate Gaussian process regression with the *laGP* [17] and fixed rank kriging with the *FRK* [47] R packages. Neural networks are implemented with the *tensorflow* Python library [24]. Visualizations are created with the *ggplot2* R package [45]. While neither R nor Python is very fast for processing massive datasets, the good support for efficient C/C++ libraries make them fast enough and suitable for proof of concept and research work.

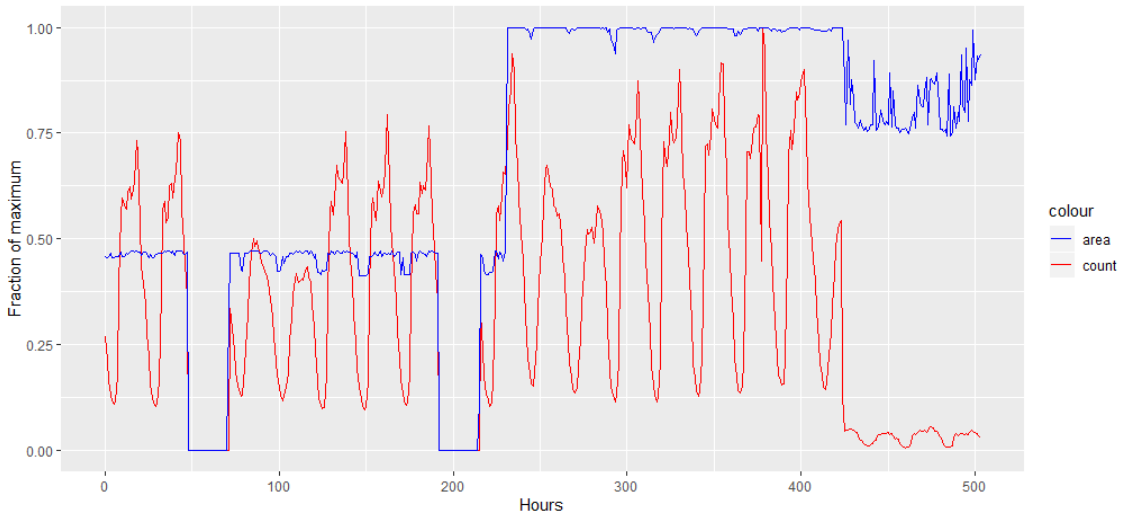

Figure 3.1 *Shows the number of observations and the area of a rectangle bounding all the observations as a function of time. Both values are given as fractions of the maximum observed values, aggregated over one hour time period. The day-night cycle of the observations is clearly visible as well as the two outages.*

3.2.2 Pre-Processing

First off to limit the amount of error given by the GPS location, all observations that report any problem with the GPS are removed. This covers surprisingly many observations, 20 million out of the 60 million. After removing those 20 million there are still 27 cases where the timestamp is several years off and those obviously get deleted too. Finally, there are 16 cases that report distance to the serving cell to be more than 6000 km at the same time the GPS seems to be correctly placed, either indicating data corruption or incorrectly defined cell, to avoid any surprises later on the ones reporting too high distance are deleted but ones having missing value are kept. Not so much of an error in data but more of an outlier, there is a total of seven observations that are using different frequency band than the most commonly used three and are excluded from the analysis.

After these cleaning steps are performed 42.6 million observations are left, which is still plenty when compared to other similar studies.

3.2.3 Exploratory Analysis

When the number of observations and area they cover is plotted on a timeline, as seen in figure 3.1, it is easy to see that there are three regimes and two outages in the data collection. The first regime could be characterized as having a very high volume of observations coming from a medium-sized area. The second regime more than doubles the area with a slight increase in the observations and the third regime has a very large covered area despite a very small number of observations.

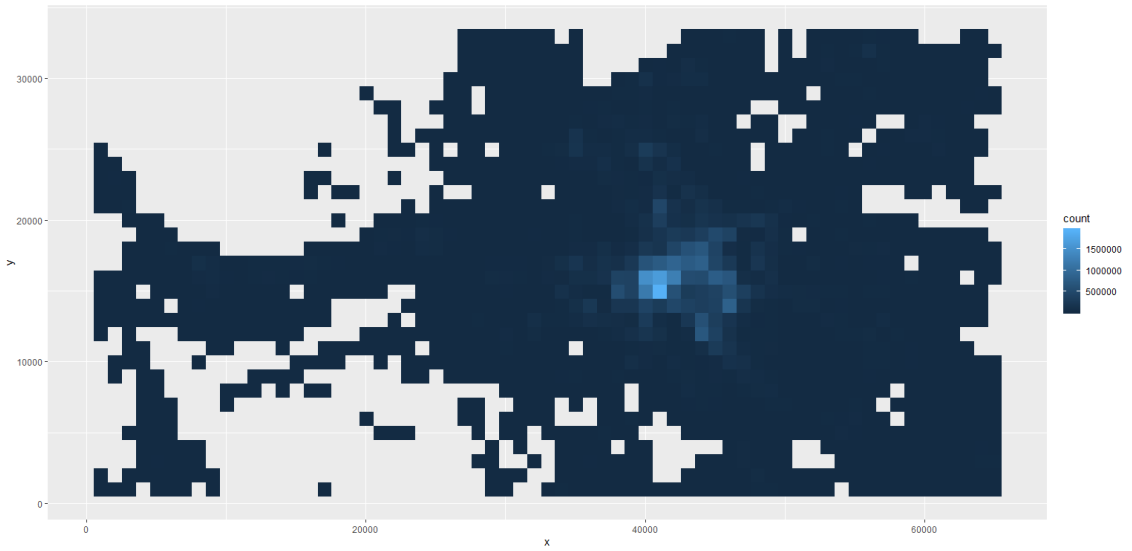

Figure 3.2 *Shows the number of observations in each 10 km^2 square after cleaning but before further splitting, thus the total number of observations in the image is 42.6 million. The x and y-axis are given as meters from the bottom left corner. Grey areas do not have any observations.*

It was mentioned in section 3.1 that most of the observations are clustered to the centre of the city. To get an idea just how clustered the data is, it can be divided into 1 km^2 squares and count the number of observations found in each square, the result can be seen in figure 3.2. The densest square has 1.95 million observations giving it a density of 1.95 obs/m^2, more than twice the previous study with the highest density and 64 times the average density in this data. The flip side of this is that 36.9 % of the squares do not have any observations and 54.1 % have less than 100 observations.

To assess whether the assumption about temporal dependency has any ground, it would be sensible to plot one of the measurements of interest as a function of time. For this purpose the data is divided into 15 minute time windows from which a mean value is calculated to make any trend or seasonality more visible from the noise, the resulting plot for RSRP is given in figure 3.3. RSRP clearly has periodic behaviour, since a change in environment or human behaviour cannot reasonably explain the change it is assumed to be caused by a change in network parameters. While the change is very quick and easy to notice in the plot, a 4 dBm change is not very large when the whole range of values is considered. Nonetheless using temporal dimension in predicting the value can be expected to give some benefit.

Between the three most common frequency bands the observations are split as A) 59.3%, B) 26.6% and C) 14.1%, where band C has the highest frequency and B the lowest. As the used frequency band affects the signal propagation it would be sensible to consider the different bands individually when estimating the signal qual-

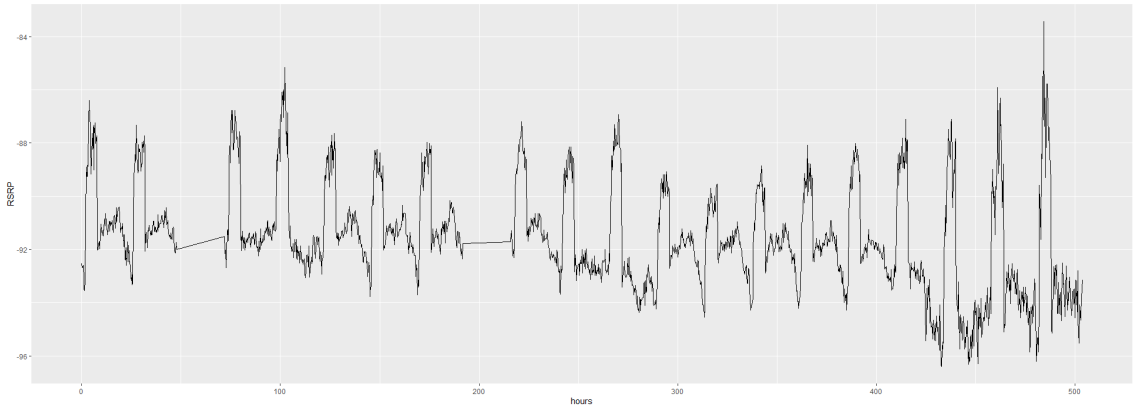

Figure 3.3 *RSRP averaged over 15-minute timeslots in the whole network. The value seems to change about 4 dBm within one timeslot, this could be caused by a change in network parameters such as power-saving during the night.*

ity, just as they are considered separately in the mobile network planning phase. Furthermore, the cells for different frequency bands could be planned differently yielding contradicting measurements at a given location. This will limit measurements available for creating a single map. It will be interesting to see how the maps will be formed with a different number of observations and with a different spatial distribution.

To give the predictive models an easier environment and to make the plots more sensible the latitude and longitude that are given in degrees will be changed to be measured in meters instead. The conversion is done by calculating distance, separately for latitude and longitude, from ”bottom left” corner of a square that is bounding all the datapoints using the Haversine distance function.

The observations are split into three different sets that will be called training, evaluation and test set from now on. The training set consist of about 70% of all the observations and the rest will be in the test set. The test set is composed of observations that come after the training set in time, so the test set is not just a sample from the data that is not found from the training set but it is also separate in the time dimension. The evaluation set is a 10 000 datapoints randomly sampled from the training set, the training points sampled to the evaluation set are removed from the training set. The evaluation set is used especially for finding the hyperparameters. This also means that the hyperparameters will be estimated completely with the data that comes before the test set when the time domain is considered. This could have some consequences in the spatial prediction case where the time domain is not considered and instead it is expected that training and test sets come from the same distribution that is not necessarily the case. In the other hand, it will give a more realistic view of how the models perform since no information is leaked from the test set.

3.2.4 Measuring Map Quality

The problem definition and data gives rise to multiple valid ways to calculate the error of the result. In the end, the aim is to generate the most accurate possible presentation/prediction map but especially when finding values for hyper-parameters it would be beneficial to limit the computation required. I will try to portray these different ways to measure quality and their strengths and weaknesses. Each quality measure will optimize slightly different thing, also not all methods of calculating quality are available to all methods so it is important to find at least some metric that can be used to compare the different methods.

Given that the data has individual measurements, the simplest possible quality measure would be to take a testing sample from the measurements and then predict the value at the location of each measurement. The prediction error would then be the mean error between the predicted values and measurements. Despite being simple it has a few problems: Neural networks (especially convolutional ones) cannot produce prediction at an arbitrary location but only on a predefined grid, same is somewhat true for FRK procedure which cannot produce predictions for a finer grid than what was used for training. Additionally, the original measurements have quite a bit of noise so it would be beneficial to aggregate multiple measurements together to smooth out the outliers. Furthermore, we have the problem of sampling the test measurements to use, random sampling would sample more measurements from locations with more measurements overall, which seems reasonable but it could leave some other areas untested. On the positive side, it is easy to control the running time by changing how many locations get sampled.

Trying to improve on the problems in the earlier approach, the next natural step would be to generate the predictions on some grid and then use the same prediction for all the test samples falling to that grid square. This would make the number of predictions to be independent of the number of test samples. Also, it allows the ANN and FRK to be used just the same as k-NN and laGP. On the downside, this approach does not address the noise in the test samples and using a grid will add a new source of error to the predictions. This new source of error is caused because the prediction is made at the centre of the grid square and not at the exact location of the test sample. Throughout the thesis, I will be calling this type of an error an quantization error. The amount of quantization error will depend on how fine of a grid is used, finer grid will produce more accurate results but it will also require more computing.

To smooth out the noise in data also the samples can be aggregated over the same grid that was used for the predictions. The degree of smoothing could be controlled by leaving out grid squares that do not have enough samples. This is

essentially what will be done to the training samples used by the ANN since the kind of ANN used in the thesis cannot work with point data as an input.

It is also interesting to consider minimizing the quantization error or in other words finding the optimal grid to use for the map generation. When combined with the computation considerations the minimization task would be to provide the lowest possible quantization error with fewest possible grid squares. If we also define that the whole area needs to be covered by the grid the only option is to consider an irregular grid of different sized squares. Predictions on such irregular grid would be easy to provide with Gaussian process regression and k-nearest neighbours interpolation but the neural network would again struggle and FRK would probably require new training on the new grid. Building at least in some sense optimal grid is seen so beneficial that is studied more thoroughly in section 3.2.5.

Extending the models to time dimension adds more combinations to consider as time can also be taken as an exact measure or discretized. Either exact or discretized time dimension can be combined with both exact or discretized spatial dimension for a total of four combinations. Also for finding the optimal grid it has to be considered if the grid needs to be static in time or if the grid structure can change from one step to another. If the testing samples were aggregated to the grid it would make comparing spatial and spatio-temporal results extremely hard as the aggregation would be different between them.

Based on the above consideration the final results will be given in three categories:

1. Uniform grid, the predictions will be made on a uniform grid and the error will be calculated for each test sample based on which grid square the sample falls to. Valid for all the models.

2. Adaptive grid, same as above but the grid will be optimized so that it adapts to the data. Details of the optimization will be given in section 3.2.5. Valid for k-NN, laGP and FRK.

3. Exact location, a random subsample will be taken from the test samples and predictions are calculated at the exact locations. Valid for k-NN and laGP.

This far the actual error metric is left unspecified as the discussion above is not dependant on the exact metric used to measure the error yet some consideration is required to pick suitable error metric. We expect the ground truth values to be noisy with even quite large outliers and as aggregating the test samples to smooth the values will not be done, a metric that does not exaggerate the error based on magnitude is preferred. One such metric that is commonly used is mean absolute error (MAE), which will be used in this thesis due to its robustness to outliers.

3.2.5 Optimizing the Grid

All the literature sampled for section 2.2 work with areas that are a lot smaller than the approximately 30km x 60km area used in this thesis. Aiming for similar-sized squares in the measurement grid would inevitably lead to computational problems, using the 25m x 25m square size that was the largest among studies cited in 2.2 would require approximately 3 million squares to fill the area. For comparison, most squares used in the cited studies is less than 2 million found in the paper by Sohrabi and Kuchn[41]. For this reason, optimizing the grid used for prediction is necessary, otherwise either the computations would take longer than what is acceptable or only a subset of the area could be used.

For now, we focus on finding the grid in two-dimensional spatial space without considering time, the time dimension will be considered towards the end of this section.

A prominent option for creating a grid with smaller computational requirements comes from numerical analysis where this kind of irregular grid creation is called adaptive mesh refinement [6]. In numerical analysis/computational physics the needs are somewhat similar as in this thesis; get more accurate results without spending the computational resources required by the uniform fine-grained grid. While the procedure described by Berger et al. is complex, the basic idea is simple; have a coarse grid and refine the areas that fulfil some criterion to include finer details [6]. It is possible to take large shortcuts in implementing adaptive grid refinement for this thesis. Since we do not need to care about interactions between the cells and also because in this thesis the functions are valued in the middle of the cell and the error is expected to rise as distance from the evaluated location gets larger it seems sensible to limit the grid to contain square-shaped cells instead of arbitrary rectangles.

So for the purpose of this thesis, the grid refinement is done by taking a coarse grid of equally sized square cells and splitting the grid cells that fulfil some criterion into four equally sized smaller squares. Given this easy to implement grid creation method the problem is to find a criterion for splitting the cell so that the quantization error is optimized with respect to available computational resources.

A good start for optimizing the grid is to minimize the amount of computation spent on the areas that have almost no observations at all. As already mentioned in section 3.1, the observations are clustered around the centre of the city and there is plenty of space without any measurements. Computing the evaluations on a fine-grained grid on an area that can be estimated poorly at best is a waste of computational time. Thus it makes sense to impose some kind of density limit on the grid, if there are not enough observations for fine-grained estimation why bother

calculating fine-grained estimation. This is easy to implement using the k-nearest neighbour search; search for k-nearest observations and use the distance of furthest away observation to determine the density at that location, or constrain it so that all the k-nearest observations must be within some predefined distance (for example size of the cell) for the cell to be split.

However, it is difficult to define one value for the required density as that will be dependant on the data and possibly also on the prediction method that will be used on top of that grid. Additionally, the density alone does not take into account whether the splitting is necessary or if the same accuracy could have been obtained with a coarser grid. Furthermore, calculating density only based on k-nearest observations would give an estimate of the density only at the centre of the square, there might still be some part of the square with very high density making it a candidate for splitting. For these reasons it might be more useful to use k-nearest search to check if there is a very high number of observations within the square and have a secondary criterion for the variability of the observations.

Using such a variability criterion is difficult in practice as the data is expected to be noisy and vary over time, additionally, finding all the observations that are within a square takes time proportional to the total number of observations so it would be beneficial to reuse the nearest neighbour search results as an approximation. The effect of the noise can be reduced by using a robust variance statistic such as interquantile range or median absolute deviation that give high values only when a reasonable portion of the samples are far away from each other unlike variance that has a problem with outliers. Additionally, the nearest neighbour reuse can be done by using two different limits for how many points need to be within the square. Let us name these limits for the number of points as upper limit u and lower limit l. If more points than the upper limit are found, the square is considered very dense and will be split regardless of the variance, this means we have to search for u closest points to the grid centre. In the other hand, if enough points are found to satisfy the lower limit (but not the upper limit) the variance can be considered. And finally, if there are fewer samples than l within the distance limit the square will not be split. Now the variance estimator can be calculated based on all the datapoints within the square without explicitly searching for them by reusing the search results obtained in earlier step when checking if the number of datapoints satisfies upper limit.

Berger et al. consider the grid to change over time because the in physics simulation the area of interest often moves and better accuracy to computation ratio can be obtained by changing the grid, the downside is a more complex algorithm and data structures as the squares (or rectangles in Berger et al.) need to be splittable and mergeable depending on the situation in the simulation. In this thesis keeping the map the same over time is seen as more beneficial as it will allow comparing

different timesteps directly with each other. Of course, this will limit the benefits, both computational and accuracy, that can be gained by using adaptive grids since one dimension will be excluded from the adaptivity. On the positive side, if the algorithm only has forward steps (splitting the squares), there is no reason to have a complex data structure that keeps track of the splits unlike if the algorithm also had backward steps (merging the squares).

Based on the above considerations the proposed algorithm would have rather simple steps:

1. Start with a coarse grid of squares.
2. Find u nearest neighbours for each square centre.
3. If all u nearest neighbours are within the square, consider the square to be so dense that it needs to split regardless of the variability.
4. If not all u nearest neighbours are within the square but still at least some $l < u$ are, calculate robust variability estimator (ie. inter quantile range) based on the points that were within the square and tag the square for splitting if the estimator is larger than some *limit* v.
5. Consider squares that were not tagged for splitting to be part of the final grid.
6. Split squares that were tagged for splitting into four smaller squares and continue with this grid again from step 2. Repeat until there are no more squares to split or minimum size for a square is achieved.

The above steps lead to naive approximation for the optimal grid that is very fast to calculate using nearest neighbour search. However, there are four parameters that need to be decided, u, l, v and the minimum size for a square. The minimum size of a square is required so that the algorithm stops at some point even if there is a large number of points in the exact same location. All of the parameters are more or less data dependant so optimal parameters could be found by optimizing them within the training dataset using the method that is shown in section 3.3.1.

The hyperparameter optimization shown in 3.3.1 requires some error function that gets optimized. A simple way of finding error metric for the grid is to actually use it for predicting. And since we are already searching nearest neighbours when creating the grid why not use them for the error metric calculation as well. Earlier it was defined that the grid should minimize quantization error, the quantization error can be found by predicting the test samples at the exact locations and then by using the grid and finding the difference between the two. However, naively optimizing quantization error leads to an infinitely fine-grained grid which leads to

computational problems. So there needs to be some way to balance between the number of locations to evaluate and the quantization error. Another way to look at the quantization error is to create a uniform grid with (approximately) the same number of squares as the adaptive grid and see how much better the adaptive grid is than the uniform grid. As generating finer grid would help both the adaptive and uniform grid and as the quantization error of both grids approaches zero as evaluations approach infinity, it is unlikely that this error function would generate an extremely fine-grained grid. However, there is a chance that this error does not optimize the reduction in quantization error of adaptive grid, but rather tries to maximise the quantization error of the uniform grid by using as few evaluation locations as possible. To create an error function that does not have immediately obvious pitfalls the former two can be combined; take the difference between errors of adaptive and uniform grids and reduce the quantization error of the adaptive grid. When this error function is optimized it gives a better score for those grids that have the biggest advantage over the uniform grid but at the same time, it penalizes grids that are bad overall.

As the quantization error should be similar across different prediction methods, only one adaptive grid is created and then used for all prediction methods that can use irregular grids. Calculating the error function for the adaptive grid creation requires predictions to be provided for the exact locations, adaptive grid and uniform grid, these predictions will be calculated with k-nearest neighbours interpolation using 100 nearest neighbours with no inverse distance weighting. Again, the prediction method selected here is not expected to hugely affect the created grid and k-nearest neighbour interpolation with relatively small k is fast to compute. Furthermore, the sides of the squares in the initial coarse grid are defined to be 8192 meters long, the method should not be sensitive to the exact value so a rather large value is chosen, also the value being a power of two means that each of the subsequent smaller squares will have side length that is a nice integer.

3.3 Model Fitting

All the predictive models considered in the thesis require some optimization: neural network weight parameters are found entirely trough iterative training process but some hyper-parameters are left for the user, locally approximate Gaussian process regression has some parameters that are found with maximum likelihood estimation but especially the approximations have hyper-parameters also for the user to specify, k-nearest neighbours interpolation does not have a fitting process in the same manner but for optimal accuracy, hyper-parameter tuning is required. First, in this section, a method for optimizing these hyper-parameters is introduced and applied to the predictive models and to the generation of the predictive grid.

The different models used in this thesis have very different fitting processes. K-NN and laGP are mostly based on the data itself with no or minimal training done before the predictions can be obtained. For these two, most of the time before predicting is spent on creating an efficient data structure that can be used for fast nearest neighbour queries. However, these two algorithms have some parameters that need to be set before one can start predicting and these hyper-parameters can be optimized to give the best results in this particular data. For FRK and ANN, it is a different story as both of them require long training process before any prediction can be made. And even though FRK and ANN also have hyper-parameters that could be optimized, the long training process before obtaining any predictions makes the parameter search slow and computationally much more expensive than in k-NN and laGP case. Because of this the FRK and ANN hyper-parameters are left at "default" values, for FRK this means the defaults of the FRK library and for ANN it means the values that were used in the paper by Teganya et al. Since increasing the learning capacity of ANN model should give better results, given enough learning samples, also a larger model is tried where the number of nodes in the "low" dimensional layer is significantly increased.

3.3.1 Hyperparameter Optimization

Finding values for the hyper-parameters is important to make sure that the methods perform as well as possible, however, all of the methods in this thesis are computationally demanding and finding optimal parameters often requires testing many parameter combinations. Thus limiting the number of combinations to try is beneficial, one method that tries to cleverly choose the parameter combinations to test in order to limit the total number of combinations to test is called Bayesian optimization [40].

Interestingly, the Bayesian optimization can use Gaussian process, as described in section 2.3.2, to provide a functional form for how the test metric is expected to behave under different combination of training parameters, this functional form is known as a surrogate model. The training parameters are seen as the locations x and the test metric value as response y. The new set of parameters x_* is selected so that they are expected to be better than the best parameters found so far. The idea is that it is faster to evaluate the surrogate model than the predictive model, but that the surrogate model can offer some guidance of where to try next as well as getting more accurate at each new tested parameter combination.

For the purpose of estimating hyperparameters, the training results are validated against the evaluation set, the evaluation set is used to avoid leaking information from the actual test set and to allow less computation than evaluating all the locations in the test set. For all of the optimization tasks, the Bayesian optimization is

started with five randomly chosen parameter combinations, after that the Bayesian optimization chooses the next training location based on Gaussian process upper confidence bound [40] for the next 100 parameter combinations, finally the best combination is chosen to be used for the final test.

The error function for the hyperparameter optimization in adaptive grid case is described at the end of the section 3.2.5,k-NN and laGP error function is computed by evaluating the methods on the exact locations given in evaluation set (the first method described in section 3.2.4).

The different frequency bands found in the data need to be optimized separately because the data characteristics might vary, it can then be assessed if the parameters are meaningfully different or if the parameters could have been transferred directly. This is especially important for the adaptive grid as the different frequency bands have a different number of observations and the adaptive grid construction process as described in 3.2.5 is sensitive to the absolute number of observations.

The result of the optimization for laGP and k-NN can be found from table 3.2. For laGP the parameters with the strongest effect are *end* and *selection start* as those limit the most how the local construct is created. Considering that all frequency bands have low *end* value I suspect that *selection start* is never even used and both *length-scale start* and *nugget start* are both optimized locally with MLE anyway. With k-NN the optimization makes it clear that no sample should be dropped from the calculation no matter how far it is, meaning that *min points* parameter is never used. The k and *weight* parameters have meaningfully large differences between the frequency bands so it seems unlikely that the parameters could have been transferred from one band to another.

And as expected, there are large differences in the parameters that optimize adaptive grid creation on different frequency band as is seen in the table 3.3.

While doing hyperparameter optimization for laGP it was noticed that parameters that resulted in very few observations were chosen as best, also choosing nearest neighbours over ALC criterion-based observations were seen to provide better results in one frequency band, note that it was already seen in figure 2.2 that ALC-ray does not follow ALC results in this dataset and was not used. This is somewhat contradictory with earlier experiments done by Gramacy, where ALC resulted in better accuracy out-of-sample. To study this phenomenon better, only the selection criterion and number of selected observations were varied and everything else was fixed. The frequency band used in this test is B-band, the same that was seen to benefit from choosing more nearest neighbours and less ALC-criterion based observations.

The results of the deeper look can be seen in the figure 3.4, ALC-ray criterion was added to see how it performs (in this case) despite giving widely different results than ALC in earlier tests. To create the visualization the *selection start* parameter was

Table 3.2 *Functions that are optimized with Bayesian optimization. Contains two predictive models; locally approximate Gaussian process regression and k-nearest neighbours interpolation. The* range *column give the range of searched parameters.*

Function	Parameter	Range	Optimized value	Explanation
laGP	length-scale start	100 → 50000	A)4018, B)100 C)100	Start value for MLE of length-scale
	nugget start	0.001 → 8	A)0.001, B)0.07 C)0.09	Start value for MLE of nugget (noise variance)
	selection start	1 → 14	A)1, B)14 C)5	How many first observation locations are chosen using nearest neighbours
	close	100 → 3000	A)100, B)1566 C)3000	How many closest observations are sampled to be considered in the selection process
	end	15 → 99	A)16, B)16 C)15	How many observations are selected to final construct total
k-NN	k	10 → 1999	A)410, B)156 C)1070	How many nearest neighbours are used
	max distance	10 → 20000	A)20000, B)19390 C)15250	Maximum distance between from the predicted location to a point
	min points	1 → 9	A)9, B)7 C)2	Minimum number of points to use even when those would be further away than *max distance*
	weight	-5 → 5	A)-2.93, B)-2.53 C)-3.28	IDW-interpolation weight value, exponent value for distance

Table 3.3 *Parameters for the adaptive grid construction function. The* range *column give the range of searched parameters and* optimized value *column gives the value after optimization in the different frequency bands.*

Function	Parameter	Range	Optimized value	Explanation
adaptive grid	maximum points	1000 → 5000	A)1000, B)1337, C)2301	Number of datapoints within the distance from center to the edge (Euclidean distance) that will trigger a split
	minimum points	40 → 999	A)611, B)561, C)527	Number of datapoints within the distance from center to the edge (Euclidean distance) that can trigger a split
	variation limit	3 → 30	A)9.28, B)10.61, C)3.00	Amount of variation required within the points found within the square to trigger a split when number of points is between maximum and minimum
	minimum cell size	16	16	Minimum side length of a cell
	start cell size	8192	8192	Side length of cells in the crude grid used to start the algorithm

Table 3.4 *Predicitive models that were not optimized with Bayesian optimization along with some of their parameters.*

Model	Parameter	Tested values	Explanation
ANN	Size of the lowest dimensional layer	64 (default), 512 (large)	Controls the overall complexity of the model
FRK	Number of basis functions	depends on other parameters	The basis functions are constructed by the *auto_basis* function from the *FRK* package. However there is multiple ways to affect the number and form of the basis functions. More elaborate explanation in section 3.3.2.
	resolutions		The number of different resolutions used in the basis functions
	regular basis functions		How many basis function is placed regularly, 0 for irregular placement

fixed at 4, *length-scale start* and *nugget start* were fixed to the best parameters found by the Bayesian optimization (see table 3.2 for parameter explanations). From the image, it is clear that all the criteria perform best with very small local construct, meaning only a few points selected, with the minimum error being at 20 for all criteria. This is pretty close to the Bayesian optimization results where the best error was attained with 16 locations, 14 of which were chosen with nearest neighbours and 2 with ALC. Computation time-wise the ALC is the most expensive of the three criteria, although at only a few selected locations there is hardly any difference. Despite the results gathered here that show that only using nearest neighbour could provide better results in this particular case, the ALC criterion was used in all computations as other frequency bands did not show similar behaviour.

3.3.2 Spatial Models

Four different spatial models are considered in the thesis: k-nearest neighbours interpolation with inverse distance weighting, fixed rank kriging, locally approximate Gaussian process regression and neural network autoencoder. K-nearest neighbours interpolation represent the very fast and simple method, FRK will provide a global Gaussian process regression model and be the baseline to compare with other papers, locally approximate Gaussian process regression is a novel approach in this context and neural networks are presented as a curiosity towards a different kind of modelling

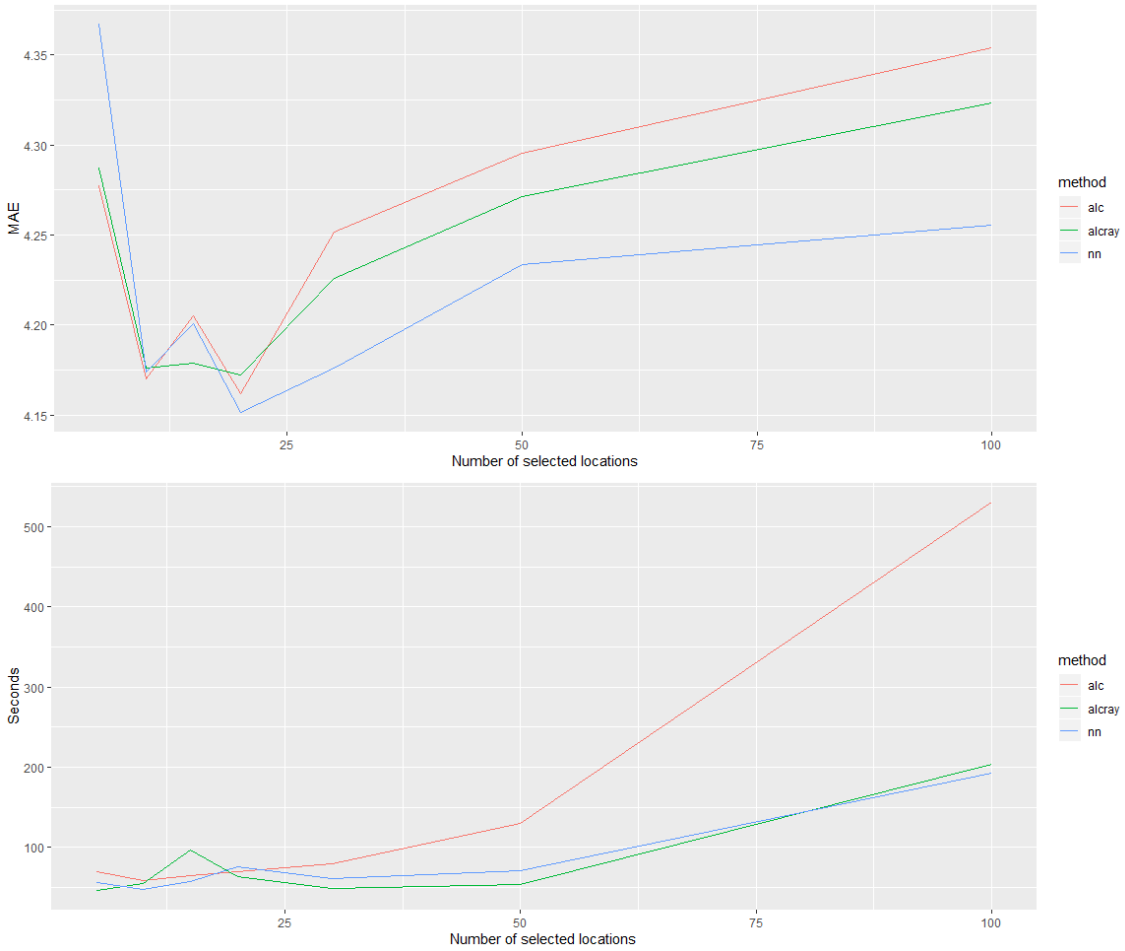

Figure 3.4 *Comparison of ALC, ALC-ray and nearest neighbour selection criteria in the frequency band B data. Top image shows mean absolute error as a function of how many locations were chosen for constructing the laGP model. The second picture shows time spent creating a prediction for 1000 predictive locations as a function of the number of locations chosen for constructing the prediction for each predictive location.*

approach.

As noted in section 2.3.2 and in table 3.4 the FRK model is very dependant on how the basis functions are chosen. The number of the basis functions determines the computation time and mostly the flexibility of the model. The main parameter affecting the number of functions is the number of different resolutions used, Cressie and Johanneson emphasis that using multiple different resolutions for creating the basis is important and three different resolutions are chosen as the default in the FRK package. Additionally, the FRK package allows pruning some of the basis afterwards. Furthermore, the basis functions can be placed on the grid on regular or irregular intervals, the irregular spacing is based on the data locations so that denser regions have more basis functions. Now especially the irregular spacing of the functions seems like an interesting idea considering that the spatial distribution of the observations forms clusters.

Training the ANN autoencoder differs from the other models, for the autoencoder the training data is first aggregated on the 512×256 regular grid. To generate more training samples and to keep the input size as small as possible the grid is then divided to 32×32 sized smaller grids which also matches with the size used by Teganya et al. To use these smaller grids for training, some of the observed values will be masked (hidden) from the input and these masked values are then used to calculate error from the output.

Thus also in a purely spatial context where training and testing data are seen as just two separate pools of data without regard for time dimension, the ANN assumptions differ from the other models; while other models fit a function to the training data and the accuracy of that function is then tested on the training set, the ANN autoencoder as described in 2.3.3 learns a relationship between the input (where some of the values are masked) and expected output, so in practice, it would make sense to give input values (again masked) from the test set and see if the ANN has been able to generalize from the training to the testing set. However, this makes a large difference between ANN and other models as other models do not get any information from the test set before having to make the prediction. If one wishes to keep the results from ANN and other models exactly comparable, then in the test phase the input to the ANN needs to be from the training set. This is what is done in the spatial test, but arguably this defeats much of the purpose of using ANN in the first place where the point is to use a different kind of modelling approach to be able to update the prediction based on current data. This problem is not present in the temporal forecasting case as input to ANN are from earlier timesteps regardless if that timestep is from training or testing set.

3.3.3 Spatio-Temporal Extension

For k-nearest neighbours interpolation and Gaussian process regression, the temporal aspect can be considered by adding the time as the third dimension in the distance calculation. Since spatial distances and temporal distances are not measured in the same units there needs to be some scaling between the spatial and temporal dimensions. The scaling can be seen as extra hyperparameter when doing the optimization for k-NN. And in laGP case, a separable kernel is used which allows different length-scale for different dimensions. It means that each predictive location has its own scaling allowing even more flexible model.

However, if either k-NN or laGP model is to be used for forecasting into the future, there needs to be some stationarity in the time dimension. By making an assumption that each day behaves approximately similarly we can archive stationarity. This stationarity allows making a prediction for one day and using that prediction the next. Additionally, leveraging this assumption would allow using all the training

data from different days together to predict the value at a specific spatio-temporal location. This kind of naive assumption of stationary would not take into account longer-term trends or patterns that do not repeat daily, neither could it react to unexpected change to update the forecast.

Neural networks are in this case a lot more flexible choice as shortly discussed at the end of chapter 2.3.3. Training a neural network with previous timesteps to forecast next timestep would allow updating the forecast based on the most recent available measurements without having to rely on temporal stationarity. The downside is increased complexity of the network increasing the training time and training sample requirements.

3.4 Test setup

All of the prediction methods can be tested on a regular grid, so that will be used to compare different models. To make the results universally comparable between all combinations a fixed grid of 256×512 cells are used, this leads to about $125m \times 125m$ cell size. Since k-NN and laGP can additionally be evaluated on arbitrary locations, counting error on exact observation locations and on the adaptive grid is added for these methods to assess the strong points and weaknesses of these models more thoroughly and to see how well the adaptive grid fares against the uniform grid. FRK is learned separately for the regular and irregular grids to see if the method benefits from the adaptivity. Finally, to get a good comparison of how much information the spatial context is adding, a naive mean model is used as a comparison, the model predicts mean of the training set for all test samples.

The test set is somewhat large so for exact location prediction a subset needs to be selected, 100 000 datapoints will be randomly selected to make it more comprehensive than the evaluation set but still small enough to be feasible.

The frequency bands need to be tested separately as well, this will give some information about how many measurements are required for accurate mapping. This means that a total of 18 models (six predictive models on three different frequency bands) need to be evaluated for the spatial case.

3.4.1 Modifications for Spatio-Temporal Case

The runtime of the computations is significantly increased because there are many more locations in the grid that need to be predicted so the spatio-temporal predictions are reported only for one frequency band. To be able to use as many observations as possible, the frequency band A is chosen for the experiments.

Also, the length of single timestep needs to be decided, this decision can have a large effect on the outcome as well as the time required for fitting the models.

Selecting a short length for the timestep will mean that there are more steps but each of them is more sparse. In the other end, choosing long timesteps will provide fewer steps total, but it might mean that the whole time span is not covered by the one prediction that is made at the midpoint of that span. I decide to use one hour window as a timestep, it offers a good tradeoff between the number of steps and sparsity and being able to detect changes on one-hour granularity seems like a worthwhile goal. Using exact spatio-temporal locations naturally avoids all the problems of choosing the length of the timestep, but again only k-NN and laGP (and naive) models can use exact locations.

Moving to spatio-temporal predictions, the laGP model will begin using "separable kernels" allowing it some extra flexibility compared to k-NN. The extra flexibility is gained by estimating the length-scale parameter separately for each of the dimensions. Furthermore, it avoids separately estimating correct scaling for the time dimension as the estimation is done as part of the length-scale estimation now, the downside is slower computation.

The ANN will be modified for the spatio-temporal case so that the input consist of data from eight subsequent timesteps and the expected output that the network is supposed to predict is the map at the next timestep after the eight given as input. Also, the structure is modified so that the encoder part will reduce the input dimensionality from $8 \times 32 \times 32$ to $1 \times 32 \times 32$ and the decoder part can be the same as before.

4 Results

The results for spatial prediction given in the table 4.2 show consistent difference among the models across all frequency bands and prediction location types. In every category, the order between the top three models is the same. The best result was obtained with inverse distance weighted k-NN, a close second being laGP, the third one was FRK that was the last one to beat the naive mean model. And finally, after the naive model, there are the two neural networks that clearly couldn't learn well enough to be competitive with the top three. The smaller one of the networks did slightly better than the large one and was able to beat the naive model in frequency band C.

When reading the results in another direction it becomes clear that using an adaptive grid was beneficial for all the models that are able to use it and naturally using the exact locations was the most accurate. This is a very promising result for the adaptive grid when contrasted with the table 4.1 that shows the number of predictions that are done using the adaptive grids on different frequency bands. The number to compare these against is the uniform grids $256 \times 512 = 131072$, so not only was the adaptive grid more accurate but it also required three to eighteen times less computation. K-NN was the most accurate in every category, but it also looks like it gained the most from using an adaptive grid. Interestingly the trend did not continue when exact locations were used as laGP was capable to produce almost identically good results at that point.

Turning to the spatio-temporal results that are reported in table 4.3 the general theme looks very much the same. K-NN providing the best results, this time with a larger gap to the laGP and ANN coming last. FRK was not tested due to too many problems with the library implementation (or user error) when used for spatio-temporal data, a sad loss because the library did have functions specifically for spatio-temporal case. The ANN results given here are not really comparable to the spatial case as the problem definition has been changed from interpolation to forecasting. Considering that the problem became a lot harder the loss is accuracy is not that bad, then again it would still have been better on average to forecast the

Frequency band	Number of predictive locations	
A	38 312	
B	7 225	
C	26 735	

Table 4.1 *Number of predictive locations in the adaptive grid for each frequency band. Uniform grid has 131 072 predictive locations*

Model	Frequency band	Uniform grid MAE	Adaptive grid MAE	Exact MAE
Naive mean model	A	8.86	8.78	8.78
ANN (small)	A	9.07	NA	NA
ANN (large)	A	10.06	NA	NA
laGP	A	7.02	6.28	5.5
k-NN	A	6.9	5.9	5.49
FRK	A	8.24	7.95	NA
Naive mean model	B	7.46	7.47	7.46
ANN (small)	B	10.37	NA	NA
ANN (large)	B	12.46	NA	NA
laGP	B	6.66	6.04	5.36
k-NN	B	6.39	5.66	5.17
FRK	B	7.19	6.79	NA
Naive mean model	C	11.41	11.39	11.49
ANN (small)	C	11.29	NA	NA
ANN (large)	C	12.02	NA	NA
laGP	C	7.77	7.08	5.79
k-NN	C	7.41	6.55	5.78
FRK	C	9.15	8.8	NA

Table 4.2 *Results for all the compared models in the three different frequency bands and with three different ways of defining the prediction grid.*

Model	Frequency band	Uniform grid MAE	Adaptive grid MAE	Exact MAE
Naive mean model	A	8.78	8.78	8.77
ANN (small)	A	9.81	NA	NA
ANN (large)	A	10.8	NA	NA
laGP	A	6.8	6.65	6.31
k-NN	A	6.61	6.04	5.87
k-NN (modified time)	A	6.72	6.11	5.91

Table 4.3 *Spatio-temporal results for the models that were able to be tested in the spatio-temporal setting.*

mean of the training data than use ANN.

While the general results are not very interesting, comparing the spatial and spatio-temporal results show that adding the temporal dimension gave very little benefit or even hindered the accuracy of the models. The only case where accuracy is gained is the uniform grid case and even then the benefit is a lot smaller than moving from the uniform grid to the adaptive grid. In the adaptive grid and exact location case the accuracy is actually lost by adding time, an unexpected result that needs a little bit more studying.

The first guess of why the accuracy was lost is that I have specified the time dimension wrong. For k-NN and laGP the time is taken as seconds from the previous midnight, this has a clear disadvantage at around midnight. When time normally

advances linearly, this kind of definition means that at midnight the time value goes from its maximum value to minimum value. For example, meaning that 23:59 and 00:01 are almost the maximum 86 400 seconds apart from each other when actually the difference is only 120 seconds. To correct this we can consider each training sample to be at two locations at the same time. So, for example, the 23:59 that is same the as 86 340 seconds would also be found at -60 seconds, meaning that a query that asks closest points to 00:01 (or 60 seconds) would correctly see it as being 120 seconds away. The same needs to be done to the sample at 00:01 by moving it to 86 460 seconds. This way the queries done close to midnight can get the correct values, the downside being that the data is effectively doubled in memory. The result of this exercise can also be found from the same result table with ”modified time” identifier.

Clearly modifying the time as described above does not help. There are other ways to modify time as well, like using two coordinates instead of one and mapping the time values to circle. Or I could drop the assumption that days are similar (as is done with ANN), but then I would lose stationarity and the training data would be only used for finding the hyper-parameters. The other option is to assume that adding a extra dimension to the prediction made it harder for the models to generalize correctly, and that might very well be the case since both k-NN and laGP are using Euclidean distance which is very problematic in high dimensional spaces. Looking back at the exploratory analysis, the daily difference in the RSRP average was only 4 dBm it might well be that adding the extra dimension gave more problems for the model than it was able to gain from the extra knowledge.

4.1 Visualization

The figure 4.1 shows how the results from different prediction methods look when plotted as a map. Each of the maps in the figure is plotted over the same area using the same scale for colours, although some of the values have been ”squeezed”, meaning that the extreme values have been clipped to a more narrow range, in order to keep the details visible. But no amount of ”squeezing” is going to help the models on the lower row. FRK and both of the neural networks are just providing predictions that are too smooth, additionally, the neural network has some edge effect where one subsection turns into the next one. In these plots, and also in the results, the amount of the edge effect is reduced by only using the centre 16×16 area of the full 32×32 prediction. The edge effect could potentially be eliminated completely by using even smaller area from the centre, only using one pixel from each 32×32 prediction at extreme, but that would significantly increase the computing requirements for the inference. K-NN and laGP in the other hand have no problems providing predictions with very sharp changes, although it is hard to say from these

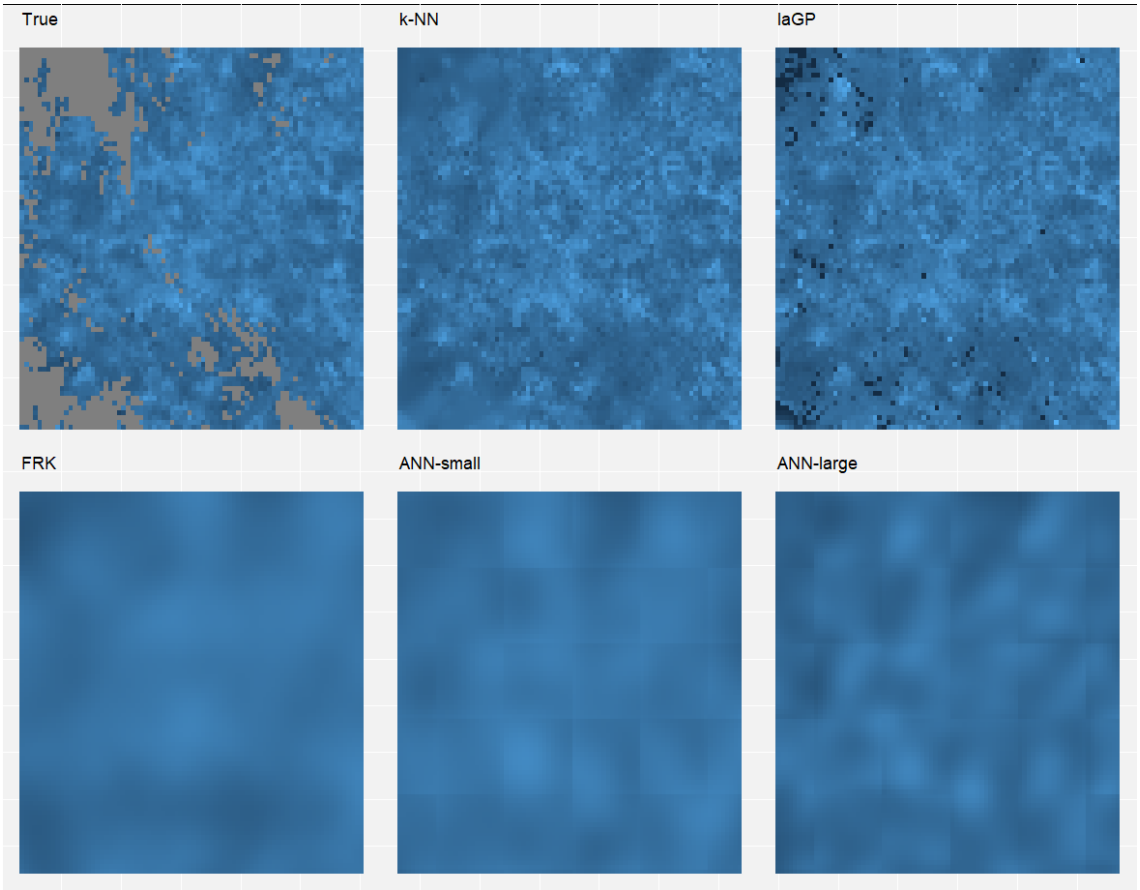

Figure 4.1 *Visualizations of the uniform grids in the form of maps, generated by all the predictive models and the "true" values aggregated from the test set. Instead of showing the whole city, the maps show an approximately* $10km \times 10km$ *area close to the city centre. The grey areas in the "true" values are un-observed since this is close to the city centre and consist of all the observations in the test set there is not many missing areas. All the maps are done for the frequency band A.*

images alone how well the predictions follow the "true" values from the test set.

While even the best predictions and the ground truth lack details on the uniform grid, on the adaptive grid very distinct details can be seen. The adaptive grid visualizations are given in figure 4.2, where all but FRK are able to have enough detail to show even individual roads. It is also easy to see how the adaptive grid is partitioned into different sized squares, the squares along the roads are visibly smaller than the ones in the top left corner for example. The squares "missing" on the edges are ones that are partially outside the window. Again it is hard to say how good the predictions actually are, but it is clear that FRK produces too smooth results.

However, finding roads was not the aim of the thesis. The true power of the predictions in the context of the thesis can be seen when predictions done on more than one measurement are combined. This can be easily done when the same grid

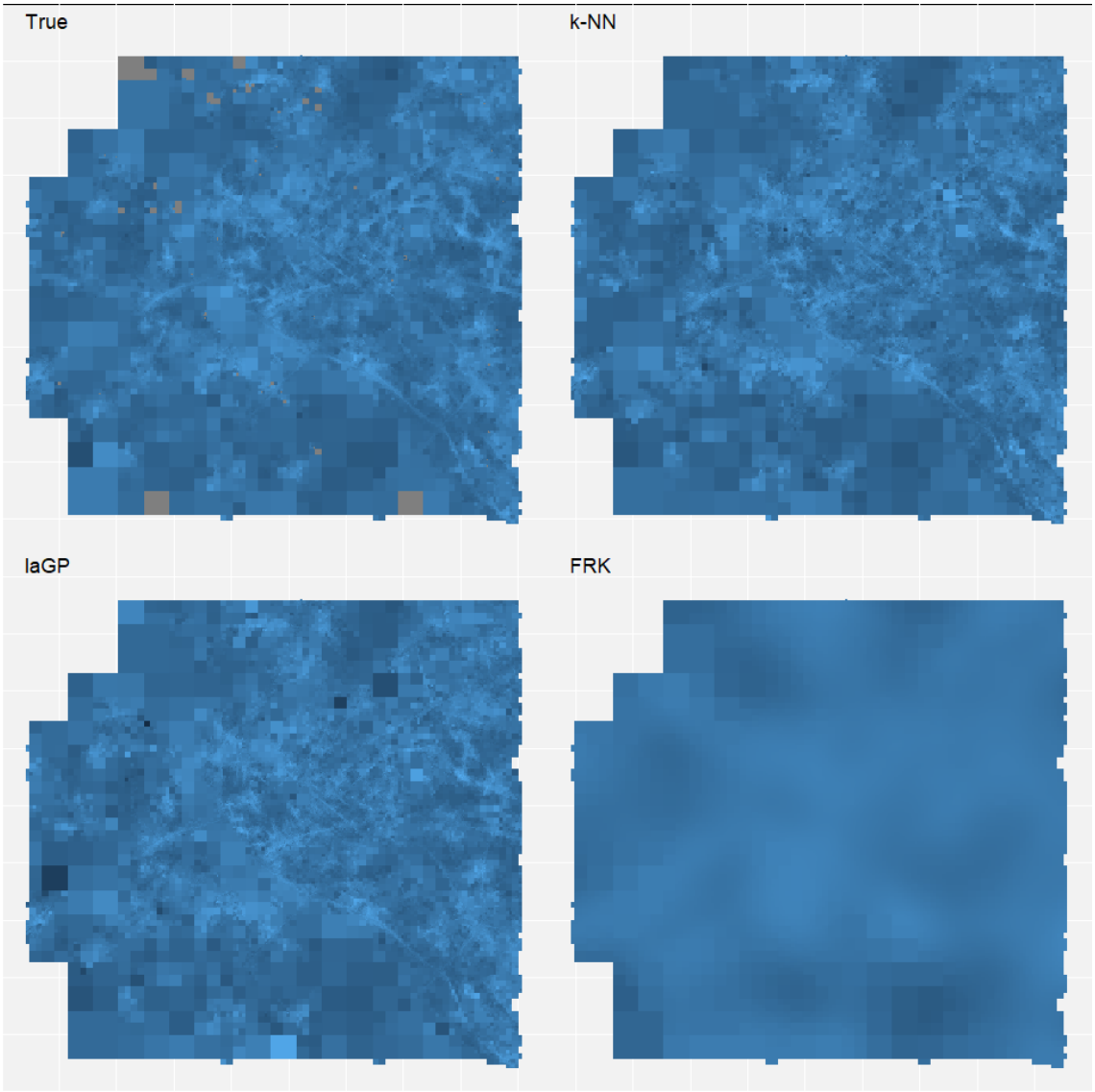

Figure 4.2 *Predictions on an adaptive grid shown as a map from all 3 predictive models as well as the "true" values from the test set. The area is the same* $10km \times 10km$ *area that was used in the figure 4.1. The missing squares on the edge mean that the square starts outside the selected area.*

is used for predicting both of the values. The visualizations from such combination can be seen in figure 4.3. For these images the parameters were tuned again to get more accurate results. Combining RSRQ or SINR with RSRP can be used to detect areas that might have lower data transmission rates than could be expected from the RSRP alone. The values on the plots on the bottom row have been transformed so that three different "classes" are created, red for "bad" areas where both the RSRP and the other measurement have bad values, green for average areas where either one has good values (or both have average values) and blue for areas where both measurements have good values. These areas no longer have higher values along roads unlike the predicted values of RSRP alone did, however, the areas are

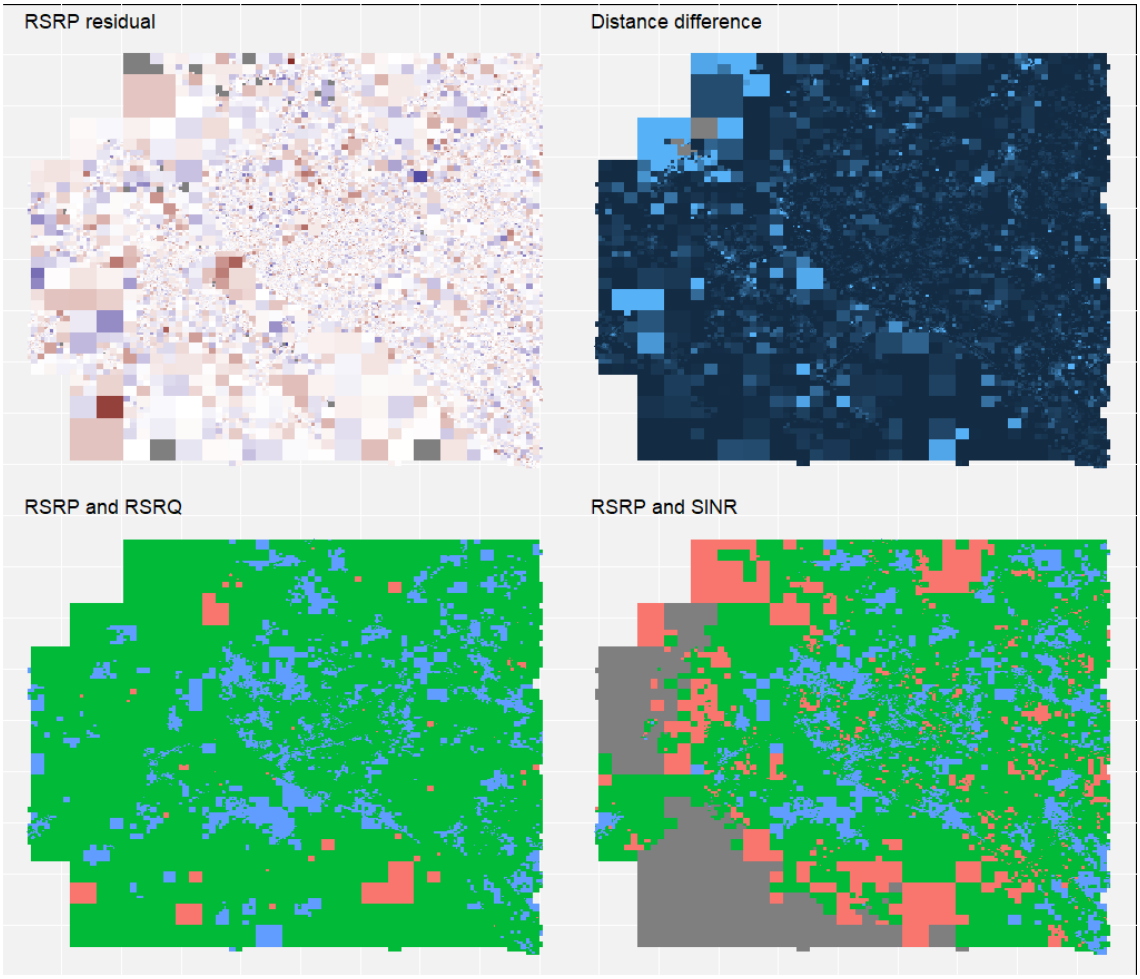

Figure 4.3 *Top left plot shows the spatial distribution of residuals from predicting the RSRP, red meaning predicting too high values and blue too low. While there are large differences between true and predicted values, there does not seem to be a clear spatial correlation. Top right shows the absolute difference between distances given by GPS and timing advance based distance calculations, the colours are squished to be between 30 and 300 meters to allow enough contrast in values where it matters. Bottom left is a combination of RSRP and RSRQ, red means that both are showing a bad connection, green means average and blue is for good. Bottom right is the same as the bottom left, but this time between RSRP and SINR. Since SINR is not observed in the whole network there are some extra missing values as well.*

clustered together as could be expected. The grey areas in the RSRP vs. SINR plot are due to SINR not being observed in the whole network. The data is not filtered based on if the SINR is observed or not and on the grey areas, all the samples used for k-NN are ones that do not have SINR. It would be possible to provide a prediction on this area as well, but it is unlikely to be very accurate because the samples used to make that prediction would be far away.

The top right plot could be used to determine areas with strong multipath fading effect. It looks like the difference in the distances given by timing advance and GPS are strongest on the area in the middle that has fewer observations and on the top

left. The area in the middle is, in fact, a river and the top left is the sea. So it can be concluded that the results are as expected since water is known to reflect the signals from wireless networks, causing the signals to take multiple different paths and cause multipath fading.

On the top left plot, the difference between the k-NN prediction and the true values can be seen. There does not seem to be an obvious spatial correlation between the residuals so it can be said that the model can capture most of the information from the spatial context that there is available.

Turning to spatio-temporal predictions it is clear that the predictions look much smoother on areas with a lower number of samples, as can be seen from image 4.4. Now that there is time-domain there are two different ways to look at the data, one can look at it as one pool as was done in the spatial only case or the data can be filtered by time. The image has two different ”true” images, the first one has data at 12:00-13:00 from every day in the test data and the second one has data between 12:00-13:00 from one day only.

It is clear that if we only had images from observed data without any interpolation it would be hard to compare data from different hours as the same areas may not be observed at all. Equally clear is that the ANN with more nodes was unable to learn anything useful. Comparing the k-NN and laGP results to true values are harder, but based on the MAE from the table 4.3 they are supposed to be a little bit better than their spatial only alternatives.

The same can be said from the predictions done on adaptive grids as well, which can be seen in image 4.5. Although from the adaptive plots it is clear that laGP predictions are much smoother all around than the spatial counterparts.

In image 4.6 we have clarifying example on how the ANN results are formed. As an input, we have eight 32×32 matrices of aggregated values. The light blue areas are zeroes, meaning not observed. Also, another set of eight 32×32 matrices (not shown here) of zeroes and ones are given to show whether a particular value is observed or not. Then on the bottom row, the second last is the true value that is expected and as the last image, we have the output from the neural network.

Finally in image 4.7 a potential use case for temporal predictions is shown. By taking the difference between two maps it is easy to see what has changed. In this case, the time difference between the maps used for differencing was one hour and the measurement was RSRP, but the same technique could be applied to any time difference or measurement or even combination of measurements as was done in image 4.3. When the grid is filled by the interpolation it does not matter if the exact same grid square is observed on different timesteps as the missing areas are interpolated, without interpolation the resulting image would be very sparse.

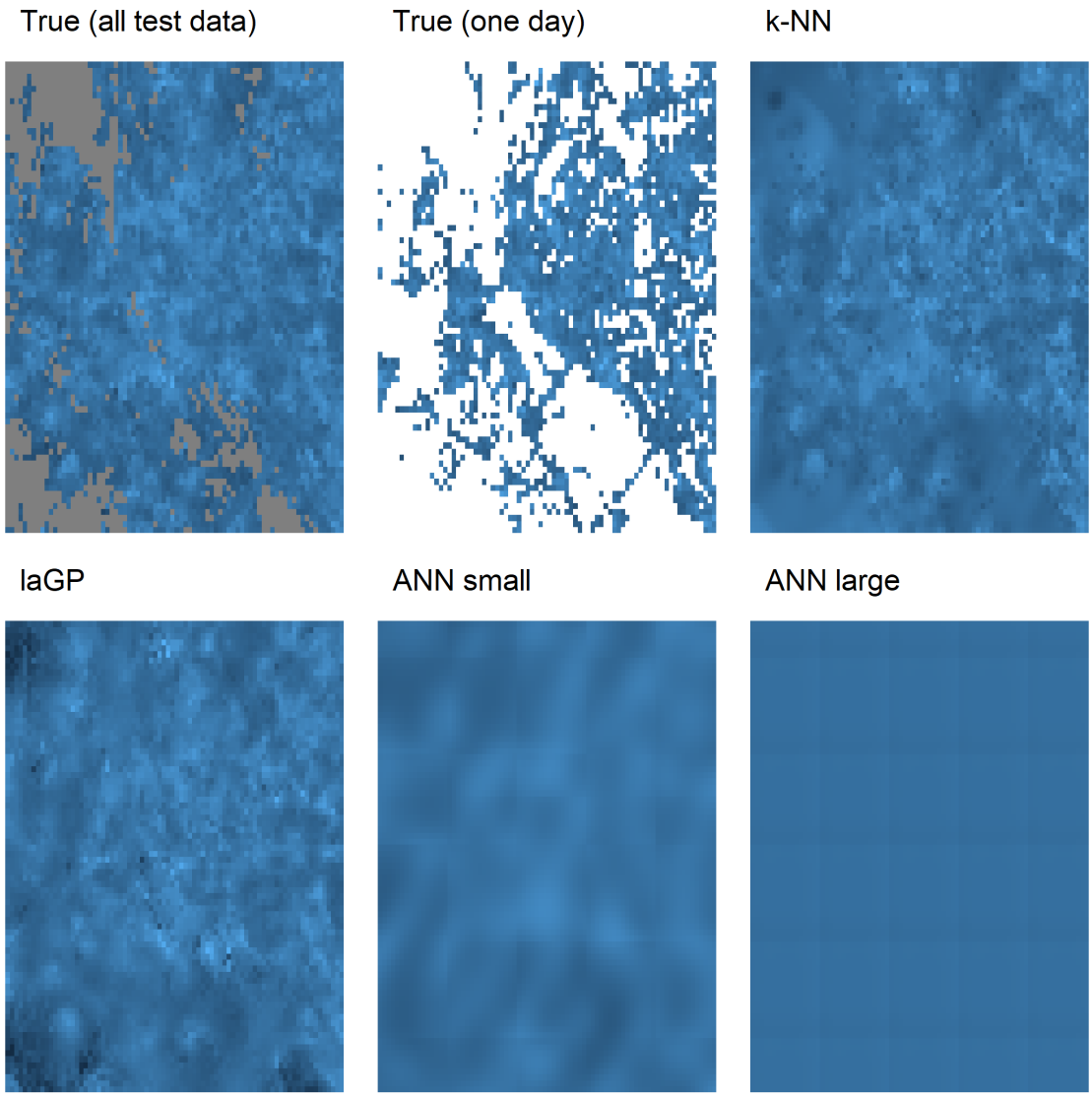

Figure 4.4 *Comparison of the spatio-temporal predictions on a uniform grid. Two true values, one by combining the same one hour window from each day and the other one by taking the same one hour window on one day. And four predictions from the models.*

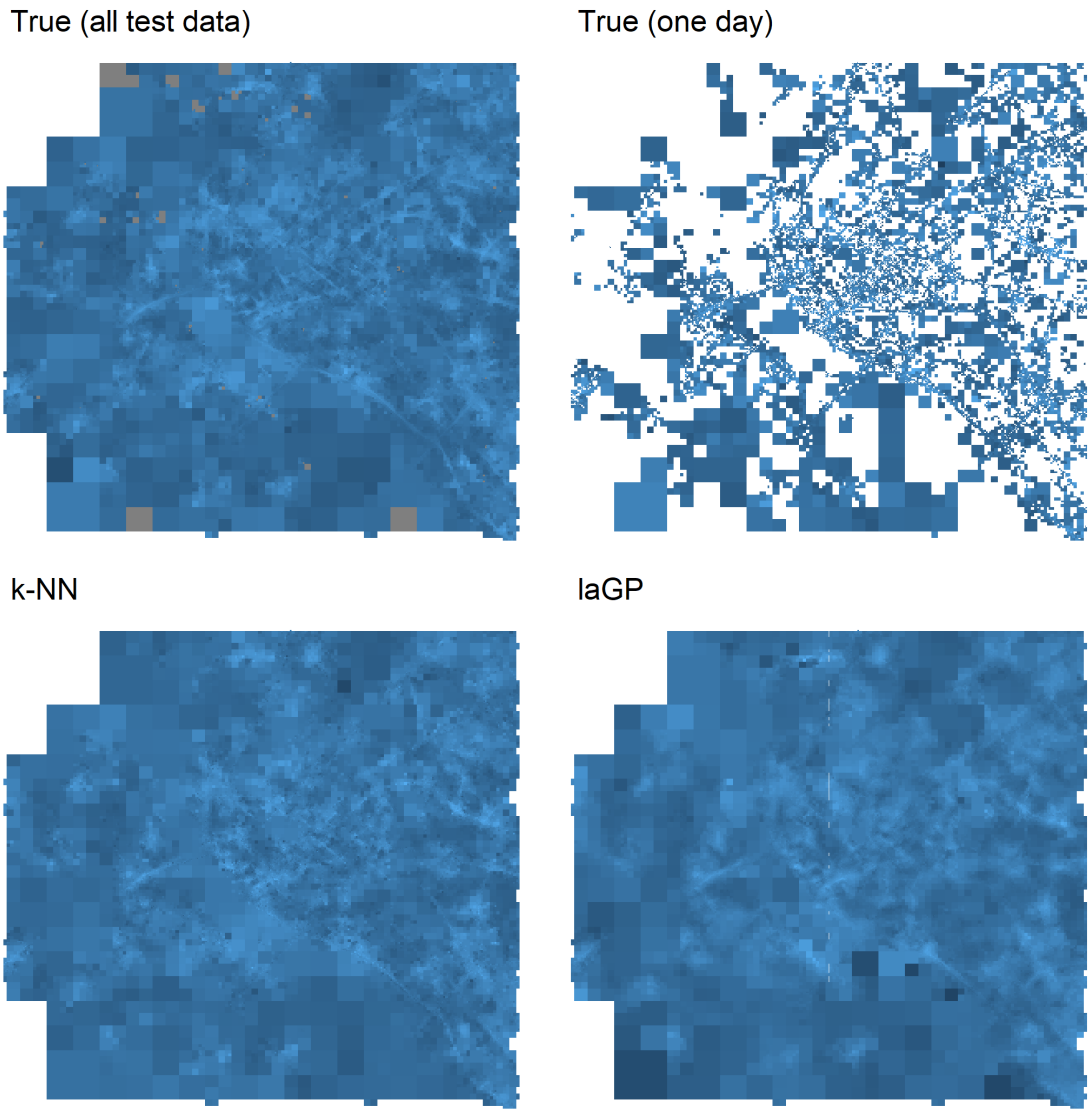

Figure 4.5 *Comparison of the spatio-temporal predictions on a adaptive grid. Two true values, one by combining the same one hour window from each day and the other one by taking the same one hour window on one day. And two predictions from the models.*

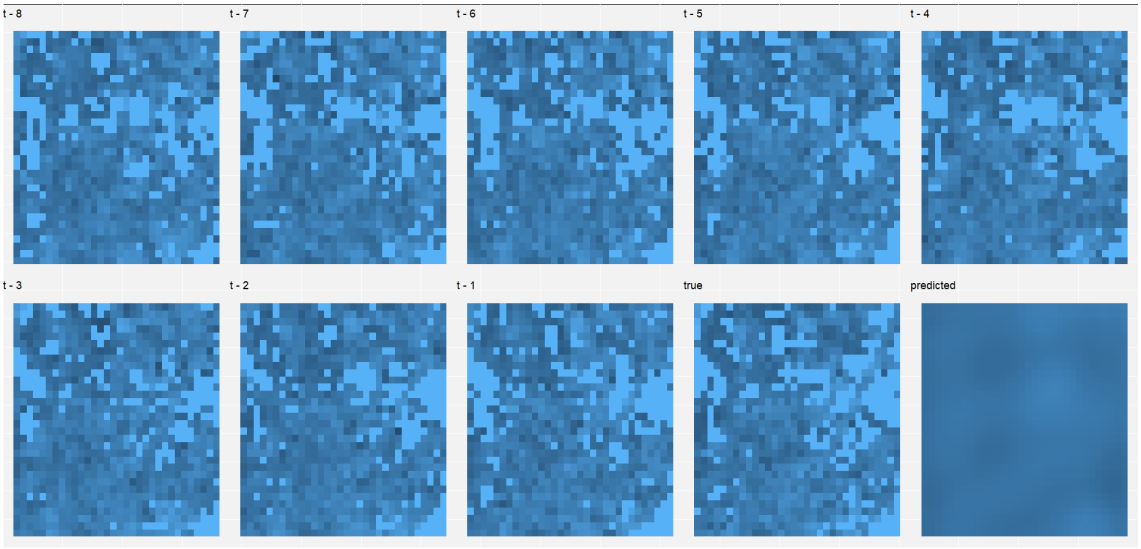

Figure 4.6 *Example of how the inputs and outputs of the spatio-temporal ANN model. First eight inputs, then the expected outcome and finally the actual result.*

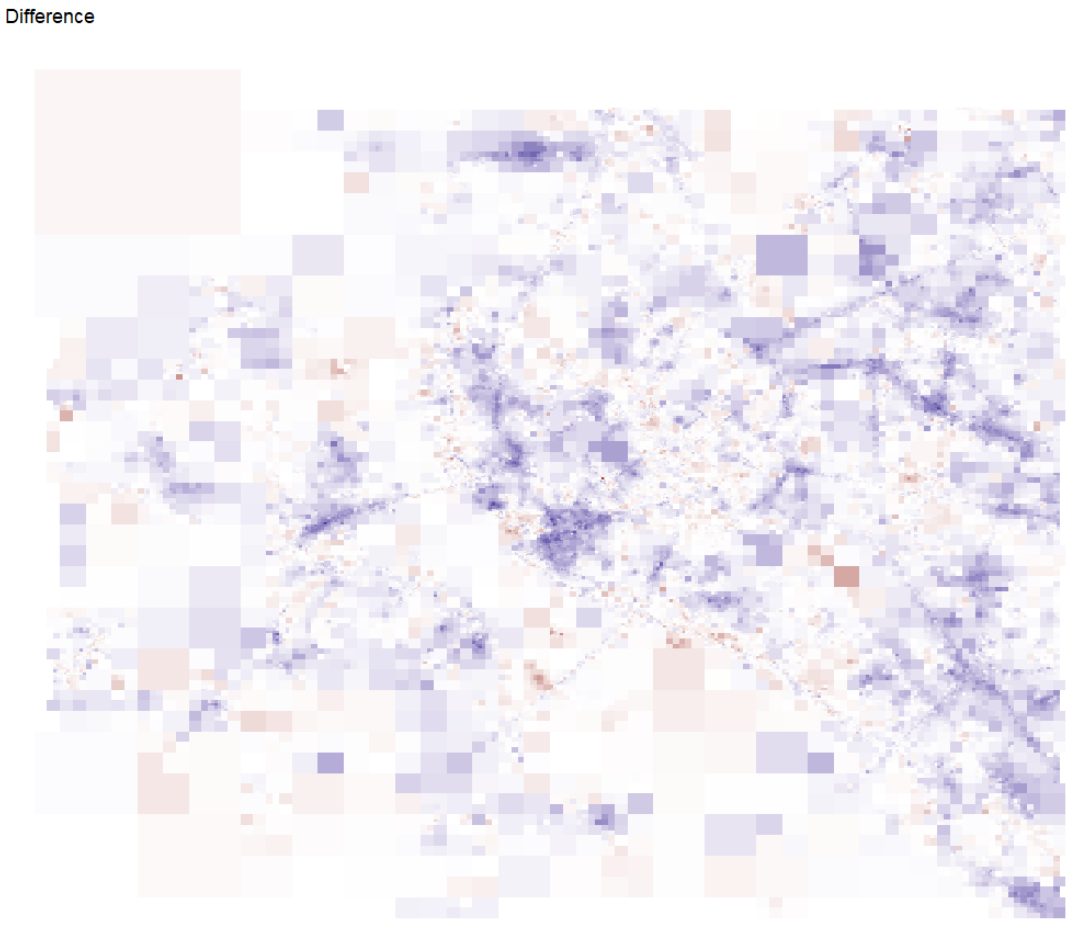

Figure 4.7 *The image shows the difference between k-NN prediction on adaptive grid generated on subsequent timesteps. The blue squares indicate that the RSRP has become worse and red squares indicate that RSRP has become better.*

5 Conclusion

Four models were tried for spatial prediction of LTE mobile networks RSRP values and three of them were also tested in spatio-temporal setting. The aim of testing them was three-fold: create accurate interpolations from the observations, forecast the future values and to create visualizations that could aid in detecting network degradation. The models were chosen based on literature review; k-nearest neighbours interpolation based on speed, fixed rank kriging because it is somewhat established method for spatial interpolation in large data sets, locally approximate Gaussian process regression for its promising performance in other large data sets and novelty in this context and finally artificial neural network because of its great success in recent years and prominence in literature.

While searching for accurate predictions to be shown as visual maps it became clear that uniformly selecting the locations to predict is not optimal for accuracy-computing time trade-off. Because of that, a way of selecting the prediction locations non-uniformly based on the data distribution was created. Adaptive selecting of the prediction locations is a novel approach in this context based on the literacy reviewed for the thesis. In the tests, the adaptively chosen prediction locations had better accuracy with less computation required than the uniform alternative.

From the results it can be seen that k-NN, which was chosen for its speed and not accuracy, actually did the best job predicting the values, closely followed by laGP. Both of these models focus on a small area instead of using a larger context, which seems to be beneficial in this case. FRK which tries to create "global" functions to describe the values and ANN that used approximately $4km \times 4km$ area at once did not perform so well. Also based on the test results it seems that those models that did well in interpolation in the spatial case did not really benefit from adding the time information and commonly performed worse. Neither did ANN really benefit when changing from interpolation to forecasting. While the interpolations were accurate, the forecasting was not good enough to be used for proactive network optimization.

Visualization wise the most notable thing is the adaptive grid. Commonly the visualizations are based on a uniform grid that has lower resolution to offset the computational burden or the data is mapped as points that do not allow efficient comparison across time. The adaptive grid enables high resolution on densely observed areas and easy comparison across time as the grid is not changing. With accurate interpolations it does not matter if some of the grid squares do not have observations, the comparison can be done anyway. Comparing maps from different times makes it efficient to see sudden changes in network quality.

I feel like that the adaptive grids could have been taken further. The algorithm as presented in the thesis uses parameters that depend heavily on the data, those could be made more general to allow transferring them from one case to another. Also, the loss function used to optimize the parameters could probably be better, now it simply avoids the most obvious problems. Also, it would be interesting to test changing the grid over time, in the thesis this was dropped due to problems it would cause with the visualizations and extra complexity it would bring overall.

Also, the ANN could have been taken a lot further. There has been a lot of research done around ANNs and time-series forecasting with ANNs has taken leaps beyond what is shown in the thesis. Then again a whole thesis could have been done around testing and optimizing different kind of networks to forecast the data. The rather simple approach was preferred over more complex ones because there was a paper that showed that this network architecture had been promising in a similar context and due to computational limitations. I think that even the spatio-temporal forecasting case could be possible with more up-to-date network architecture and enough training resources. One interesting research topic that also arises is generating predictions on an irregular grid using ANNs. In the thesis, it was just said it is not possible, but I think at least some predictions could be done with graph convolutions. And seeing how effective the irregular grid was over the uniform one I think it would be worthwhile to study how to irregular grids could be used more effectively. The same thing could be said about using point data as an input to ANN, now the point data was aggregated on the uniform grid but that loses many features that the point data has that cannot be found from the aggregated matrix.

References

[1] Farhana Afroz et al. "SINR, RSRP, RSSI and RSRQ Measurements in Long Term Evolution Networks". In: *International Journal of Wireless & Mobile Networks* 7.4 (2015), pp. 113–123. ISSN: 09754679. DOI: 10.5121/ijwmn.2015.7409.

[2] Emmanouil Alimpertis et al. "City-wide signal strength maps: Prediction with random forests". In: *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019* (2019), pp. 2536–2542. DOI: 10.1145/3308558.3313726.

[3] Muhammad Zeeshan Asghar et al. "Assessment of deep learning methodology for self-organizing 5G networks". In: *Applied Sciences (Switzerland)* 9.15 (2019). ISSN: 20763417. DOI: 10.3390/app9152975.

[4] Muhammad Zeeshan Asghar et al. "Towards proactive context-aware self-healing for 5G networks". In: *Computer Networks* 128 (2017), pp. 5–13. ISSN: 13891286. DOI: 10.1016/j.comnet.2017.04.053.

[5] Jon Louis Bentley. "Multidimensional Binary Search Trees Used for Associative Searching". In: *Commun. ACM* 18.9 (Sept. 1975), pp. 509–517. ISSN: 0001-0782. DOI: 10.1145/361002.361007. URL: https://doi.org/10.1145/361002.361007.

[6] M. J. Berger and P. Colella. "Local adaptive mesh refinement for shock hydrodynamics". In: *Journal of Computational Physics* 82.1 (1989), pp. 64–84. ISSN: 10902716. DOI: 10.1016/0021-9991(89)90035-1.

[7] Alina Beygelzimer, Sham Kakade, and John Langford. "Cover trees for nearest neighbor". In: *ACM International Conference Proceeding Series* 148 (2006), pp. 97–104. DOI: 10.1145/1143844.1143857.

[8] Hajer Braham et al. "Fixed rank kriging for cellular coverage analysis". In: *IEEE Transactions on Vehicular Technology* 66.5 (2017), pp. 4212–4222. ISSN: 00189545. DOI: 10.1109/TVT.2016.2599842. arXiv: 1505.07062.

[9] Noel Cressie and Gardar Johannesson. "Fixed rank kriging for very large spatial data sets". In: *Journal of the Royal Statistical Society. Series B: Statistical Methodology* 70.1 (2008), pp. 209–226. ISSN: 13697412. DOI: 10.1111/j.1467-9868.2007.00633.x.

[10] D. Denkovski et al. "Reliability of a radio environment map: Case of spatial interpolation techniques". In: *Proceedings of the 2012 7th International ICST Conference on Cognitive Radio Oriented Wireless Networks and Communications, CROWNCOM 2012* (2012), pp. 248–253. DOI: 10.4108/icst.crowncom.2012.248452.

[11] Etsi. *Evolved Universal Terrestrial Radio Access (E-UTRA) ETSI TS 136 214 V9.1.0 (2010-04)*. 2010.

[12] *Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Self-configuring and self-optimizing network (SON) use cases and solutions*. 36.902. 3GPP. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2581.

[13] Tim Farnham. "Radio environment map techniques and performance in the presence of errors". In: *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC* (2016). DOI: 10.1109/PIMRC.2016.7794911.

[14] Mah Rukh Fida and Mahesh K. Marina. "Impact of Device Diversity on Crowdsourced Mobile Coverage Maps". In: *14th International Conference on Network and Service Management, CNSM 2018 and Workshops, 1st International Workshop on High-Precision Networks Operations and Control, HiPNet 2018 and 1st Workshop on Segment Routing and Service Function Chaining, SR+SFC 2* Cnsm (2018), pp. 348–352.

[15] Steven Fortune. "A sweepline algorithm for Voronoi diagrams". In: *Proceedings of the 2nd Annual Symposium on Computational Geometry, SCG 1986* (1986), pp. 313–322. DOI: 10.1145/10515.10549.

[16] Ana Galindo-Serrano et al. "Harvesting MDT Data: Radio Environment Maps for Coverage Analysis in Cellular Networks". In: (2013), pp. 38–40. DOI: 10.4108/icst.crowncom.2013.252055.

[17] Robert B. Gramacy. "LaGP: Large-scale spatial modeling via local approximate Gaussian processes in R". In: *Journal of Statistical Software* 72.1 (2016). ISSN: 15487660. DOI: 10.18637/jss.v072.i01.

[18] Gregory Jefferis, Sunil Arya, and David Mount. *RANN2: Fast Nearest Neighbour Search ('Rcpp' Wrapper of 'Arya' and Mount's 'libANN')*. R package version 0.1.0.9001. 2020. URL: https://github.com/jefferis/RANN2.

[19] Xin Kang et al. "Sensing-Based Spectrum Sharing in Cognitive Radio Networks". In: *October* 58.8 (2009), pp. 4649–4654.

[20] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2323. ISSN: 00189219. DOI: 10.1109/5.726791.

[21] Jingming Li et al. "Recent Advances in Radio Environment Map: A Survey". In: *Machine Learning and Intelligent Communications*. Ed. by Xuemai Gu, Gongliang Liu, and Bo Li. Cham: Springer International Publishing, 2018, pp. 247–257. ISBN: 978-3-319-73564-1.

[22] Chengyong Liu et al. "Cost-effective signal map crowdsourcing with auto-encoder based active matrix completion". In: *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS* 2019-December (2019), pp. 761–768. ISSN: 15219097. DOI: 10.1109/ICPADS47876.2019.00112.

[23] Haitao Liu et al. "When Gaussian Process Meets Big Data: A Review of Scalable GPs". In: *IEEE Transactions on Neural Networks and Learning Systems* (2018), pp. 1–19. ISSN: 2162-237X. DOI: 10.1109/tnnls.2019.2957109. arXiv: 1807.01065.

[24] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[25] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259.

[26] Joseph Mitola and Gerald Q. Maguire. "Cognitive radio: Making software radios more personal". In: *Software Radio Technologies: Selected Readings* (1999), pp. 413–418. DOI: 10.1109/9780470546444.ch4.

[27] Marko Modsching, Ronny Kramer, and Klaus ten Hagen. "Field trial on GPS Accuracy in a medium size city: the influence of built-up". In: *3rd Workshop on Positioning, Navigation and Communication 2006* 2006 (2006), pp. 1–10. URL: http://modsching.com/papers/FieldtrialonGPSAccuracy10pages2006-02-10%7B%5C_%7D06crcit.pdf%7B%5C%%7D5Cnpapers2://publication/uuid/1C0C31BA-478C-4EB3-B096-FBFC04EDED53.

[28] Maureen N. Mureithi, Peter K. Kihato, and Agnes Mindila. "On the use of machine learning for temporal performance prediction in Lte advanced networks". In: *International Journal of Scientific and Technology Research* 8.10 (2019), pp. 1388–1393. ISSN: 22778616.

[29] Jyrki T J Penttinen. *The LTE-Advanced Deployment Handbook : The Planning Guidelines for the Fourth Generation Networks*. New York, UNITED KINGDOM: John Wiley & Sons, Incorporated, 2016. ISBN: 9781118678855.

[30] Marko Pesko et al. "Radio environment maps: The survey of construction methods". In: *KSII Transactions on Internet and Information Systems* 8.11 (2014), pp. 3789–3809. ISSN: 22881468. DOI: 10.3837/tiis.2014.11.008.

[31] Python Software Foundation. *Python Language Reference*. 2019. URL: https://www.python.org/.

[32] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2019. URL: https://www.R-project.org/.

[33] Md Shaifur Rahman et al. "Creating Spatio-temporal Spectrum Maps from Sparse Crowdsensed Data". In: *IEEE Wireless Communications and Networking Conference, WCNC* 2019-April.i (2019), pp. 1–7. ISSN: 15253511. DOI: 10.1109/WCNC.2019.8885811.

[34] C E Rasmussen et al. *Gaussian Processes for Machine Learning*. 2006. ISBN: 026218253X.

[35] Jacqueline Ryan, George R. MacCartney Jr., and Theodore S. Rappaport. "Indoor Office Wideband Penetration Loss Measurements at 73 GHz". In: (2017). eprint: arXiv:1703.08030.

[36] Andrea Scaloni. *Minimization of Drive Test (MDT) An Innovative Methodology for Measuring Customer Performance on Mobile Network*. 2019. URL: https://www.itu.int/en/ITU-T/Workshops-and-Seminars/20190311/Documents/Andrea_Scaloni_Presentation.pdf.

[37] Andrea Scaloni et al. "Multipath and Doppler Characterization of an Electromagnetic Environment by Massive MDT Measurements From 3G and 4G Mobile Terminals". In: *IEEE Access* 7 (2019), pp. 13024–13034. ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2892864.

[38] Amar Shah, Andrew Gordon Wilson, and Zoubin Ghahramani. "Student-t processes as alternatives to Gaussian processes". In: *Journal of Machine Learning Research* 33 (2014), pp. 877–885. ISSN: 15337928. arXiv: 1402.4306.

[39] SHEPARD D. "Two- dimensional interpolation function for irregularly- spaced data". In: *Proc 23rd Nat Conf* (1968), pp. 517–524. DOI: 10.1145/800186.810616.

[40] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems 25*. Ed. by F Pereira et al. Curran Associates, Inc., 2012, pp. 2951–2959. URL: http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf.

[41] Foad Sohrabi and Edgar Kuehn. "Construction of the RSRP map using sparse MDT measurements by regression clustering". In: *IEEE International Conference on Communications* (2017), pp. 1–6. ISSN: 15503607. DOI: 10.1109/ICC.2017.7997073.

[42] Marek Suchanski et al. "Radio Environment Maps for Military Cognitive Networks: Deployment of Sensors vs. Map Quality". In: *2019 International Conference on Military Communications and Information Systems, ICMCIS 2019* (2019), pp. 1–6. DOI: 10.1109/ICMCIS.2019.8842720.

[43] Yves Teganya and Daniel Romero. "Data-Driven Spectrum Cartography via Deep Completion Autoencoders". In: (2019). arXiv: 1911.12810. URL: http://arxiv.org/abs/1911.12810.

[44] *Universal Terrestrial Radio Access (UTRA) and Evolved Universal Terrestrial Radio Access (E-UTRA); Radio measurement collection for Minimization of Drive Tests (MDT); Overall description; Stage 2*. 37.320. 3GPP. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2602.

[45] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN: 978-3-319-24277-4. URL: https://ggplot2.tidyverse.org.

[46] Gabriel Porto Villardi et al. "Primary Contour Prediction Based on Detailed Topographic Data and Its Impact on TV White Space Availability". In: *IEEE Transactions on Antennas and Propagation* 64.8 (2016), pp. 3619–3631. ISSN: 0018926X. DOI: 10.1109/TAP.2016.2580164.

[47] Andrew Zammit-Mangion and Noel Cressie. "FRK: An R Package for Spatial and Spatio-Temporal Prediction with Large Datasets". In: (2017). arXiv: 1705.08105. URL: http://arxiv.org/abs/1705.08105.

[48] Xincheng Zhang. *LTE Optimization Engineering Handbook*. Newark, SINGAPORE: John Wiley & Sons, Incorporated, 2018. ISBN: 9781119158998.