國立臺灣大學電機資訊學院電信工程學研究所
碩士論文
Graduate Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

Using Machine Learning to Predict 3 Telecommunication Parameters in the Case of Metro Passengers Itineraries

Alexandre Benayoun

指導教授：魏宏宇 博士
Advisor: Hung-Yu Wei, Ph.D.

中華民國 111 年 9 月
September, 2022

doi: 10.6342/NTU2022XXXXX

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

Using Machine Learning to Predict 3 Telecommunication Parameters in the Case of Metro Passengers Itineraries

本論文係君 (R10942164) 在國立臺灣大學電信工程學研究所完成之碩士學位論文，於民國 111 年 9 月 12 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Graduate Institute of Communication Engineering, National Taiwan University on 12 September 2022 have examined a Master's Thesis entitled above presented by Alexandre Benayoun (R10942164) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

_____ _____ _____

（指導教授 Advisor）

所　長 Director: _____

doi: 10.6342/NTU2022XXXXX

誌謝

中文致謝格式，舉凡學生撰寫論文後的感想，及在論文完成的過程中，獲得指導教授及其它老師有實質幫助之研討及啟發，或行政、技術人員、同學及親友等幫忙者，皆可在此項次誌謝，內容力求簡單扼要，以不超過一頁為原則。

doi: 10.6342/NTU2022XXXXX

摘要

中文摘要內文，不超過三頁，其內容應包含論述重點、方法或程序、結果與討論及結論。經系、所同意以英文撰寫論文者，仍需附中文摘要。

關鍵字： 中文、關鍵、詞

doi: 10.6342/NTU2022XXXXX

Abstract

This study focuses on the use of machine learning techniques to predict three important telecommunication parameters in the context of metro passengers itineraries. These parameters are the base station changes, the latency of the signal and the number of packet loss. They represent abnormal phenomenon, or events that can alter the phone's performance. Being able to succesfully predict it can lead to a better anticipation of these issues and enhance the user's experience.

The primary objective of this research is to compare different machine learning models for real-time predictions of the studied parameters. To do so, different algorithms (Neural Networks, Recurrent Neural Networks, LSTM and ARIMA), as well as several sets of features will be used. A comparison on the error metrics will also be conducted. The study took novel approaches in the nature of the studied parameters and the prediction delay, as it aims to forecast the value a few seconds into the future. Additionally, it proposes new solutions to make predictions with a dataset mainly composed of zeros.

Overall, this study contributes to the understanding of machine learning applications in predicting telecommunication parameters in the case of metro passengers itineraries. The findings suggest that the selected machine learning algorithms, combined with appropriate error metrics and innovative approaches, offer reliable solutions for real-time predictions for the three studied

doi: 10.6342/NTU2022XXXXX

parameters. Besides, the proposed solutions to avoid models that always predict zero with datasets mainly composed of null values proved to be successful. The predictions from the chosen models will help in the decision-making of the settings of the phone, to avoid abnormal phenomenon and maintain a good performance throughout the user's route.

Keywords: Machine Learning, latency, handover, packet loss, error metric, time series, Neural Networks

doi: 10.6342/NTU2022XXXXX

Contents

doi: 10.6342/NTU2022XXXXX

doi: 10.6342/NTU2022XXXXX

 doi: 10.6342/NTU2022XXXXX

List of Figures

doi: 10.6342/NTU2022XXXXX

doi: 10.6342/NTU2022XXXXX

List of Tables

doi: 10.6342/NTU2022XXXXX

 doi: 10.6342/NTU2022XXXXX

Chapter 1. Introduction

1.1 Motivation

In communication, certain parameters, such as handover, high latency, and packet loss, can significantly impact the performance. Thus, being able to predict them in real time during a metro itinerary could greatly enhance the performance by helping to choose the safest setting and prevent these abnormal phenomena.

Many different machine learning models are valuable tools to predict parameters. Consequently, it is important to study which ones perform better on this specific application. A survey of different algorithms, parameters and features is necessary to have a global vision of the problem and make the right conclusions.

Moreover, error metrics are very important in the decision making throughout the process and the conclusion. The error metric is used to measure an error between the predictions and the actual values. The result from it translates the accuracy of the prediction and is the principal indicator of the model ' s performance. It is used during the cross validation phase, leading to the decision of the parameters values. It is also crucial in the testing part, influencing the final conclusion on the model performance. Therefore, knowing which error metric is more suitable for this application is a key point.

Additionally, I faced a unique issue that has not been addressed in prior research. Specifically, more than 99% of the datasets I used exhibit a number of packet losses equal to

doi: 10.6342/NTU2022XXXXX

zero. Consequently, the prediction of this parameter is challenging. The issue of datasets predominantly composed of zeros is crucial in the field of machine learning and many researchers have thought of possible solutions, but not for such a high proportion. Finding new solutions to increase the performance of the prediction with this new kind of dataset is a noteworthy challenge.

To sum up, this work is focusing on the comparison of several Machine Learning models for the prediction of three telecommunication parameters. These parameters are collected from the user's side. Indeed, the data is taken from metro passengers itineraries, following their movement. On the contrary, the server's side is when the data is collected from the base station, it can include multiple phones and does not follow one specific user path, it is simply taking all, or a part of, the data from the phones connected to one or several base stations. The predictions from the models are meant to be analyzed in real-time to choose the safest phone settings to enhance the performance.

1.2 Related Works

There are studies about the prediction of the Packet Loss Rate using Machine Learning techniques. The authors of [1] developed a Machine Learning tool based on a Random Forest Regression model to predict, evaluated with real world data. [2] also studied the use of Decision Tree, and added a Logistic Regression algorithm.

[3] and [4] propose real time packet loss rate prediction at t+1. [3] proposes a novel prediction model with an adaptative nonlinear approach. [4] focuses on data mining models development, and uses a Multilayer Perceptron. [5] is a statistical analysis of data traffic measurements, specifically on packet loss. The authors of [6] have conducted a study of packet loss behavior, analyzing big network traffic data and using a classification approach with the Machine Learning classifier XGBoost.

All of these works are about the prediction of the packet loss rate. I chose to take a different approach by predicting the number of packet losses. Moreover, these studies were conducted from the network operator ' s side. My work will take the user side by follow-

doi: 10.6342/NTU2022XXXXX

ing a metro passenger itinerary.

As regards the prediction of the latency, there are studies using classification models. For example, [7] contributed by studying real operational network and real data on the user end. Their model is predicting 3 different classes: low latency, medium latency and high latency, using Logistic Regression, Support Vector Machine and Decision Tree classification algorithms. [8] also studied a classification model in the aim of optimizing NoC architecture components, with a Random Forest algorithm. Other studies took a regression approach. [9] studied the validation of Support Vector Machine algorithm and performed a feature selection analysis, but is not performing real time forecasting, unlike [10] and [11]. The authors of [10] are presenting a novel approach for latency estimation with an online learning algorithm in a mobile scenario. [11] identified the main challenges of the problem and validated the benefits of the Recurrent Neural Network algorithm with LSTM cells for the prediction of the latency.

I found that the studies using a regression algorithm were more appropriate to achieve my goals, even if the others gave me useful informations on the features, error metrics and algorithms commonly used for that kind of data. Similarly to [10] and [11], my study focuses on real time forecasting. As said before, my work is in the case of a user ' s movement. [10] did the same using an online learning algorithm. It would be interesting to see how well other algorithms do in this application, as [10] only tried one model. I chose to use the algorithm from [11] (LSTM) in my specific application. Even if the contexts are different with the data taken from the user's side in my case, the model will be working on the same kind of data and the study showed that it performed well on their realistic settings. Moreover, this algorithm is a well known machine learning time series model that can be applied to many different problems. Thus, my work will extend this study by exploring an additional application.

Finally, a few studies are using Machine Learning to deal with the handover predic-

doi: 10.6342/NTU2022XXXXX

tion. [12] is studying the case of actual vehivle movement, the authors aim to predict the handover and learn the variation of performance over the course of the day. Their model is using a Neural Networks classification algorithm to predict the fog node used at a given location in real time. [13] is also predicting in real time which base station is used in a mobility scenario. The authors want to exhibit that simple methods can outperform more sophisticated models by studying a probabilistic approach to determine the probability of the handover to each neighboring base station. [14] has a different approach. It is forecasting the future number of handover attempts to meet their goal to manage handover for a huge number of cells.

As I am interested in real time forecasting, the studies about the prediction of the next cell are more pertinent than the one predicting the number of handover attempts, even if the latest ([14]) is giving interesting informations on the studied parameters. [12] and [13] are proposing solutions to forecast the base station the phone will be connected to at the time t+1. With this prediction, we know if there will be a handover at t+1. However, if there is no handover, these methods do not give any indication on when the next change of base station will happen. Therefore, my work studied the prediction of the time left before the next change of base station.

The previous studies about real time forecasting are all performing predictions at t+1, i.e. they are predicting the very next value of the parameter under investigation. However, this solution could be complicated to apply in real life scenarios, as the delay for the prediction is short. Indeed, the data is at least collected every second, and often has higher collection frequency. Therefore, a model producing predictions at t+1 only have a delay of a second. In my work, I will consider forecasting the parameters a few seconds in the future.

Some studies are proposing solutions for datasets mainly composed of zeros. For example, [15] is taking care of this issue by using a binomial regression. However, it is not the case with 99% of zeros. Indeed, the examples taken in the studies are at most

doi: 10.6342/NTU2022XXXXX

around 60 or 70% of zeros. Thus, my research will propose new solutions for this issue.

1.3 Contribution

This work compares different Machine Learning models for real-time predictions of base station changes, latency, and packet loss for a user's itinerary in the metro.
The studied models are different combinations of a machine learning algorithm, a set of features and an error metric. 21 distinct combinations are studied for the base station change prediction, 13 others are used to forecast the latency, while 12 models are compared in the case of the packet loss inverstigation.

To the best of my knowledge, this study is the first to forecast these parameters a few seconds into the future.
The advantage of this approach is that it enables to have more time to analyze the predictions and change the phone settings if necessary. Moreover, it gives a better view of how the parameter ' s value is likely to evolve in the near future.

To the best of my knowledge, this study is the first to design and compare error metrics specifically for this application.

This study employs novel approaches for predicting the number of packet losses and the time before the next change of base station. To the best of my knowledge, it is the first study to investigate these predictions.
Predicting the number of packet loss is relevant in a context of very few packet loss, as the packet loss rate is even lower. The idea to forecast the time before the next change enables to know how far the next change of base station is. Focusing on the prediction of the cell to which the phone will be connected to at the next time step is successfully giving the information on whether a base station change happens at the next time step. Nevertheless, it does not give any indication on the next change if the phone is predicted to be connected to the same base station. Therefore, this new method is very valuable as it gives much

doi: 10.6342/NTU2022XXXXX

more information.

This study proposes innovative solutions for prediction using datasets primarily consisting of zeros. To the best of my knowledge, it is the first study attempting to do so with datasets composed of more than 99% zeros.

This study is made in collaboration with other students working on finding the best phone settings to avoid abnormal phenomena. The predictions of the machine learning models are meant to help them in the selection of the settings. One of their goals is to reduce the packet loss during metro itineraries. It makes sense that the datasets contains very few packet loss, as the issue with this parameter has already been treated. However, the number of packet loss still needs to be predicted to see if a change of setting to try to reduce the latency will have an impact on packet loss for example. Having a global vision on the three parameters is necessary for final the decision making. Finally, this was a good opportunity to work on solutions for using machine learning techniques in such context. The proposed solutions could be useful in other applications.

doi: 10.6342/NTU2022XXXXX

Chapter 2. Methodology

2.1 System Architecture

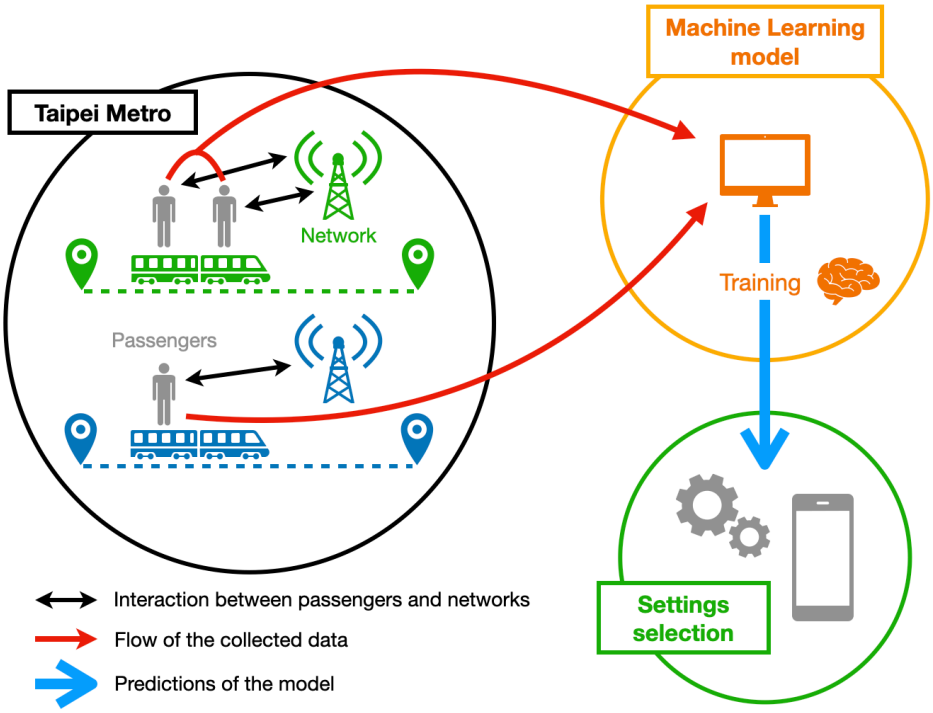


Figure 2.1: System Architecture

Figure 2.1 shows the proposed system architecture of this study. The system is composed of three parts, each with its own characteristics: the Taipei Metro, the Machine Learning model and the settings selection.

Several passengers from different itineraries in the Taipei metro are experiencing interactions between their phone devices and the telecommunication network. A lot of data can be collected and gathered from this connection. Then, the collected data from the passengers is transmitted to the Machine Learning model, which is the second component of the

doi: 10.6342/NTU2022XXXXX

system.

The received data undergoes analysis and processing in such ways that will be detailed later. This work contribute to the development of a machine learning model. After training the model, the proposed algorithm is applied to the tested data and delivers predictions for the future values of the parameters under investigation.

These predictions are examined and interpreted, leading to the decision-making process for the phone settings selection. The aim of the system is to help choose the safest settings to prevent abnormal phenomena by forecasting these parameters with the Machine Learning model built with the data from the Taipei metro.

Table 2.1 is listing all the variables from the system and their description.

Variable	Description
N	Number of samples in a dataset
N_1	Number of samples in a training set, $N_1 = 0.8 * N$
N_2	Number of samples in a testing set, $N_2 = 0.2 * N$
M	Number of features in a dataset
T	t_{step} (for Recurrent Neural Networks models)
T_2	t_{pred} (for Recurrent Neural Networks models)
k	Number of units (NN) or size of the vector h (RNN)
H	Number of hidden layers
S_i	Number of collections of the feature latency during the i^{th} second
L_i	Number of collections of the feature loss during the i^{th} second
tp_1	Starting collection time in a pcap file
tp_2	Ending collection time in a pcap file
tc_1	Starting collection time in a csv file
tc_2	Ending collection time in a csv file

Table 2.1: Table of the variables of the problem

doi: 10.6342/NTU2022XXXXX

2.2 Processing Workflow

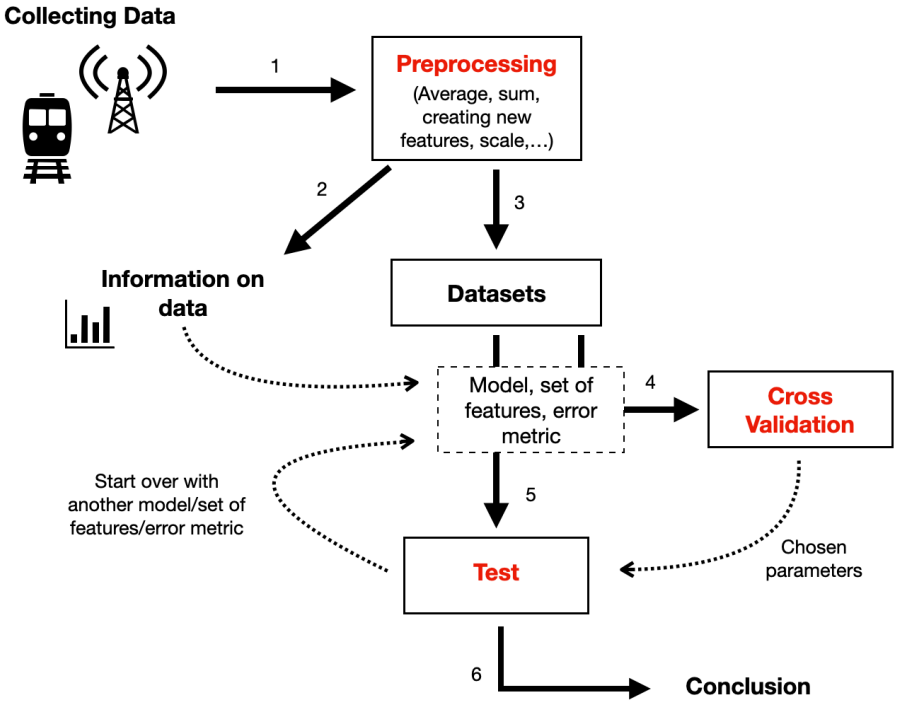


Figure 2.2: Processing Workflow

The processing workflow from Figure 2.2 shows the different steps in the study ' s work, that will be detailed and explained in the following sections.

2.2.1 Data Collection

The data was collected in the Taipei metro in two different ways with different collecting time and frequencies: one for the latency and packet loss, gathered in $pcap$ files, and another for the other data, put together in csv files.

Figure 2.3 shows the architecture of these files. In this figure, x_1 and x_2 are representing two features from the csv file, lat is the latency value and $loss$ is a packet loss.

As explained before, the collection frequency is not the same for the two files. Indeed, we see that at each time, i.e. each second, x_1 and x_2 are collected once whereas lat and $loss$ have respectively S_i and L_i collections (for the i^{th} second). Moreover, the files do not have the same beginning and ending collection times. The data from the csv files is collected from tc_1 to tc_2 and the latency and packet loss from the $pcap$ files are collected from tp_1

doi: 10.6342/NTU2022XXXXX

to tp_2.

Finally, we see that the loss feature has some missing values. In the example of Figure 2.3, $loss$ does not have any value at the times tp_1 and tp_2. That is because this feature exists only if a packet loss is detected and is equal to 1. However, there is no packet loss in 99% of the used datasets.

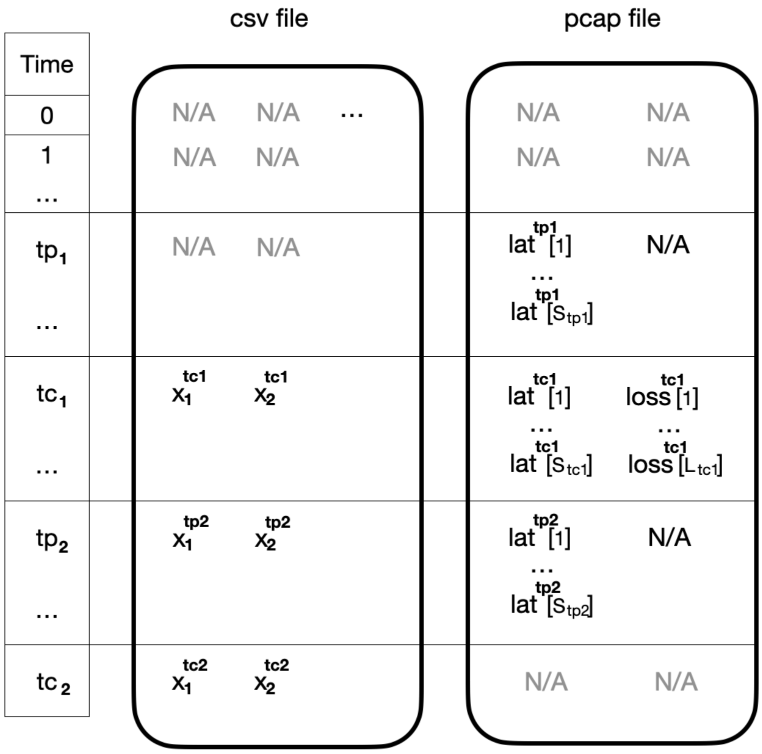


Figure 2.3: Architecture of the csv and pcap files

The issues raised about the difference of collection frequencies, beginning and ending times, and missing values will be adressed in the next phase of the study.

2.2.2 Preprocessing

After collecting the data, the study is going through the preprocessing part with several actions to do to gather information on the data and build the datasets.

In order to have the same collection frequency for every feature, I decided to take the average of the latency for every second to match the collection frequency of the data from *csv* file.

doi: 10.6342/NTU2022XXXXX

Let's call x the new feature created from this operation.

$$\text{For } i \in [tp_1, tp_2], \text{ we have } x[i] = \frac{1}{S_i} \sum_{k=1}^{S_i} lat^i[k]$$

```
# returns the average of the latency at each second
function transf_latency ( latency )   # latency is a list such as [ [t1, t2, ...], [latency at t1, latency at t2, ...] ]
la1 ← latency [0]            # la1 = [t1, t2, ...] (list of the times)
la2 ←latency [1]             # la2 = [latency at t1, latency at t2, ...] (list of the latency values)
nlatency ← length of la2
avg_latency ← [ ]
time ← 0
s ← 0                # s will be the sum of the latency values in a second
c ← 0                # c will be the count of the values in a second
for i ← 0 to nlatency
        time2 ← time of the i-th element in second
        valuelatency ← value of the latency of the i-th element
        if time != time2   # if we are not in the same second than before
                add a new element to avg_latency: [time, s/c]     # s/c is the average of the latency
                s ← valuelatency            # initialize s and c for the new second
                c ← 1
        else                # if we are in the same second than before
                s ← s + valuelatency        # update s and c
                c ← c + 1
        time ← time2
return avg_latency          # we return a list of [ [t1, average latency at time t1], [t2, ...], ...]
```

Each $loss$ in the $pcap$ file is equal to 1 and corresponds to a packet loss. Therefore, to match the collection frequency of a second, I chose to sum the number of packet losses that happen during a second with the function $transf\_loss$.

Let's call x the new feature created by this operation.

$$\text{For } i \in [tp_1, tp_2] \text{ and } L_i > 0, \text{ we have } x[i] = \sum_{k=1}^{L_i} loss^i[k] = L_i$$

As explained before, there is initially no value for the packet loss feature of the $pcap$ file when there is no packet loss at a given second. Therefore, a zero should be added each time it happens.

$$\text{For } i \in [tp_1, tp_2] \text{ and } L_i = 0, \text{ we have } x[i] = 0$$

doi: 10.6342/NTU2022XXXXX

```
# counts the number of packet loss for every second when there is at least 1 packet loss
function transf_loss ( loss )          # loss is a list of the times when there is a packet loss
nloss ← length of loss
count_loss ← [ ]  # an empty list
time ← 0
for i ← 0 to nloss
        time2 ← time of the i-th packet loss in second
        if time != time2               # if we are not in the same second than before
                add a new element to count_loss: [time2, 1]
        else                           # if we are in the same second than before
                update the last element of count_loss: [time2, k] updated to [time2, k+1]
        # we have another packet loss at the same second so we update the number of packet loss for this
        time
        time ← time2                   # update to the current time
return count_loss           # we return a list of [ [t1, number of packet loss at time t1], [t2, …], …]
```

After dealing with the collection frequency disparity, the differences in the starting and ending collection times should be taken care of. The features values from the times $max(tc_1, tp_1)$ to $min(tc_2, tp_2)$, i.e. when the data from both files is collected, are kept.

```
# takes the value of the features at the times when the data from the csv file AND from pcap files is collected
function transf ( csv, pcap, … ) # csv and pcap are examples of features respectively from csv and pcap file
csv2 ← [ ]
pcap2 ← [ ]
t_csv ← time of the 1st element of csv
value_csv ← value of the 1st element of csv
t_pcap ← time of the 1st element of pcap
value_pcap ← value of the 1st element of pcap
while there is still at least one element left in each feature's list
        if t_csv == t_pcap                  # if the collection time is the same
                add value_csv to csv2               # add the values to the new lists
                add value_pcap to pcap2
                t_csv ← time of the next element of csv
                value_csv ← value of the next element of csv
                t_pcap ← time of the next element of pcap
                value_pcap ← value of the next element of pcap
        else
                if t_csv < t_pcap           # if the time from the csv file is lower
                        t_csv ← time of the next element of csv          # take the next time and the
                        value_csv ← value of the next element of csv             next value of the csv
feature
                else                        # if the time from the csv file is lower
                        t_pcap ← time of the next element of pcap        # take the next time and the
                        value_pcap ← value of the next element of pcap   next value of the pcap feature
return csv2, pcap2
```

New features were created from the data. One of them is called "change". It indicates if there is a change of base station. This feature is equal to 1 if there is a change of base station and 0 otherwise.

doi: 10.6342/NTU2022XXXXX

Let ' s call base the feature from the csv file indicating the base station the phone is connected to.

For $i \in]tc_1, tc_2]$, we have

$$change[i] = 0 \text{ if } base[i-1] = base[i]$$

$$change[i] = 1 \text{ otherwise.}$$

Another created feature called "time" is giving the time in seconds before the next change of base station. This new feature will not have the same ending time of collection tc_2. Indeed, after the last change of base station collected in the dataset, we do not know when the next one will be. Therefore, the time before the next change of base station is unknown. The ending time for this feature will be the time of the last change of base station. If this feature is taken in the dataset, all the other features should also be taken at the right collection times, when all the features are collected, considering this new ending time.

```
# builds the feature "time before the next change of base station" and rebuilds the other features
function transf_pci ( PCI, data1, ... )              # data1 is an example of a feature returned by get_datacsv
time_PCI ← [ ]              PCI is a feature =1 if there is a change of base station and =0 otherwise
d1 ← [ ]
n ← length of data1
time ← 0
c ← 0
k ← 0
while c == 0
        c ← (n-k)th element of PCI                    # we start from the end of the PCI list
        k ← k + 1                                     the while loop end at the last change of base station of the list
for i ← k to n
        c ← (n-i)th element of PCI
        t ← time of the (n-i)th element of PCI
        if c == 1
                time ← t                              # we set the time to the time t of the change
                time2 ← 0   # time2 is the gap between the current time t and the time of the last change
        else      we are going from the end of the list so it corresponds to the time before the next change
                time2 ← time - t
        add a new element to time_PCI: [t, time2]
        add a new element to d1: (n-i)th element of data1
n2 ← length of d1
time_PCI2 ← [ ]
d1bis ← [ ]
for i ← 0 to n2             # we go backward a second time to restore the chronological order
        add a new element to time_PCI2: (n2-i)th element of time_PCI
        add a new element to d1bis: (n2-i)th element of d1
return time_PCI2, d1bis
```

"loss0/1" is a feature indicating if there is at least one packet loss. It is equal to 1 if there is at least one packet loss and 0 otherwise.

doi: 10.6342/NTU2022XXXXX

For $i \in [tp_1, tp_2]$, we have

$$loss0/1[i] = 0 \text{ if } L_i = 0, \text{ and } loss0/1[i] = 1 \text{ otherwise.}$$

```
loss2 ← [ ]                    # loss2 will be the list indicating if there is at least 1 packet loss
n ← length of loss             # loss is a list that contains the number of packet losses at each
time
for i ← 0 to n
        if loss [ i ] = 0
                add 0 to loss2
        else
                add 1 to loss2
```

As mentioned before, the number of packet loss is 0 for 99% of the datasets. Therefore, predicting it using machine learning techniques is very difficult as models will tend to always predict 0. ”enlarged loss” is a feature enlarging the peaks of packet loss. The idea behind this feature is to “ enlarge the peaks ” to create curves in the graph of the packet loss upon time and therefore reduce the number of null values. The real number of packet losses and this new feature from dataset 2 are plotted on Figure 2.4 to compare them. As observed in this graph, the newly created feature has much more non-null values, while still keeping the overall shape of the true values evolution. Even if the information about the exact number of packet loss at each time is lost, this approach could be beneficial for the selection of a pertinent model for the packet loss prediction in such context.

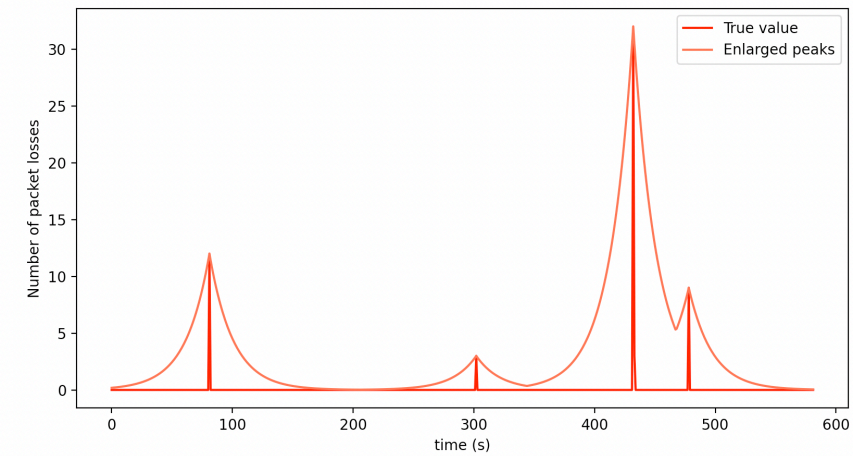


Figure 2.4: Graphs of the packet loss en enlarged loss for the testing set of dataset 2

This feature is created in a recursive form, such as if a value v at a rank i is not null, the values of the ranks $i+1$ and $i-1$ are set to $0.95 * v$ if they are lower than this new value.

doi: 10.6342/NTU2022XXXXX

```
# transforms the loss feature to have "enlarged peaks"
function transf_nbloss ( loss )        # loss is a list that contains the number of packet losses at each time
loss2 ← copy of loss
n ← length of loss
for i ← 0 to n
        v ← i-th element of loss
        if v > 0                       # if we have a non-zero value
                k ← 1
                while v > 0.01
                        v2 ← 0.95 * v
                        if loss2 [ i - k ] < v2          # if the previous value is not already greater than v2
                                loss2 [ i - k ] ← v2             # the previous value is updated to v2
                        if loss2 [ i + k ] < v2          # if the next value is not already greater than v2
                                loss2 [ i + k ] ← v2                     # the next value is also updated
                        v ← v2
                        k ← k + 1                        # in the 2nd iteration, if v is still > 0.01, the elements that
return loss2                                             # are 2 ranks higher and lower will be updated, and so on…
```

Two other features are created from the initial data. At each time, the strength of the 4G signal is evaluated with two features: LTE RSRP and LTE RSRQ. Similarly, NR SSRSRP and NR SSRSRQ are defining the strength of the 5G signal. These four features are collected at each step, regardless of the actual type of connection, i.e. if it is under 4G or 5G connection. The study investigate 4G base station changes, so the data corresponding to this type of connection is taken in this case. However, the latency of the signal and the number of packet loss are dependant on whether the connection is 4G or 5G, and the correct features should be used depending on the scenario. The features ”RSRPbis” and ”RSRQbis” are created with this idea and are calculated such as:

$$RSRPbis[i] = LTE\_RSRP[i] \text{ in the case of 4G connection}$$
$$RSRPbis[i] = NR\_SSRSRP[i] \text{ in the case of 5G connection}$$

$$RSRQbis[i] = LTE\_RSRQ[i] \text{ in the case of 4G connection}$$
$$RSRQbis[i] = NR\_SSRSRQ[i] \text{ in the case of 5G connection}$$

All the features are specified in Table 2.2. A number is associated with each feature, and they will be referred to by their corresponding number in the following parts of the study. The table also provides a description for every feature.

doi: 10.6342/NTU2022XXXXX

N°	Name	Description
1	LTE RSRP	Average value of the strength of the reference 4G signal transmitted by the base station
2	LTE RSRQ	Ratio between RSRP and RSSI (which represents the total power of the 4G signal)
3	NR SSRSRP	Same than n°1 but for 5G signal
4	NR SSRSRQ	Same than n°2 but for 5G signal
5	SigStrength	Received signal strength from the nearest base station
6	SigStrength1	Received signal strength from the second nearest base station
7	Speed	Speed of the phone (in m/s)
8	RxRate	Total number of bytes received per second
9	TxRate	Total number of bytes transmitted per second
10	Latency	Latency of the signal (in ms)
11	Loss	Number of packet loss during a second
12	RSRPbis	= n°1 if the phone is in 4G connection and = n°3 if it is in 5G connection
13	RSRQbis	= n°2 if the phone is in 4G connection and = n°4 if it is in 5G connection
14	Change	= 1 if there is a change of base station and = 0 otherwise
15	Time	Time (in s) before the next change of base station
16	Loss0/1	= 1 if there is at least 1 packet loss and = 0 otherwise
17	Enlarged Loss	If Loss(t)>0, then Loss(t-1) = Loss(t+1) = 0.95*Loss(t)

Table 2.2: Table of the features

doi: 10.6342/NTU2022XXXXX

The features are gathered into a set that is taken as input by the algorithm. This work is using numerous sets of features in the different experiments. They are listed in Table 2.3, their composition is detailed with the corresponding numbers of the features included and a name is given to each set.

Set	Features used
S_1	[1, 2, 5, 6, 7, 8, 9, 10, 11]
S_2	[1, 2, 7, 8, 9, 10, 11]
S_3	[(12, 13, 5, 6, 7, 8, 9, 11) at t, (10) at t-1]
S_4	[(12, 13, 7, 8, 9, 11) at t, (10) at t-1]
S_5	[(12, 13, 7, 8, 9, 10) at t, (16) at t-1]
S_6	[(12, 13, 7, 8, 9, 10) at t, (11) at t-1]
S_7	[12, 13, 7, 10]
S_8	[1, 2, 7]
S_9	[12, 13, 7, 8, 9, 10, 11]
S_{10}	[10]

Table 2.3: Table of the sets of features

Another goal of the prepocessing part is to gather information on the data. It is important to analyze the data to choose which features should be used, spot the unusual tendencies and outliers, see if a pattern can be detected, and know which model can be used.

The first approach to gather information is by visualizing the data on graphs. By doing so, we can see the global evolution of the data, if there are outliers and how far and frequent they are, the range of the data, etc. Several graphs can also be superposed to compare the evolution of different parameters, their range, etc.

Then, a mathematical approach is necessary to put numbers on the observations made by visualization.

doi: 10.6342/NTU2022XXXXX

Let ' s call $x \in \mathbb{R}^N$ a feature of the dataset, we can calculate the mean of x:

$$mean(x) = \frac{1}{N} \sum_{k=1}^{N} x[k]$$

The maximum of x is $max(x) = x[i]$ with $i \in [1, N]$ such as $x[i] \geq x[k], \forall k \in [1, N]$.
The minimum of x is $min(x) = x[i]$ with $i \in [1, N]$ such as $x[i] \leq x[k]\ \forall k \in [1, N]$.
The standard deviation $std(x) = \sqrt{\frac{1}{N} \sum_{k=1}^{N} \mid x[k] - mean(x) \mid^2}$

Besides, we can calculate correlation coefficients between features to see if they are correlated. If two features are perfectly correlated, one of them should not be used. Indeed, they are bringing the same information to the model and are redundant. Therefore, one of them is useless and removing it will increase the speed of the model. Moreover, multicollinearity can lead to solutions that are wildly varying and can be numerically unstable. Finally, it is preferable to keep a simpler model, and have fewer features. It will make the interpretability of the model easier.
The correlation can be calculated with the Pearson correlation coefficient, that translates the linear relationship between 2 features x_1 and x_2:

$$Pearson(x_1, x_2) = \frac{cov(x_1, x_2)}{std(x_1) * std(x_2)}$$

with $std(x)$ defined earlier and $cov(x, y)$ such as:

$$cov(x, y) = \frac{1}{N} \sum_{k=1}^{N} (x[k] - mean(x)) * (y[k] - mean(y))$$

with $mean(x)$ defined earlier.

The Spearman correlation coefficient is also useful. This coefficient reflects nonlinear relationships between two features x_1 and x_2. It shows the strength of the relationship between these features and can also be used for linear relationships even if the power will be lower than with the Pearson correlation coefficient.

$$Spearman(x_1, x_2) = \frac{cov(rank(x_1), rank(x_2))}{std(rank(x_1)) * std(rank(x_2))}$$

where $rank(x)$ is alist of the ranks of the values in the ascending order.

doi: 10.6342/NTU2022XXXXX

For example, if $x = [2, 4, 0]$ is considered, the ascending order is $0, 2, 4$. Therefore, we would have $rank(x) = [2, 3, 1]$.

The two defined coefficients give a number between -1 and 1. 1 meaning a perfect positive correlation and -1 is the result for a perfect negative correlation. If the two variables are totally independent, the result is 0.

Feature	5	6	7	8	9	10	11	12	13
5		0.836	0.135	-0.004	0.052	-0.028	-0.268	0.389	0.036
6	0.836		0.111	-0.002	0.019	0.000	-0.232	0.479	0.014
7	0.135	0.111		-0.064	-0.036	0.060	0.060	0.149	0.136
8	-0.004	-0.002	-0.064		0.066	-0.021	-0.096	-0.046	-0.061
9	0.052	0.019	-0.036	0.066		-0.011	0.072	-0.019	0.021
10	-0.028	0.000	0.060	-0.021	-0.011		0.033	-0.035	-0.071
11	-0.268	-0.232	0.060	-0.096	0.072	0.033		-0.222	0.050
12	0.389	0.036	0.149	-0.046	-0.019	-0.035	-0.222		0.273
13	0.036	0.014	0.136	-0.061	0.021	-0.071	0.050	0.273	

Table 2.4: Spearman correlation coefficients table for dataset 1

As an example, Table 2.4 gathers the Spearman correlation coefficients between the features n°5 to 13 for dataset 1. In this table, the coefficients that are higher than 0.3 or lower than -0.3 are highlighted in red, while those whose absolute value is between 0.2 and 0.3 are highlighter in yellow. It can be seen that the features used have mostly relatively small correlation coefficients, which shows that they provide complementary informations. However, the features n°5 and 6 are resulting in some higher coefficients. Indeed, they respectively have a value of 0.389 and 0.479 with the feature n°12. Most notably, the Spearmann coefficient between them is by far the highest among the table, at 0.836. That means that these two features are very similar, may bring redundant information already carried by feature n°12, and therefore be detrimental to the model. This result is not very

doi: 10.6342/NTU2022XXXXX

surprising considering the nature of these features. Indeed, they represent the same physical parameter taken from two different base stations. Some of the sets of features that will be used in the study's experiments include these features while others do not. By doing so, the effect of their presence will be studied to confirm the conlusions drawn from this analysis.

Besides, it is during the preprocessing part that the dataset is built. The first step is to collect the data contained in the *csv* and *pcap* files. The way to collect the latency and the number of packet loss from the *pcap* files has already been detailed earlier. Concerning the *csv* files, the data is taken for each time step with its corresponding time to ensure the syncrhonizing part between the data, as well as the creation of the feature n°15 that needs the time of each base station change to be calculated.

```
# collects the data from the csv files (data that are not latency and packet loss)
function get_datacsv ( file )
n ← length of file
data1 ← [ ]
data2 ← [ ]                    # data1 and data2 are examples of features extracted from the csv file
for i ← 0 to n
        time ← time of the i-th element of file
        add a new element to data1: [time, i-th element of data1 in the csv file]
        add a new element to data2: [time, i-th element of data2 in the csv file]
return data1, data2
```

After extracting the data, synchronizing it, creating new features and gathering information helping on the feature selection, we can finally build the dataset with the N samples of the M chosen features.

The function X_y creates a dataset X of size $N * M$ and a vector Y of size N with the output values.

If the chosen model is a time series model, there is another part in the construction of the dataset. Indeed time series models are taking the features on several steps of time in the input, therefore the dataset should be rebuilt with this new shape.

doi: 10.6342/NTU2022XXXXX

```
# builds a dataset, for example with feat_csv and feat_pcap as features and label as the output value
function X_y ( feat_csv, feat_pcap, label )
n ← length of label
X ← table of shape ( n , number of features (2) )
Y ← [ ]
for i ← 0 to n
        X [ i , 0 ] ← i-th element of feat_csv
        X [ i , 1 ] ← i-th element of feat_pcap
        add the i-th element of label to Y
return X, Y
```

For time series dataset, we call t_{step} the step of time that is taken in the input. Therefore, each of the M features will be taken t_{step} times in the input. t_{pred} is the gap between the last time taken in the input and the prediction. We put $T = t_{step}$ and $T_2 = t_{pred}$.

The size of the previous dataset X is $N * M$, meaning that each of the M features has N values. The size of the new inputs is $M * T$. In each input, the features are taken T times. The first input takes the values from 1 to T, the second input takes the values from 2 to $T + 1$, and so on. Therefore the number of inputs is at most $N - T$.

Moreover, there is a gap of T_2 between the last element of the input and the output. The first output will be the value at $T + T_2$, the second output will be the value at $T + T_2 + 1$, and so on. When the last output, i.e. the value at the time N, will be taken in the Y vector, the corresponding input will take the values from $N - T - T_2$ to $N - T_2$. Therefore, the new dataset will contain exactly $(N - T - T_2)$ inputs. The size of the new dataset X will be $(N - T - T_2) * (M * T)$ and the size of the vector Y will be $(N - T - T_2)$.

```
# rebuilds the dataset to have a time series dataset
function rebuild_dataset ( X, y, t_step, t_pred )        # t_step is the step of time taken in the input
n ← length of y                                          t_pred is the gap of time before the prediction
X_rebuilt = [ ]
y_rebuilt = [ ]
for i ← 0 to ( n - t_step - t_pred )
        L ← [ ]
        for j ← 0 to t_step
                add the (i+j)th elements of X to L    # we add the elements from rank i to rank (i + t_step)
        add L to X_rebuilt
        add the ( i + t_step + t_pred )th element of y to y_rebuilt
return X_rebuilt, y_rebuilt
```

Finally, the last step is to divide the dataset between training and testing sets and scale it.

doi: 10.6342/NTU2022XXXXX

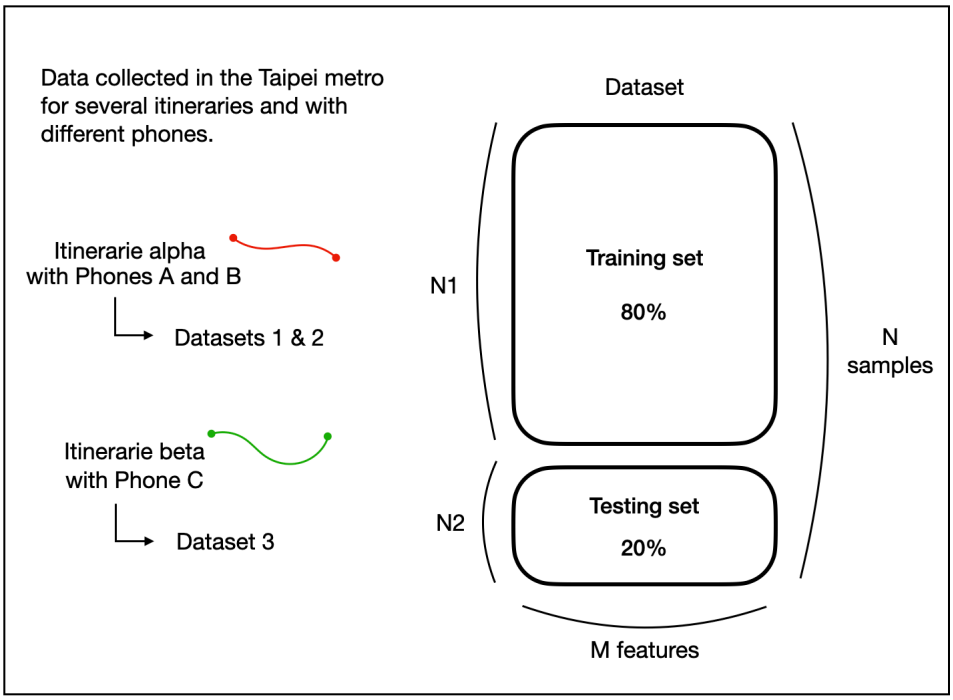


Figure 2.5: Architecture of the datasets

Figure 2.5 explains how the three datasets are made and divided between training and testing sets. The datasets used in the study contains data collected from three different phones and two different metro itineraries. The training set takes the values of the dataset from 1 to N_1 and the testing set takes the values from $N_1 + 1$ to N. The sizes of the training set X_{train} and the testing set X_{test} are respectively $N_1 * M$, with $N_1 = 0.8 * N$ and $N_2 * M$, with $N_2 = 0.2 * N$.

Then, the data can be scaled. Two different ways to scale the data are used in this study. Let ' s call x a feature and x_{scaled} the corresponding scaled feature.

The first scaling method is the Standard Scaling.

For $i \in [1, N]$, we have $x_{scaled}[i] = \frac{x[i] - m_{train}}{std_{train}}$

where m_{train} is the mean of the feature x in the training set: $m_{train} = \frac{1}{N_1} \sum_{k+1}^{N_1} x[k]$

and std_{train} is the standard deviation of the feature x in the training set:

$$std_{train} = \sqrt{\frac{1}{N_1} \sum_{k=1}^{N_1} | x[k] - m_{train} |^2}$$

Then, we have the MinMax Scaling.

doi: 10.6342/NTU2022XXXXX

For $i \in [1, N]$, we have $x_{scaled}[i] = \frac{x[i] - min_{train}}{max_{train} - min_{train}}$

where max_{train} and min_{train} are respectively the maximum and minimum of the feature x within the training set.

2.2.3 Cross Validation

After the datasets are built and the model, set of features and error metric chosen, the next step is the cross validation. In this part, the training set from 2 datasets out of 3 are taken to determine the best parameters to use.

For this phase, I chose to emlpoy a time series format. All the training set is usually used in every iteration of the cross validation, randomly divided into training and validation sets. For my experiments, I kept the newest data in the validation set and the data that came before are gathered in the training set.

The disadvantage of this method is that the full set is not used at each iteration and the unused parts could be beneficial in the training. However, the cross validation should match the testing context, i.e. training the model on the oldest data and testing it on the newest ones. In a real-time application, the testing will be performed on the latest data available. Therefore, maintaining the chronological order between training and validation in the cross-validation phase is essential for simulating real-world conditions and ensuring the reliability of parameter decisions.

The training set is divided into training and validation parts in 5 different ways. The set is divided into 10 parts. For the 1st cross validation, the training set (X_{CV}) is composed of the first five parts of the set and the validation (X_{val}) is the 6th part. For the 2nd cross validation, the training set consists of the 6 first parts of the set and the validation is the 7th part, and so on until the 5th cross validation.

Let's call $N_{CV} = \frac{N_1}{10}$

For $i \in [0, 4]$

doi: 10.6342/NTU2022XXXXX

$$X_{CV} = X_{train}[1 : (5+i) * N_{CV}] \text{ and } Y_{CV} = Y_{train}[1 : (5+i) * N_{CV}]$$

$$X_{val} = X_{train}[(5+i) * N_{CV} : (6+i) * N_{CV}] \text{ and } Y_{val} = Y_{train}[(5+i) * N_{CV} : (6+i) * N_{CV}]$$

At each iteration, the model is trained on the training set and tested on the validation set. The average error is returned by the cross validation function to evaluate the performance and assist the decision making concerning the parameters values.
This operation should be done for all the values of the parameters we want to test. Then, the results obtained with the error metric can be compared and the best values for each parameter can be chosen for the testing part.

2.2.4 Testing

For the test, the model is trained on the training set with the parameters values chosen with the cross validation, and we test it on the testing set for each dataset. The performance is measured with the error metric. The testing has to be repeated several times and the average result must be taken as the weights of the algorithm are randomly initialized and can influence the convergence of the model.

After the test, we can conclude on the model performance considering the error metric ' s result.

In my study, I wanted to study the influence of different parameters, such as the algorithm, the set of features and the error metric, on the model performance. Therefore, all these steps should be done again in the same conditions for each of the tested parameters. After determining which model is the most suitable, it will serve as a tool to give useful predictions on base station changes, the latency and packet loss, during a metro itinerary. This will lead to the choice of a setting to enhance the phone performance and improve the user's experience.

doi: 10.6342/NTU2022XXXXX

Chapter 3. Error metrics

Error metrics are essential in many fields. It is used to measure the error of a model. In most applications, it is an indicator on how close the actual values are from what was expected or predicted. Therefore, it is used to evaluate the performance of the model. In Machine Learning experiments, error metrics compare the actual values from the dataset to the predictions from the model.

It is used during the cross validation part to measure the performance of the trained model with various parameters. It helps in the decision making for the values of these parameters. Then, when the values of the different parameters are set, it is a tool to compare the predictions to the actual values from the test dataset in order to simply have a good view on how good the results are. Moreover, it can be used to compare the results with other models and do a benchmark. Hence, error metrics are present throughout the process and are very important in several steps. It has a big influence on the choice of the parameters values, the way we see and interpret the results and the comparison with other models. Therefore, it is crucial to use an error metric that makes sense with the aims of the study. In order to do that, I studied the structure of the metrics to understand how it works and which one I should use.

3.1 Structure of the metrics

A lot of different kinds of metrics are used in various field. In machine learning regression problems, the most commonly used are the Mean Absolute Error (MAE), the Mean Squared Error (MSE) and the Mean Absolute Percentage Error (MAPE).

doi: 10.6342/NTU2022XXXXX

Many papers have tried to classify the metrics according to different criteria. For example, [16] suggested classifying metrics into four groups, based on how the result is expressed. The four groups are the scale-dependent metrics, the percentage-error metrics, the relative-error metrics and the scale-free error metrics. This classification is simple and clear and enable to understand the basic properties of the metrics, e.g. if it is scale free or not. And choose one that matchs the needs of the application.

[17] proposed a structural approach to the metrics and proposed a new typology to define a metric. They identified three key components that determine the properties of metrics: the method of determining point distance, the method of normalization, and the method of aggregation of point distances over a data set.
According to this definition, the result of an error metric is obtained by determining the point distance between the actual and predicted values, then normalizing it and finally aggregating over a complete data set.

3.2 Determining point distance

In the following formulas, p_i is the predicted value and y_i is the actual value.
For each step, many different methods exist. I chose to focus on the most used ones.

The absolute error: $e_i = \mid p_i - y_i \mid$
This error simply gives the gap between the prediction and the actual value. The absolute error keeps the same units of measurement as the data we are analysing. Therefore, it is simply interpretable. After performing the mean aggregation over a dataset, the result can be seen as the average error on the dataset. It is easily understandable and helpful to compare different models.

The squared error: $e_i = (p_i - y_i)^2$
The advantage of taking the square of the error is that large errors (higher than one) are

 doi: 10.6342/NTU2022XXXXX

emphasized and have a bigger impact on the final value. And the small errors (smaller than one) will be even smaller. This error does not have the same unit measure than the dataset as we are taking the square. For example, in the case of predicting the latency, the unit of the squared error will be ms^2. Consequently, the result does not have a simple physical meaning and can be difficult to analyse.

3.3 Normalization

The unitary normalization: $N = 1$

This method is the absence of normalization. It is simply a division by one. The advantage is that it does not change the dimension of the point distance. By associating it with the absolute error, we keep the same unit as in the dataset. It is useful to analyse single series and compare different models with the same serie. However, it can not be used to compare multiple series as the result depends on the magnitude of the values from the dataset.

The normalization by the actual values: $N = \mid y_i \mid^{-c}$

This method consists in the division of the point distance by the actual value. c is a constant and its value depends on the point distance method used. For the absolute error $c = 1$ and for the squared error $c = 2$. By doing so, the metrics using this normalization are dimensionless and we can view the result as a percentage by multiplying it by 100. Therefore, it is appropriate to compare multiple series. Nevertheless, if the actual values are zeros or very close to zeros, the method would perform a division by zero so it can not be used. A very small value can be added to the actual value to avoid this issue.

3.4 Aggregation over a dataset

The mean aggregation: $\frac{1}{N}\sum_{i=1}^{N}$

This is the most common aggregation method. It sums all the values of normalized point distances and then divides by the number of elements N. The results is the average of the normalized errors and is easy to understand and interpret.

doi: 10.6342/NTU2022XXXXX

3.5 Choice of error metrics

I have opted to use the absolute error as it effectively measures the point distance, and I have applied unitary normalization to maintain consistency with the data unit. It fits the needs of my work as it enables to compare different models on the same series. Additionally, the actual magnitude of the latency remains unknown and a percentage error may not translate the actual error of the model. On the contrary, the absolute error gives the same result regardless of the magnitude. Furthermore, all three parameters occasionally have zero values, making normalization by actual values problematic due to potential divisions by zero. As mentioned earlier, adding a very small value could avoid this issue, but it would disproportionately weigh errors when the actual value is zero. This is precisely what should be avoided, particularly for the packet loss prediction. Indeed, that would favor even more models that tend to always predict 0.

The combination of the absolute error, the unitary normalization and the mean aggregation gives the Mean Absolute Error (MAE):

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} \mid p_i - y_i \mid$$

The Mean Absolute Error assigns equal weight to every value, treating them uniformly in the final result. However, I observed that for the prediction of the time before the next change of base station, the error can be excessively high for large values, which can inflate the average error. Since my primary focus is accurately determining when the base station will change, precision becomes particularly crucial when the values are low. Therefore, I have developed an alternative error metric that specifically prioritizes low values. MAE α is a Mean Absolute Error with a threshold of α, α being a real number. This metric puts every value of the dataset and the predictions that are higher than α to α exactly, and then perform a regular MAE.

$$\text{MAE } \alpha = \frac{1}{N} \sum_{i=1}^{N} \mid \min(p_i, \alpha) - \min(y_i, \alpha) \mid$$

For this application, we only have to know when the change will happen a few seconds ahead. Therefore, we can focus on the moments when the change is imminent. I first

doi: 10.6342/NTU2022XXXXX

chose $\alpha = 60$ as the parameter for the error metric. This choice assigns greater importance to errors occurring within a minute before the change, while assuming that precise predictions of the time before the next change are unnecessary for larger values, as long as the predicted value exceeds 60 seconds.

To explore the influence of this parameter, the metric was also applied with $\alpha = 30$. With this smaller α, it highlights the errors of the values we are really interested in even more than with $\alpha = 60$. However, it should be noted that this approach tends to overlook a significant portion of the predictions. The models for the prediction of the time before the next change will be evaluated using MAE, but also these two new metrics.

One of the research challenges is to devise solutions for accurately predicting the number of packet loss and avoiding models that consistently predict 0. A first approach is to use a weighted MAE that assigns greater weights to errors when there is at least one packet loss. This adjustment aims to favor models that effectively predict the number of packet loss when the actual value is not zero. The metric that I called "NZ MAE", for "Non-Zero MAE", was designed with this idea.

$$\text{NZ MAE} = \frac{1}{\sum w_i} \sum_{i=1}^{N} w_i \mid p_i - y_i \mid$$

$$\text{with the weights } w_i = 1 \text{ if } y_i = 0$$

$$\text{and } w_i = \beta \text{ otherwise}$$

$$\text{where } \beta \text{ is a constant higher or equal to 1.}$$

This is a weighted Mean Absolute Error with higher weights when the actual values from the dataset are not zero. For zero values, the weights are 1 and for non-zero values, the weights are $\beta > 1$. We can choose the value of β depending on our problem. The bigger β is, the more the errors of non-zero values will be important. Non-zero values account for approximately 0.7% of the datasets, which is roughly 100 times fewer than zero values. Therefore, I chose $\beta = 100$ for the error metric to counterbalance this overwhelming majority of zero values.

doi: 10.6342/NTU2022XXXXX

Another solution is to use a classification model. This idea is driven by the existence of the F_1 score, which is a metric made for classification models. This metric penalizes models that solely predict 0, as the absence of True Positives results in an F1 score of 0. By combining precision and recall, the F1 score produces a value ranging from 0 to 1, using the following formula:

$$F_1 = 2\frac{precision * recall}{precision + recall} = \frac{2tp}{2tp + fp + fn}$$

with tp = number of true positive,

fp = number of false positive,

fn = number of false negative.

Similarly to the use of NZ MAE, this method using classification and the F1 score aims to favors models that do not always predict 0. The prediction of the packet loss will be performed with regression algorithm using NZ MAE as the error metric and with classification algorithm using the F_1 score. The regular MAE will also be used in regression algorithm to assess the impact of the two other methods.

The technique utilizing classification and F_1 score will also be applied for the prediction of wether there is a change of base station.

In contrast, the study of latency prediction will not involve comparing multiple error metrics and instead will focus on other aspects of the problem. The Mean Absolute Error emerges as a fitting error metric for this application as it results in a easily interpretable error that does not depend on the range of the data whose actual value is unsure.

doi: 10.6342/NTU2022XXXXX

Chapter 4. Neural Network

Figure 4.1 shows the system architecture for the Neural Networks model. The algorithm takes $(x_1, x_2, ..., x_M)$ as input, which corresponds to the M features of the dataset, and returns the outpout y. The model is composed of H hidden layers of k units each, and an output layer.

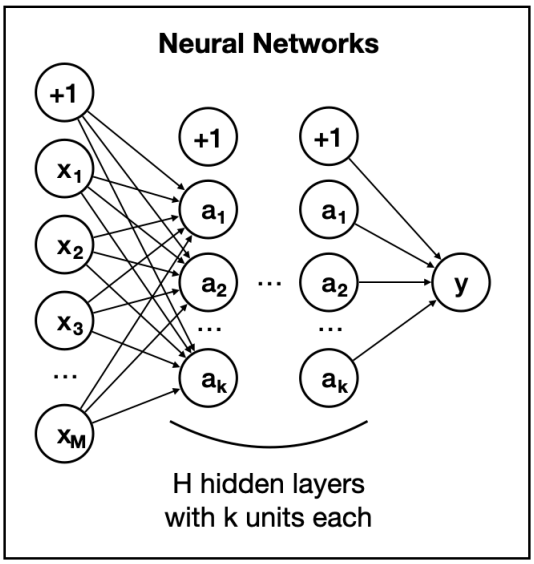


Figure 4.1: Architecture of the Neural Networks algorithm

4.1 Change of base station

As explained before, this study took two different approaches to predict the moment when the phone will change the base station it is connected to. Classification and regression models have been tested.

doi: 10.6342/NTU2022XXXXX

4.1.1 Classification

The classification approach uses the set of features S_1 to predict the feature n°14, i.e. if there is a change of base station, and the F1 score serves as the error metric. A graph with the predictions of the model and the true values for dataset 1 is plotted in Figure 4.2.

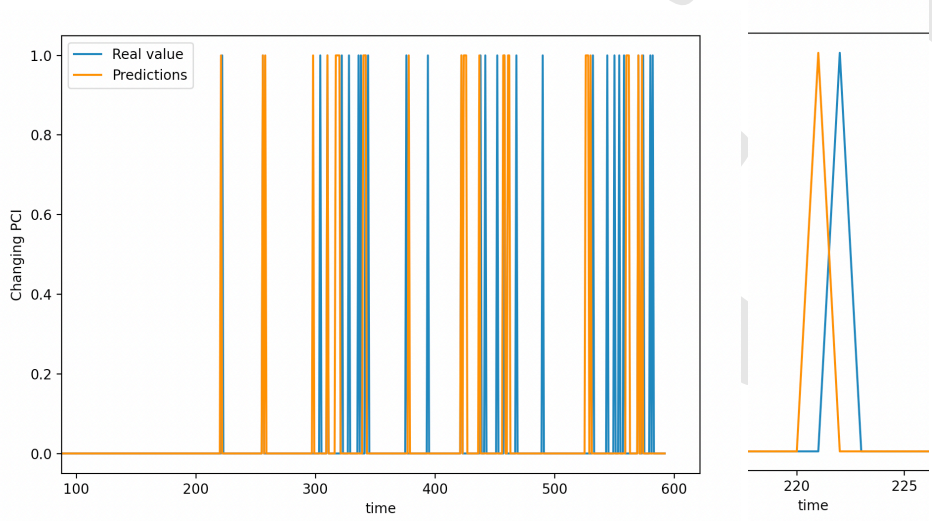


Figure 4.2: Prediction graph for the change of base station on dataset 1 and zoomed graph

This method does not have very good results, exhibiting an F1 score of approximately 0.2. This method may not be the most adequate. One of the limitations is that it only predicts the changement at t+1, without considering the significance of predicting it 1 or 30 seconds too early, which can have varying implications in reality. This issue is illustrated on the zoomed graph of Figure 4.2. Indeed, it shows a base station change prediction that came one second too early. With this method, this prediction will be counted as a False Positive case and the actual change will add a False Negative. This sequence of events is very detrimental for the performance evaluation although the change prediction was really close to the target. Furthermore, the approach does not account for potential delays in predicting the change. Therefore, this approach is not the most suitable for this application, as it does not fully address the timing and delay considerations associated with base station change predictions.

doi: 10.6342/NTU2022XXXXX

4.1.2 Regression

The regression method is tested with two different sets of features: S_1 and S_2. This choice is made to study the influence of the features n°5 and 6 that do not appear in the set S_2. As seen in the preprocessing phase, these two features have strong correlation coefficients and can be harmful to the model's performance. Three error metrics will also be used to inverstigate their effects. These three metrics are MAE, MAE 60 and MAE 30.

Table A.1 from Appendix A is gathering all the errors (in seconds) for all the model tested with the set of features S_1, i.e. the models built with each error metric and both NN and DNN algorithms. The resulting errors are expressed with the three error metrics.

To compare the results, MAE 30 will have a more significant impact as it shows the error close to the moment when the phone changes base station. Therefore, the subsequent tables will only gather the errors expressed with MAE 30 to simplify their interpretation, even if they may come from models built with other error metrics.

		Dataset 1	Dataset 2	Dataset 3
MAE	NN	6.53	6.71	7.75
	DNN	7.60	6.38	7.74
MAE 60	NN	6.61	7.26	6.85
	DNN	6.03	9.93	6.27
MAE 30	NN	6.53	6.71	7.75
	DNN	5.99	5.31	7.70

Table 4.1: Table for the comparison of simple Neural Networks and Deep Neural Networks models

Table 4.1 compares the results from the Neural Networks and Deep Neural Networks models. For each comparison, the smallest error is highlighted in green. We can see that both models perform well and present low errors of about 6s. The DNN models work slightly better for 7 of the 9 cases.

doi: 10.6342/NTU2022XXXXX

The results obtained with each error metricon the Deep Neural Networks algorithm can also be compared with Table 4.2. For each dataset, the smallest error is highlighted in green and the second smallest is in yellow. For the datasets 1 and 2, the use of MAE 30 gives an error more than a second lower than with MAE. MAE 60 works almost as good as MAE 30 for the first one (only $0.04s$ difference), but did not perform well on the second one. Finally, for the dataset 3, MAE 60 has an error of almost a second and a half lower than the two other metrics, MAE 30 being slightly better than MAE.

	Dataset 1	Dataset 2	Dataset 3
MAE	7.60	6.38	7.74
MAE 60	6.03	9.93	6.27
MAE 30	5.99	5.31	7.70

Table 4.2: Table for the error metric comparison

Therefore, it appears that we get better results when we use MAE 60 or MAE 30 to pick the model than with the regular MAE. Except for the dataset 2 for which MAE 60 abnormally did not perform well, using one of the new metrics enables to get an error about a second lower than simply using MAE.

The same reasonning can be done with the set of features S_2. It is interesting to make the same experiments with this set of feature to validate or question the precedent observations. Besides, it will allow us to draw conclusions regarding the effectiveness of this new set of features.

In a similar manner as with the other set of feature, the errors from all models utilizing the set S_2 are shown in Table B.1 from Appendix B, and the error metric MAE 30 is chosen to compare the Neural Networks and Deep Neural Networks in Table 4.3.
The comparison of the models with the different error metrics is provided by Table 4.4, that shows the error with MAE 30 from the DNN models.

doi: 10.6342/NTU2022XXXXX

		Dataset 1	Dataset 2	Dataset 3
MAE	NN	8.00	6.45	6.73
	DNN	6.94	7.66	7.66
MAE 60	NN	8.20	6.54	6.72
	DNN	6.77	5.50	6.71
MAE 30	NN	10.2	7.15	6.65
	DNN	6.93	5.92	7.01

Table 4.3: Table for the comparison between simple NN and DNN models using S_2

	Dataset 1	Dataset 2	Dataset 3
MAE	6.94	7.66	7.66
MAE 60	6.77	5.50	6.71
MAE 30	6.93	5.92	7.01

Table 4.4: Table for the error metrics comparison using S_2

These results confirm the conclusion made earlier with the other set of features. Indeed, the errors from the Deep Neural Networks models are often more than a second lower than those from simple Neural Networks. DNN works better 6 out of the 9 cases.

The models using MAE 60 or MAE 30 have also better results. Across all datasets, the lowest error is achieved by using MAE 60, MAE 30 comes in second place and the regular MAE always has the highest error. For the dataset 1, the three errors are quite close. For the datasets 2 and 3, the gap is wider. Indeed, there is respectively more than 2 seconds and almost a second between MAE and MAE 60. For both datasets, the error from MAE 30 is in between the two but closer to the best one.

The predictions from the models with Deep Neural Networks algorithm and each error metric have been plotted on Figure 4.3. The model using MAE (in orange) has more outliers and appears to deviate further from the actual values (in red). This visual observation aligns with the quantitative analysis.

doi: 10.6342/NTU2022XXXXX

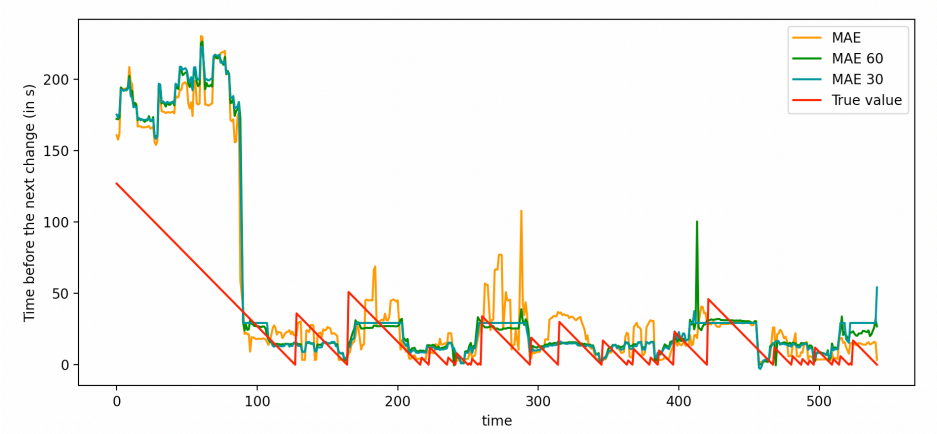


Figure 4.3: Graph of the time before the next change prediction upon time for the dataset 1

However, it is worth noting that for instances where the actual values are relatively high at the beginning of the graphs, the predictions from this model are slightly closer to the target. As previously discussed, the accuracy of predictions when the next change is far in the future is less crucial, and the newly designed error metrics aim to prioritize important moments. In this particular example, the utilization of these metrics enabled the selection of different models that perform better during critical periods despite being less effective on high values.

4.1.3 Conclusion

We saw that Deep Neural Networks are more suited than classical Neural Networks for this application.

As expected, we get better results by using the new error metrics designed for this application in the model construction. However, these metrics could favor the models that predict an average value most of the time, such as 15s or 30s. These models would have been ignored with the classical MAE. The risk is even bigger with MAE 30, as all the values are in a shorter range. Therefore, the best solution is to use MAE 60 to pick the best model.

Finally, the addition of the features n°5 and 6 in the set S_1 does not translate in better results. Therefore, the set S_2 should be prioritized to keep a simpler model.

doi: 10.6342/NTU2022XXXXX

4.2 Latency

The prediction of the latency using Neural Networks algorithm has been performed with the error metric MAE and two different sets of features S_3 and S_4 to study the influence of the features n°5 and 6 once again. These two features are only present in the set S_3.

4.2.1 Results

I compared the results obtained with classical Neural Networks and Deep Neural Networks models. Table 4.5 gathers the results calculated with the Mean Absolute Error (in ms) for the three datasets and the two sets of features defined earlier.

		Dataset 1	Dataset 2	Dataset 3
S_3	NN	10.8	225	23.4
	DNN	22.4	556	19.1
S_4	NN	7.54	55.6	21.7
	DNN	9.76	52.5	18.9

Table 4.5: Table gathering the errors for NN and DNN models and both sets of features

The second dataset does not yield satisfactory results when combined with the first set of features, S_3. That can be explained by the range of the values. Indeed the values of Latency are going from $-389ms$ to $1419ms$, and the average value of the training set is $-354ms$, very close to the minimum value. In contrast, the average value in the testing set is $552ms$. Therefore, accurately predicting the latency from the testing set becomes more challenging, as the training set is predominantly composed of very low latency values and does not adequately represent the full range of latency values in the testing set. However, we obtain more promising results when utilizing S_4 as the set of features.

Besides, we see that NN and DNN give similar results. Both are better for half of the experiments. It is worth noting that Deep Neural Network models typically entail longer computation time. Given that the results obtained from DNN models do not exhibit explicit superiority over classical Neural Networks, it is reasonable to opt for the latter to

doi: 10.6342/NTU2022XXXXX

avoid the additional computational overhead. Hence, the use of classical Neural Networks is sufficient for this application.

A good accuracy is achieved for the datasets 1 and 3. The errors from the two sets of features are respectively about $10ms$ and $20ms$. Even if the exact values of the latency are not in the dataset, these errors can still be interpreted as small values considering that the range of the latency values, i.e. the gap between the maximum and minimum values, is around $1000ms$ for the three datasets.

Table 4.5 shows the difference of performance for the models using the set of features S_3 and those using S_4. Indeed, the models utilizing the second set always have a smaller error. For the dataset 3, the gap is the smallest. Only $0.2ms$ for the DNN model but still almost $2ms$ for the NN model. For the dataset 1, the error of the model using S_3 is even more than twice as much for the DNN model and more than $3ms$ higher for the NN model. Concerning the dataset 2, the NN error is 4 times higher and the DNN error is more than 10 times higher with S_3. Therefore, the bad accuracy observed before is not entirely due to the problem of range but also to the use of the features n°5 and 6.

We can visualize the predictions of the NN models for the second dataset in Figure 4.4.

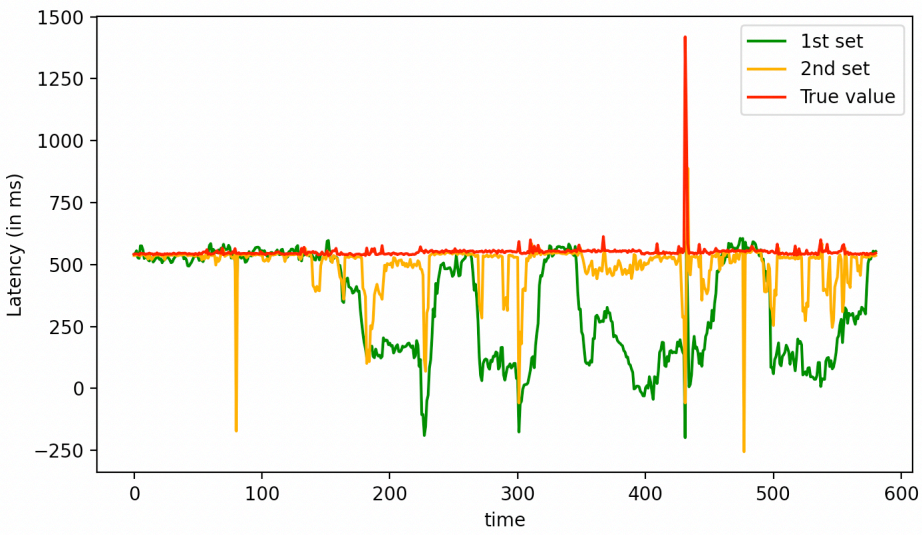


Figure 4.4: Latency comparison graph for the dataset 2

doi: 10.6342/NTU2022XXXXX

As expected after seeing the error results, the model using S_3 (in green) has more outliers. We observe that both model have a tendancy to predict lower values. This can be due to the issue of range for this dataset, as the model has been trained on lower values. We can draw the graphs for the two other datasets, using Neural Networks models.

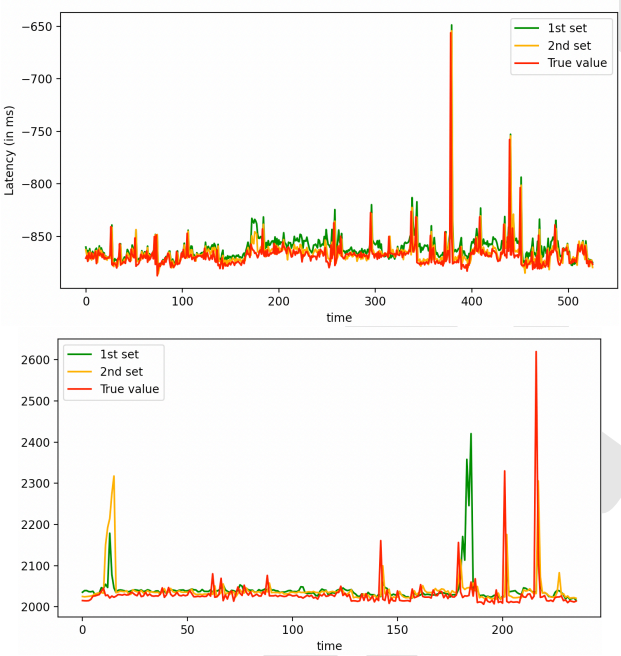


Figure 4.5: Latency comparison graphs for datasets 1 and 3

Similarly to the previous graph, the predictions from the model using S_4 (in orange) are closer to the true values (in red) than the model using S_3 (in green). These three graphs from Figures 4.4 and 4.5 showed what the table already stated: the predictions are much better without the redundant features n°5 and 6.

4.2.2 Conclusion

We established that the most adequate model was simple Neural Networks and we just saw that the set of features S_4 is more appropriate. Then, the results of the NN models using this set of features can be analyzed.

The errors of the three datasets are (in ascending order of the dataset number) $7.54ms$, $55.6ms$ and $21.7ms$ for latency values ranges respectively of $1659ms$, $1808ms$ and $731ms$. Therefore, the error from the dataset 1 represents only 0.5% of its range. For the dataset 2,

doi: 10.6342/NTU2022XXXXX

it is 3.1%. And for the last dataset, the proportion is 3.0%. To see how close the predictions are, the graphs concerning only the second set of features can be drawn.

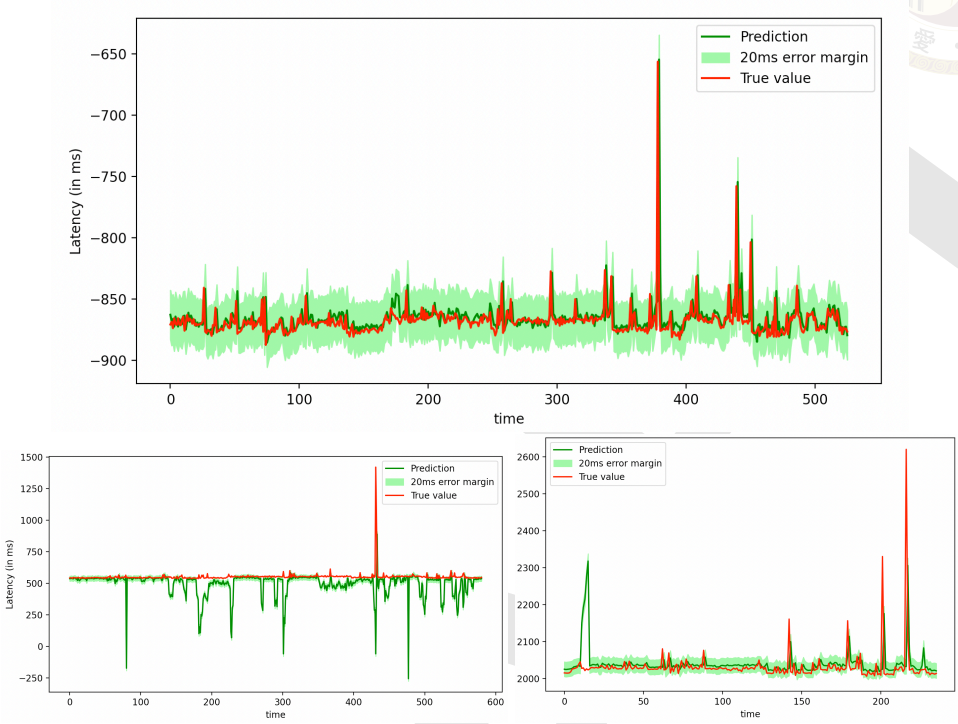


Figure 4.6: Latency graph for the three datasets, using the 2nd set

On Figure 4.6, the graphs include the predictions, a 20ms error margin around these predictions and the true value from the testing sets. For the dataset 1, whose graph is at the top, and the dataset 3, with the graph at the bottom right, the true value is almost always within the 20ms error margin from the prediction. Therefore, the models work well. Concerning the graph of the second dataset, there are some outliers but a big part of the testing data is still in the error margin.

We can conclude that Neural Networks models (classical or deep) can precisely predict the latency of the signal with the data we have at a time t. Simple Neural Networks will be prefered to avoid extra computing time and the set S_4 is more appropriate. However, this model does not forecast the latency in the future. Now that it is proven that the prediction is doable with the datasets of the study, the use of time series models will help predict it a few seconds in the future.

doi: 10.6342/NTU2022XXXXX

4.3 Packet Loss

To predict the number of packet losses, three different approaches have been tested to try to solve the problem of the overwhelming majority of zeros in the datasets. A classification model predicting whether there is at least one packet loss, and two regression models. One is using a weighted error metric to predict the number of packet loss and the other is using the modified feature n°17 called ”enlarged loss”. Finally, a last model was tested with a regression algorithm, using a regular MAE and the actual number of packet loss. This serves as a control experiment, allowing us to assess whether the three proposed solutions have a substantial impact on the prediction accuracy.

4.3.1 Classification

Considering the relatively infrequent nature of packet losses, it holds greater significance for the model to effectively predict the occurrence of any packet losses rather than precisely quantifying their exact number. Therefore, using a classification model that discards information about the actual number of packet losses is not problematic. The focus is primarily on detecting instances where packet losses occur. The model will predict the feature n°16 and be evaluated by the F_1 score.

As explained and showed with the experiments to predict the changes of base station and the latency of the signal, the presence of the features n°5 and 6 in the datasets is not beneficial for the model performance. Therefore, the set of features used (S_5) does not include them.

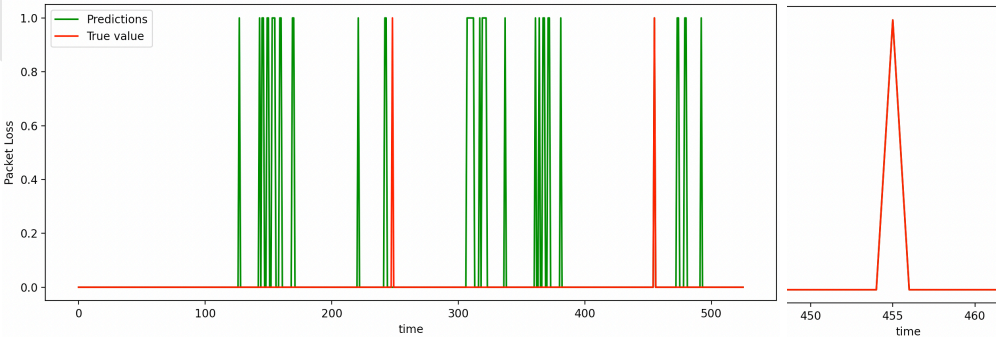


Figure 4.7: Graph of packet loss presence upon time for the dataset 1 and zoomed graph

doi: 10.6342/NTU2022XXXXX

The graph of the predictions and the true values for the three datasets are plotted in Figure 4.7. There are only two times when there is at least one packet loss in the true values of the testing set and the zoomed graph from Figure 4.7 shows that the model successfully predicted one of this two, around time = 455s. However, the model produces many False Positives, when the prediction (in green) is 1 but the true value (in red) is 0.

For the dataset 2, five moments when there is at least one packet loss are present in the true values of the testing set and the zoomed graph from Figure 4.8 shows that the model successfully predicted one. Besides, there are a lot less False Positives than for the first dataset, only a few are spotted around time = 375s.

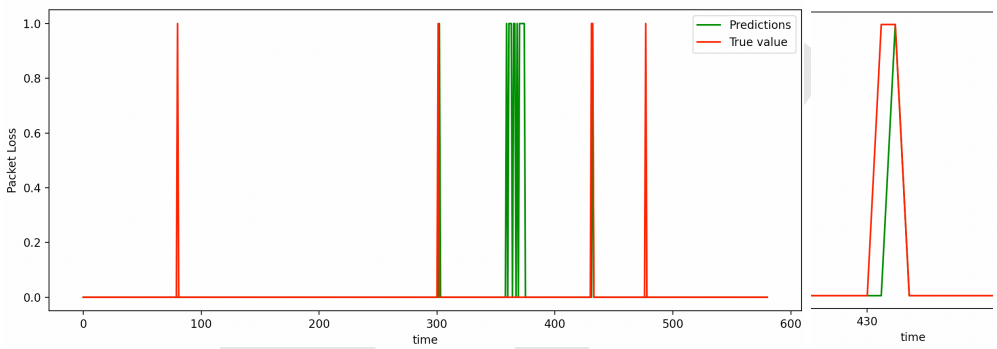


Figure 4.8: Graph of presence of packet loss upon time for the dataset 2 and zoomed graph

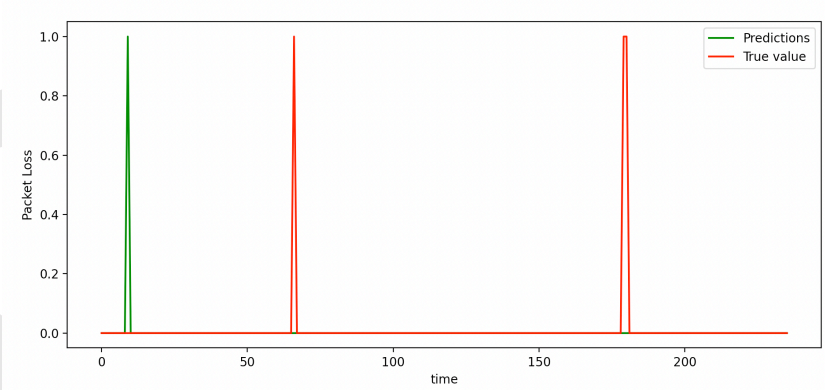


Figure 4.9: Graph of presence of packet loss upon time for the dataset 3

Finally, for the dataset 3, none of the three moments where there is at least one packet loss was detected, as depicted on Figure 4.9. Indeed, the model predicted almost only zeros, except one False Positive.

doi: 10.6342/NTU2022XXXXX

4.3.2 Regression with loss

In this second approach, the exact number of packet losses is used. This model takes the set S_6 as input and the feature loss (n°11) as output. The NZ MAE and the regular MAE are both used.

Figure 4.10 shows the graphs for the first dataset. The true values from the testing set are represented in red, while the predictions from the model using NZ MAE and MAE are respecively in green and orange. All the predictions from both models are very close to zero, so they missed the two times when packet losses happen.

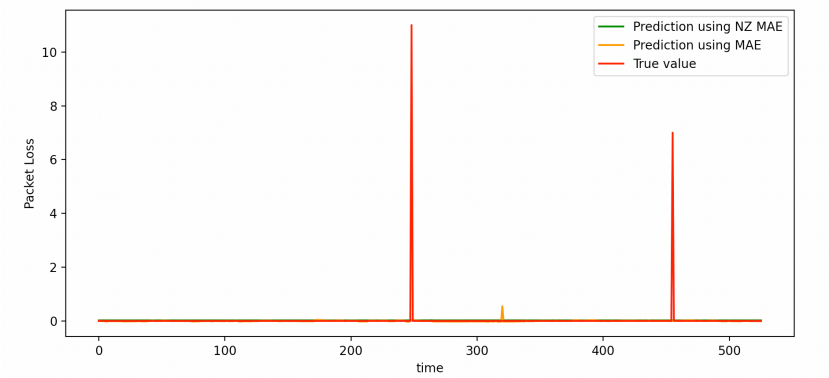


Figure 4.10: Graph of number of packet loss upon time for the dataset 1

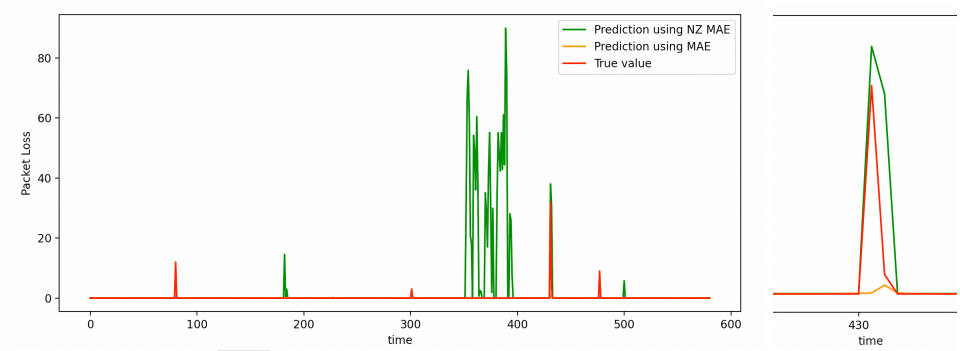


Figure 4.11: Graph of packet loss number upon time for the dataset 2 and zoomed graph

Figure 4.11 is illustrating the results for dataset 2. Again, the model that used MAE produced very low predictions, all close to zero. However, when the new metric NZ MAE is used, some variations are seen in the predictions. This leads to a few false positive cases, particularly around time = 375s, but two occurrences of actual packet losses were successfully predicted close to time = 430s, as depicted in the zoomed-in graph.

doi: 10.6342/NTU2022XXXXX

For the dataset 3, the graphs are plotted on Figure 4.12. It shows that both models have some variations in their predictions. The model built by using NZ MAE predicted non-zero packet loss for one of the three times when packet loss happens in this dataset, while the model using MAE missed all three.

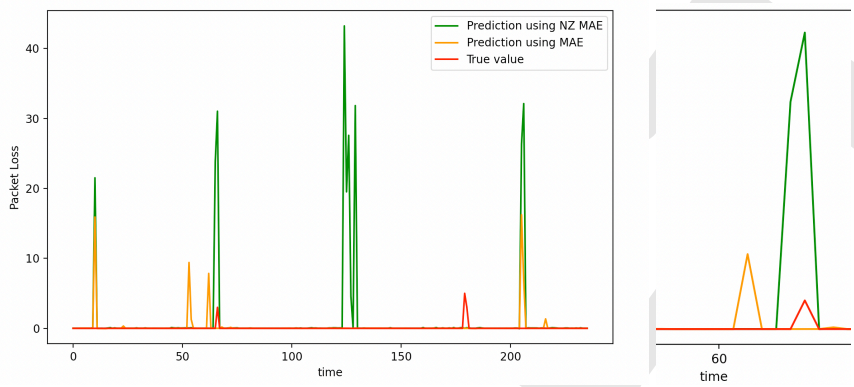


Figure 4.12: Graph of the packet loss number upon time for the dataset 3 and zoomed graph

4.3.3 Regression with enlarged loss

The third approach uses the error metric MAE and considers the set of feature S_7 to predict "enlarged loss", i.e. feature n°17. The results for dataset 1 are plotted on Figure 4.13. The two enlarged peaks are correctly predicted. Indeed, the predictions are following the global trend of the feature, with an abnormal increase around time = 350s.

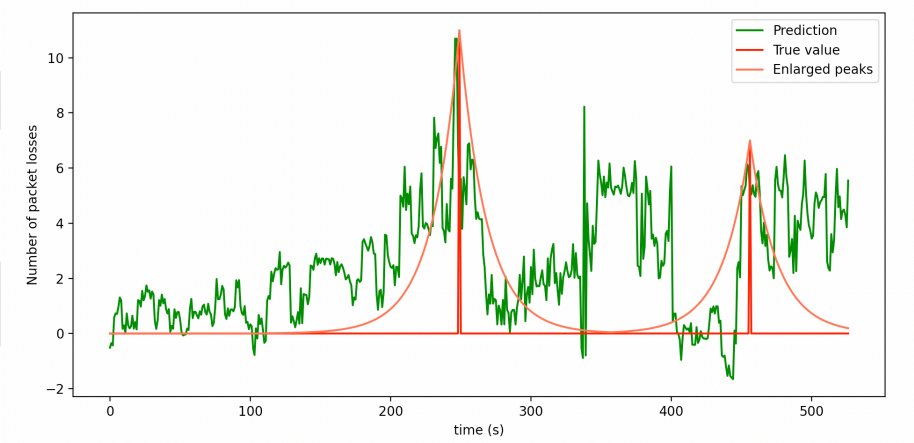


Figure 4.13: Graph of the enlarged loss prediction upon time for the dataset 1

Figure 4.14 shows the graphs for the dataset 2 and reveals that the model succesfuly predicted two enlarged peaks, which correspond to three non-null packet loss values.

doi: 10.6342/NTU2022XXXXX

Two other enlarged peaks have not been detected and many false positive cases happened around time = 200s.

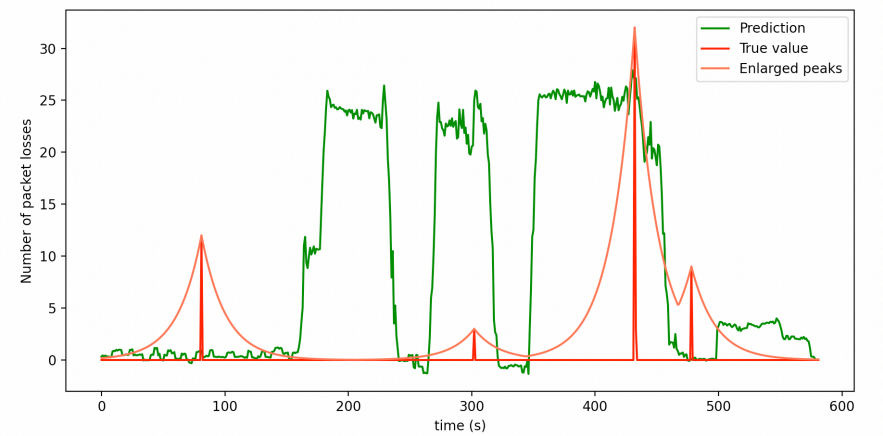


Figure 4.14: Graph of the enlarged loss prediction upon time for the dataset 2

Similarly as for the classification model and model using a regression algorithm and NZ MAE, non-null packet loss are predicted from time = 375s. Although these models have been trained differently, they all lead to the same conclusion. The likelihood of packet losses was certainly high at that specific time. In addition to their primary function of packet loss prediction, these models offer valuable insights and provide useful information about the prossibility of packet losses occurrence.

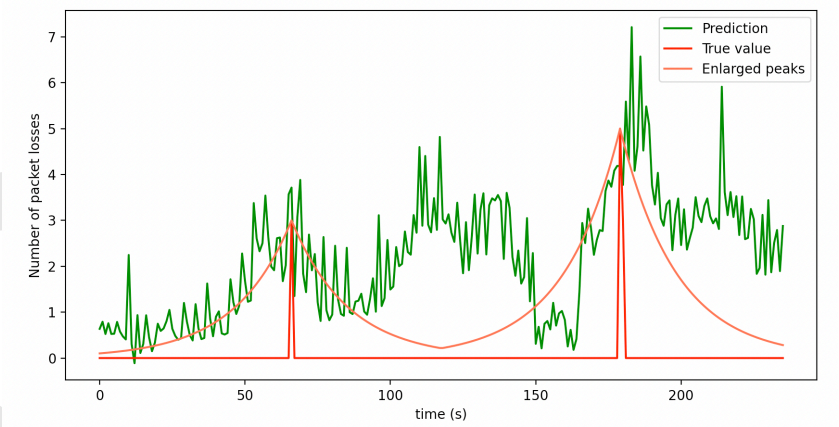


Figure 4.15: Graph of the enlarged loss prediction upon time for the dataset 3

For the dataset 3, both enlarged peaks and therefore all three packet losses occurence have been correctly predicted by the model, as shown on Figure 4.15. However, another bump can be observed in between these two, that does not correspond to an actual enlarged peak.

doi: 10.6342/NTU2022XXXXX

Predicting the new feature ”enlarged loss” rather than the actual number of packet loss has proven to be beneficial for the model's performance. Despite generating some false positive cases, the model accurately identified 8 out of the 10 non-null packet losses from the testing sets. Furthermore, the feature ”enlarged peaks” provides an additional indication by anticipating the occurrences of packet losses. By studying the evolution of the predictions, it becomes possible to foresee the moments when packet losses are likely to happen.

4.3.4 Conclusion

It is difficult to conclude as the number of packet losses is very low. It represents only 0.7% of the three datasets and there is only a few of them in the testing sets (respectively two, five and three). Making conclusions on such a little sample is very unsure. Not predicting any packet loss does not necessarily mean that the model is bad. And on the contrary, succesfully predicting one packet loss could be a lucky prediction. However, as the experiments were carried out on three different datasets, the luck factor is decreased. Still, this conclusion must be taken carefully.

		Dataset 1	Dataset 2	Dataset 3
Class.		1 packet loss detected but many fp	1 packet loss detected	almost only zeros
Regr.	MAE	Only zeros	Only zeros	Some variations but 0 packet loss detected
	NZ MAE	Only zeros	2 packet losses detected	1 packet loss detected
	Enlarged loss	The 2 packet losses detected but many fp	3 packet losses are detected but many fp	All 3 packet losses detected but many fp

Table 4.6: Table gathering the conclusions from every model tested

Table 4.6 gathers the observations made for the classification and regression models. The cases of succesful prediction of at least one non-zero packet loss are highlighted in green in the table.

doi: 10.6342/NTU2022XXXXX

The only model that did not predict any instance of packet loss is the regression algorithm predicting the feature ”loss” and using MAE. This was the control experiment and it shows that models tend to always predict 0. All three other models performed better and have succesfully predicted several non-null packet loss instances. The models using classification and regression with NZ MAE both predicted a few packet losses succesfully for two datasets and have almost only zeros as predictions for one. It can be noticed that the classification model generated more false positive cases. Finally, the model predicting the feature ”enlarged loss” succesfuly predicted almost all instances of non-null packet losses. However, it also produced many false positive cases for every dataset. This is not a surprise as the variable predicted is not the actual number of packet loss but a modified version that increases this number for most of the time.

The results have demonstrated the benefits of utilizing a classification algorithm, the NZ MAE error metric, or the ”enlarged loss” feature in the prediction of the number of packet losses. These methods put forward models that do not always predit zero. If the primary objective of the prediction is to accurately determine the exact number of packet losses at each time, the model built with NZ MAE should be employed, as it exhibits fewer false positive cases than the others. However, the model predicting the ”enlarged loss” feature can also be useful, as it successfully captured nearly all instances of packet losses. The evolution of the predictions can be analyzed to conclude on the packet loss likelihood. Indeed, the ”enlarged loss” feature exhibits a gradual increase leading up to the occurrence of packet losses. It can serve as an indication of the likelihood of packet losses occurring, which may be particularly relevant within the study’s context. When selecting the safest setting, the exact number of packet loss may be of lesser importance than understanding the overall trend in the evolution of this feature.

doi: 10.6342/NTU2022XXXXX

Chapter 5. Time series models

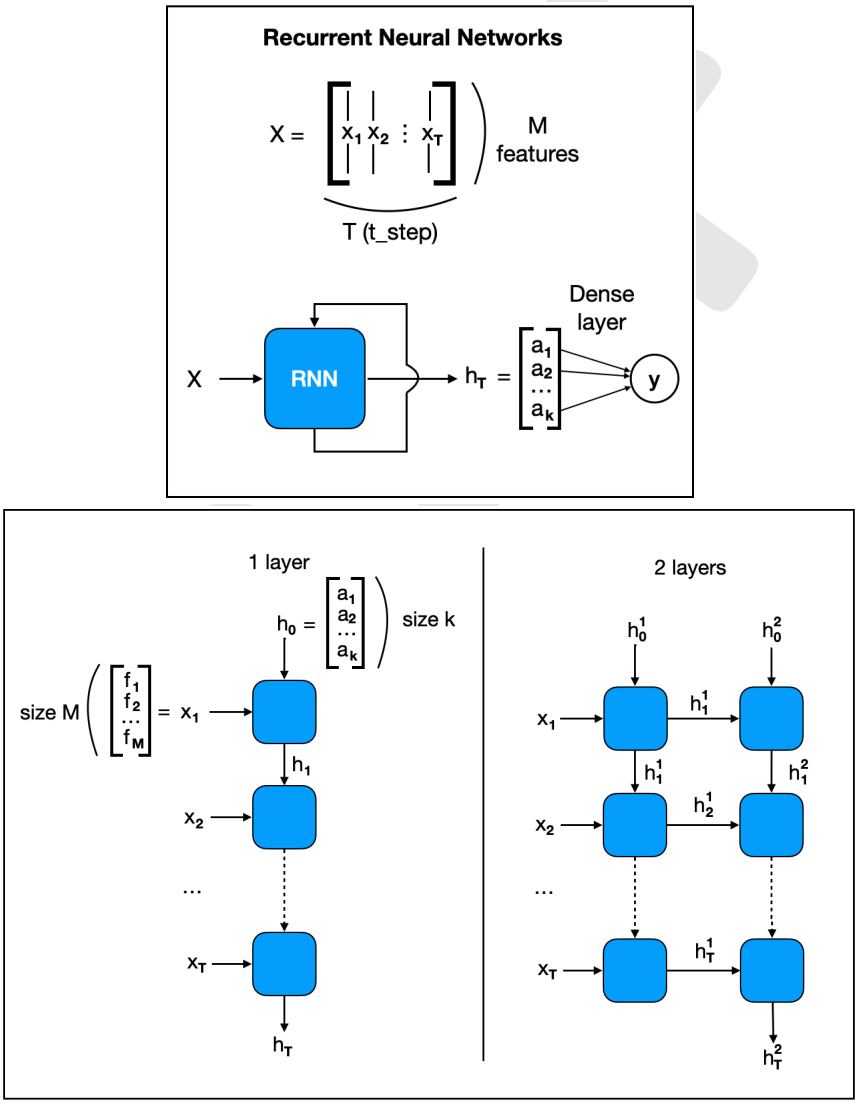


Figure 5.1: Architecture of the Reccurent Neural Networks algorithm

The study also considered time series models. The Reccurent Neural Networks algorithm has been used for the prediction of the three parameters. Simple RNN and Long Short Term Memory (LSTM) cells have been tested within this algorithm.

doi: 10.6342/NTU2022XXXXX

Figure 5.1 shows the architecture of a Recurrent Neural Networks model. The first part explains that the input is the matrix X (of size $M * T$) composed of the M features values during a period of T. X can be divided as T vectors $x_1, ..., x_T$ each composed of the values of the M features at a given time. The input goes into the Recurrent Neural Networks layers, which return a vector h_T of size k. k can be interpreted as the number of units in a hidden layer in the classical Neural Networks algorithm. Then, the vector is going through a Dense layer and the model returns the output y.

The second part details the architecture of the Recurrent Neural Networks, for $H = 1$ and $H = 2$ hidden layers. In this figure, the blue squares can be either Simple RNN or LSTM cells. A cell is taking a vector x_i composed of the M features at the time i and another vector h_{i-1} that contains the values $(a_1, ..., a_k)$, and returns the updated vector h_i. For the LSTM model, a third input called c_{i-1} is needed for each cell. The advantage of this configuration will be explained later.

In the case of simple RNN, the i^{th} cell is a dense layer of size k, that receives the concatenated input of x_i with size M and h_{i-1}, whose size is k. The activation function is the hyperbolic tangent $tanh$ function, which enables a faster convergence than the sigmoid function. The resulting formula is:

$$h_i = tanh(W^T * concat(x_i, h_{i-1}) + b)$$

with W a matrix with size $(k + M) * k$ and b a vector of size k

However, a limitation of this type of cells is the issue with vanishing gradient that makes the learning harder and slower. At each step, the algorithm updates the weights of W to minimize the loss function L such as:

$$W \leftarrow W - \alpha \frac{\partial L}{\partial W}$$

$$\text{with } \frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$

$$\text{and } \frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial h_t} (\prod_{i=2}^{t} \frac{\partial h_i}{\partial h_{i-1}}) \frac{\partial h_1}{\partial W}$$

doi: 10.6342/NTU2022XXXXX

This last expression includes a multiplication of several derivatives of h. This corresponds to derivatives of the $tanh$ function, that are giving values in $[0, 1]$. When T is large, this expression tends towards 0. Therefore the model will not learn significantly in reasonable time.

A solution to this problem is to use Long Short Term Memory (LSTM) cells. These cells divide the information between what is important at short termn through the hidden state h_i, that can be compared to the output of a simple RNN cell, and what count at long term, with the cell state c_i.

The operation of a LSTM cell can be sum up in three steps. First, the forget gate, that consists of a dense layer with a sigmoid activation function, aims to detect valuable informations from the past cell state. Then, the input gate chooses the relevant informations to be added to the new cell state. Finally, the next hidden state will be generated by the output gate by extracting important short term informations from the newly generated cell state.

These two kinds of cell will be compared in the following experiments. We saw that LSTM should peform better, especially with long input sequences, as the vanishing gradient issue is prevented. However, it usually implies longer computation time and the performance gap has to be significant enough to justify it.

The use of this time series algorithm requires the determination of two additional variables: t_{step} and t_{pred}.

t_{step} is the step of time that is taken into account for the input data. It represents how far we go into the past datas to predict the future values.

t_pred is the gap of time between the last data that is in the input and the prediction. It represents how far in the future the data predicted by the model is.

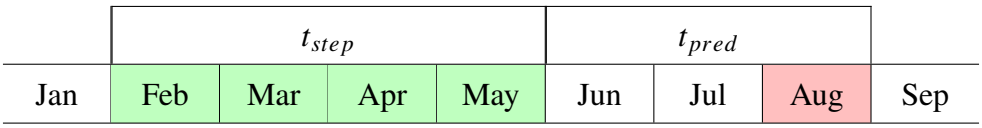


Figure 5.2: Figure explaining the meaning of t_{step} and t_{pred}

doi: 10.6342/NTU2022XXXXX

To visualize the meaning of t_{step} and t_{pred}, Figure 5.2 is an example with monthly data. Each box represents one month and the corresponding data. The time goes from left to right, following the order of the months. The boxes colored in green are the data taken in the input. The box colored in red is the output, i.e. the data the model has to predict. In this example, four steps are made into the past to build the input. Indeed, the data from February to May are taken, so $t_{step} = 4$. Three steps are made from the last "green box" to the prediction, from May to August, so $t_{pred} = 3$.

Regarding the prediction of the latency, the ARIMA algorithm has also been tested. The reasons of this choice and the operation and functioning of this algorithm will be detailed later.

With the previous experiments on simple Neural Networks models, the sets of features that do not include the features n°5 and 6 appeared to be more appropriate. Therefore, these two features will not be contained in the future sets for the time series approach.

5.1 Base station

During the Neural Networks models experiments, it has been established that the best method to predict the moments when the phone changes the base station it is connected to is to consider the time before the next change. This prediction using time series model will include the comparison of two sets of features, RNN and LSTM cells and 1-layer and deep models. The two sets of features studied are S_2 and S_8. The second one contains fewer features, with only three compared to the seven that compose S_2.

As explained, the time before the next change is forecasted. Therefore, the next change is already predicted in the future and the variable t_{pred} is not needed in this case. For generalization purposes, we take $t_{pred} = 0s$.
Then, the value of t_{step} must be defined. Ideally, this value must be the smallest possible to use less data and therefore have a smaller computation time and have more samples in

doi: 10.6342/NTU2022XXXXX

the dataset. To determine the best value, the cross validation method is used to study the influence of t_{step} on the Mean Absolute Error for both sets of features. The two criteria are a small t_{step} value and a low resulting error.

The results using RNN and LSTM for the evolution of the error depending on the value of t_{step} are very similar and the graphs present the same curves. Figure 5.3 shows the graphs obtained using RNN, but the conclusions are made for both algorithms.

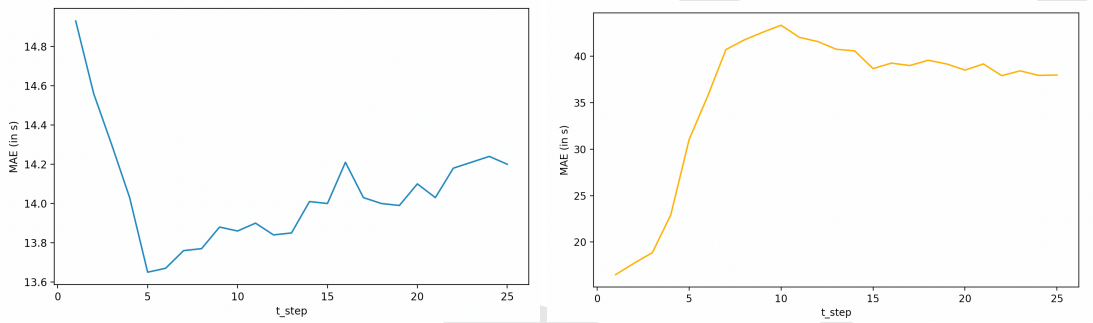


Figure 5.3: Graphs of the Mean Absolute Error against t_{step} with S_8 (in blue) and S_2 (in orange)

The set S_2, whose corresponding graph on Figure 5.3 in in orange, does not seem to be suited to time series model. Indeed, the lowest error is achieved for $t_{step} = 1$ and then the value is increasing until a stabilization on a high value. The interest of time series models is to have several time steps in the input, but this leads to a bigger error with this set. This issue may be caused by the big number of features used. The testing phase will tell if this assumption is confirmed. For the other set, the error is not widely varying, but a decrease is still observed until $t_{step} = 5s$. The models with the set S_8 will be built with this t_{step} value.

5.1.1 Results

Table 5.1 illustrates the difference between the two sets of features. The results are expressed in seconds with the error metric MAE 30. The smallest error of each comparison is highlighted in green. The models using the set S_8 are resulting in lower error for 10 out of the 12 comparisons. Notably, the difference is big for the dataset 2, for which the models with S_8 have errors twice as small as with S_2.

doi: 10.6342/NTU2022XXXXX

		1 hidden layer			2 hidden layers		
		Dataset 1	Dataset 2	Dataset 3	Dataset 1	Dataset 2	Dataset 3
RNN	S_2	7.85	10.63	7.46	10.61	10.63	7.44
	S_8	7.11	5.20	7.76	7.25	5.28	7.67
LSTM	S_2	7.40	10.63	8.12	7.48	10.63	7.58
	S_8	6.68	5.18	7.76	6.55	5.31	7.37

Table 5.1: Table gathering the errors for base station change prediction with time series models

The comparison graphs of both sets with a 1-layer LSTM model are plotted in Figure 5.4 for the datasets 1 and 2. The set of features S_2 is called Set 1 and corresponds to the orange line, while S_8 is named Set 2 and is seen with the green line.

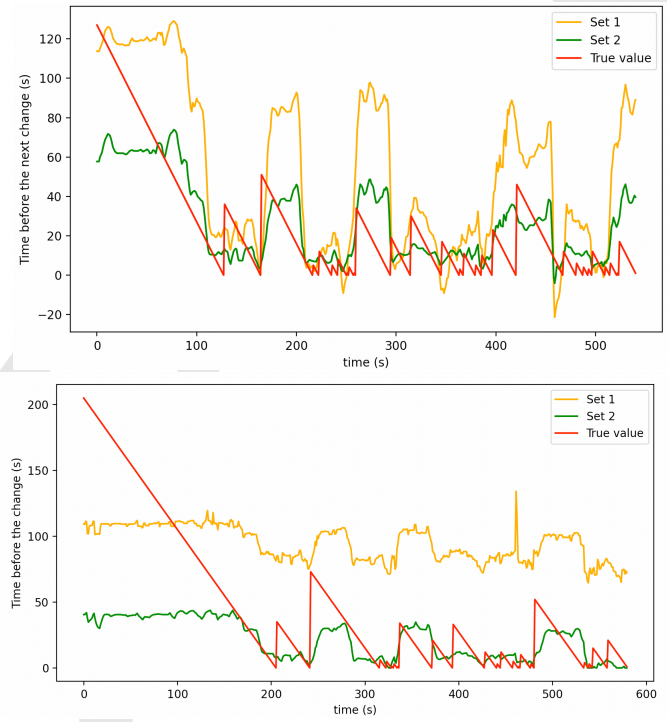


Figure 5.4: Prediction graphs using deep RNN for both sets on dataset 1 and 2

For the two datasets, the predictions from the models assiociated with S_8 (in green) are visibly closer to the true values. Indeed, the orange line presents a big gap with the red one, especially for the dataset 2 whose graph is at the bottom. As assumed before, the models with S_8 are performing better and the other set is not suited for the use of time series models.

doi: 10.6342/NTU2022XXXXX

		Dataset 1	Dataset 2	Dataset 3
1 layer	RNN	7.11	5.20	7.76
	LSTM	6.68	5.18	7.76
Deep model	RNN	7.25	5.28	7.67
	LSTM	6.55	5.31	7.37

Table 5.2: Table gathering the errors for the comparison between RNN and LSTM

The models with the more pertinent set will be used for the next comparisons, including between Recurrent Neural Networks and Long Short-Term Memory algorithms that can be seen in Table 5.2. The errors are very close between the models using RNN and LSTM for every comparison. The biggest difference is seen on dataset 1, for which the LSTM models have errors of approximatevely 0.5s lower than with RNN. Overall, the LSTM models are performing slightly better.

For the final comparison between the 1-layer and deep LSTM models, the errors of the models are expressed with three different error metrics (MAE, MAE 60 and MAE 30) to have better insights on the performances. The combined used of these three error metrics enables to have an estimation of the global error with MAE and a view on the errors on smallest data with MAE 60 and MAE 30. The errors in seconds are gathered in Table 5.3.

		Dataset 1	Dataset 2	Dataset 3
MAE	1-layer	13.69	31.59	9.54
	Deep	12.87	30.02	9.24
MAE 60	1-layer	9.67	12.72	9.53
	Deep	9.34	11.04	9.15
MAE 30	1-layer	6.68	5.18	7.76
	Deep	6.55	5.31	7.37

Table 5.3: Table gathering the errors for the comparison between 1-layer and deep LSTM

Except when it is expressed with MAE 30 on dataset 2, the error is always higher with

doi: 10.6342/NTU2022XXXXX

the 1-layer model. Even if the gaps are not wide, they are consistently in favor of the deep model and are approximatevely the same for the three datasets. This manifests the reliability of both model and the constant superiority of the deeper model. The graph of Figure 5.5 shows how close the predictions from both models are for the dataset 1. Having smaller errors calculated with all three error metrics MAE, MAE 60 and MAE 30 demonstrates the better performance of the deep model on all levels, including on every predictions aggregated on the testing sets and on more important values.

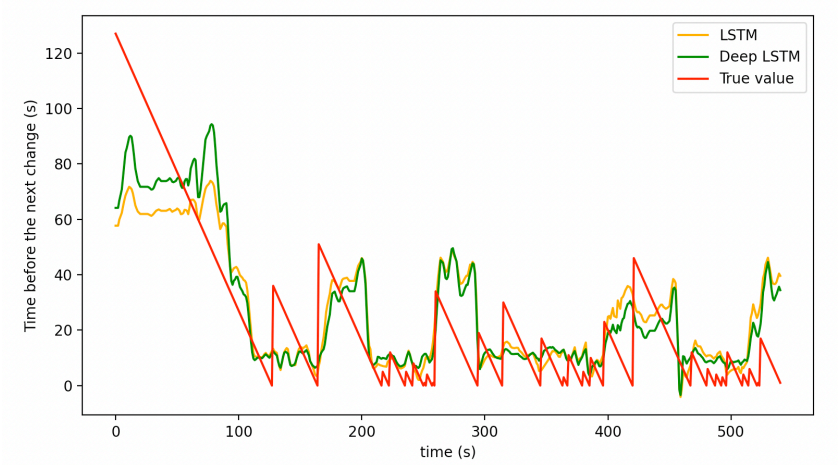


Figure 5.5: Comparison graphs between 1-layer and deep LSTM on dataset 1

5.1.2 Conclusion

As seen in the several comparison made for the prediction of the change of base station with time series model, the set S_8, that contains fewer features, should be use. Besides, the use of LSTM cells has shown better results that simple RNN ones. Finally, a deeper model proved to be beneficial for the performance.

The predictions from the chosen model on the three datasets are plotted in Figure 5.6. For all the datasets, the predictions are following the same evolution than the true values. Notably, the lower tips are often within the error margin, meaning that the moments of the change are succesfully predicted. The time before the next change of base station is well forecasted and the model gives satisfactoy results on every dataset.

doi: 10.6342/NTU2022XXXXX

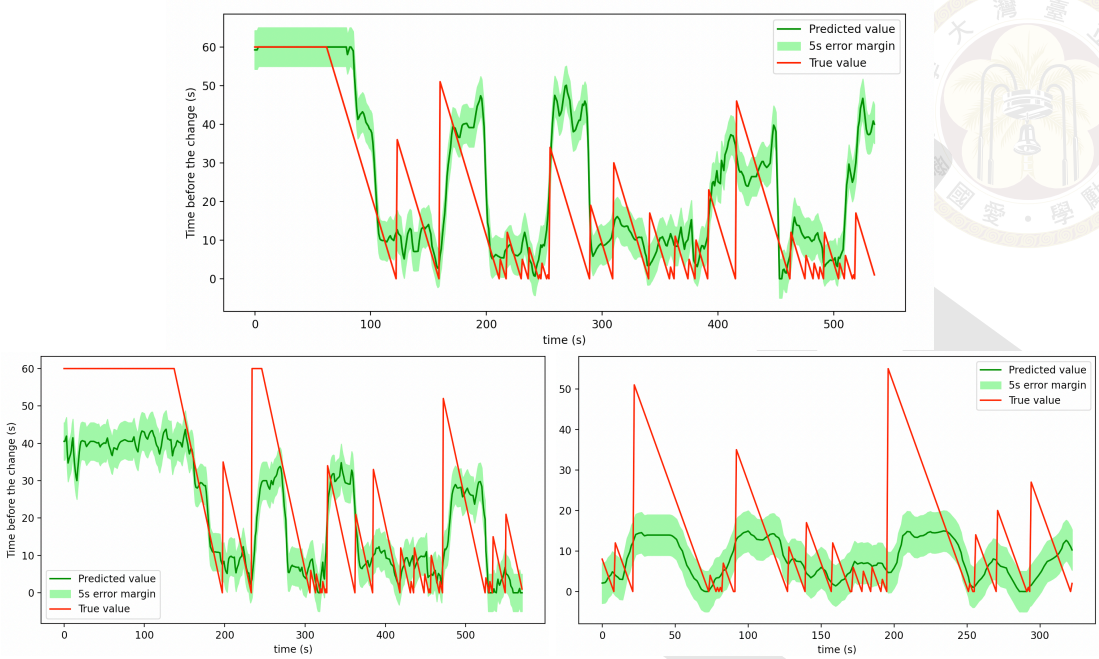


Figure 5.6: Change of base station prediction graphs for the three datasets

Finally, the result from this time series model is compared to the results obtained before with a simple Neural Networks model. Table 5.4 is used for this final comparison. The errors from the two models are expressed in seconds with MAE, MAE 60 and MAE 30.

		Dataset 1	Dataset 2	Dataset 3
MAE	NN	25.8	82.2	8.50
	LSTM	12.87	30.02	9.24
MAE 60	NN	8.12	12.0	8.50
	LSTM	9.34	11.04	9.15
MAE 30	NN	6.77	5.50	6.71
	LSTM	6.55	5.31	7.37

Table 5.4: Table gathering the errors for the comparison between Neural Networks and time series deep LSTM models

For the dataset 3, the Neural Networks model performed better, but the gap is smaller than a second for each error metric. The two models have similar performance and are reliable. The time series model outperformed the simple NN model for the two other datasets, with a MAE more than two times lower. It can be concluded that LSTM should be prioritized.

doi: 10.6342/NTU2022XXXXX

5.2 Latency

5.2.1 Recurrent Neural Networks

The experiments include the study of two sets of features: S_9 and S_{10}. These two sets are very different as S_9 countains seven features and S_{10} is only composed of the latency feature. The use of this second set may cause loss of information but keeps a simpler model. In a time series model, there can be some misinformation due to the big amount of data with a lot of features taken at each time. Similarly to the decision that has been done for the Neural Networks models, the Mean Absolute Error will be used as the error metric.

One of the aims of the study is to make predictions a few seconds ahead. t_{pred} should not be too small to fulfill this goal and have a real impact. On the contrary, the value should not be too big to keep a feasible and meaningful model. Therefore, I arbitrarily chose $t_{pred} = 10s$ after testing a few values.

Similarly to the base station part, the value of t_{step} is defined by investigating its influence on the Mean Absolute Error. Once again, the evolution of the error for the models using RNN and LSTM are very similar and the graphs of the RNN models depicted in Figure 5.7 are sufficient to draw conclusions.

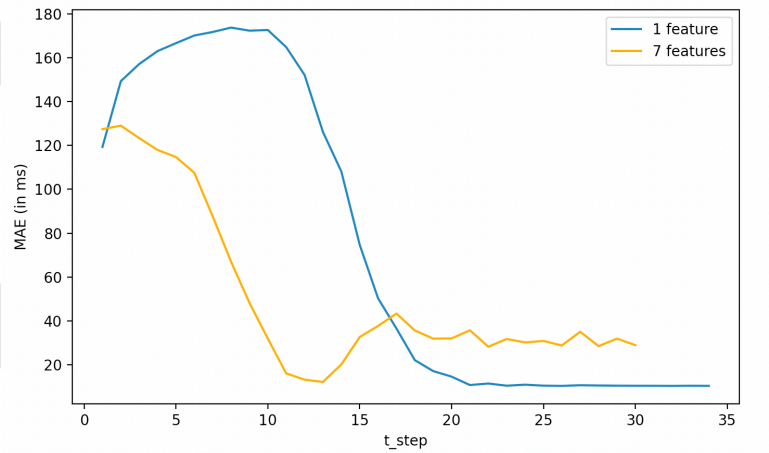


Figure 5.7: Mean Absolute Error evolution with varying t_{step} values for both sets

For the first set of features S_9 (in orange), the error is decreasing when t_{step} increases until

doi: 10.6342/NTU2022XXXXX

$t_{step} = 13s$. Then, the value is increasing and seems to reach a stable value. The value allowing the lowest error is retained, so $t_{step} = 13s$ in this case. Regarding the set S_{10} (in blue), the value is raising before decreasing and converging towards a minimum, which is obtained from $t_{step} = 22s$. Therefore, this value is taken for the model using this set. It can be observed that the second set, that only includes one feature, works better with a bigger t_{step} than the other set, composed of multiple features. Taking a large value of t_{step} increases the amount of information, as each feature is considered at each time step. However, having multiple features can bring too much information and be detrimental to the model's performance, leading to the observed results.

The results obtained with Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) models using one or several hidden layers and with both sets of features have been compared. The first comparison is made between the two sets of features. For each case, the smallest error between the models using the two sets is highlighted in green. Table 5.5 is gathering the errors (in ms) of each model using $H = 1$ hidden layer. For the deeper models, i.e. when $H > 1$ hidden layer, the results are shown on Table 5.6.

		Dataset 1	Dataset 2	Dataset 3
RNN	S_9	12.2	214	14.5
	S_{10}	10.5	92.1	14.1
LSTM	S_9	49.2	50.4	14.1
	S_{10}	46.5	9.49	14.4

Table 5.5: Table gathering the errors for models using one hidden layer

		Dataset 1	Dataset 2	Dataset 3
RNN	S_9	68.2	147	15.0
	S_{10}	11.1	32.9	14.0
LSTM	S_9	47.2	80.2	14.3
	S_{10}	58.5	9.94	14.4

Table 5.6: Table gathering the errors for models using several hidden layers

doi: 10.6342/NTU2022XXXXX

Out of the 12 comparisons, the model using the set S_{10} gives the best result 9 times. For the dataset 2, the gap between the two sets is big. Indeed, the set S_{10} has errors from 2 times to 8 times smaller than the models using the other set. That gap is also noticeable for dataset 1, with the deep RNN models, for which this set is more than 6 times more efficient. This difference can be seen in the Figure 5.8 that gathers the graphs of the prediction from the models using RNN with both sets. In this Figure, Set 1 (in orange) refers to the model using S_9 and Set 2 (in green) is for the model with S_{10}.

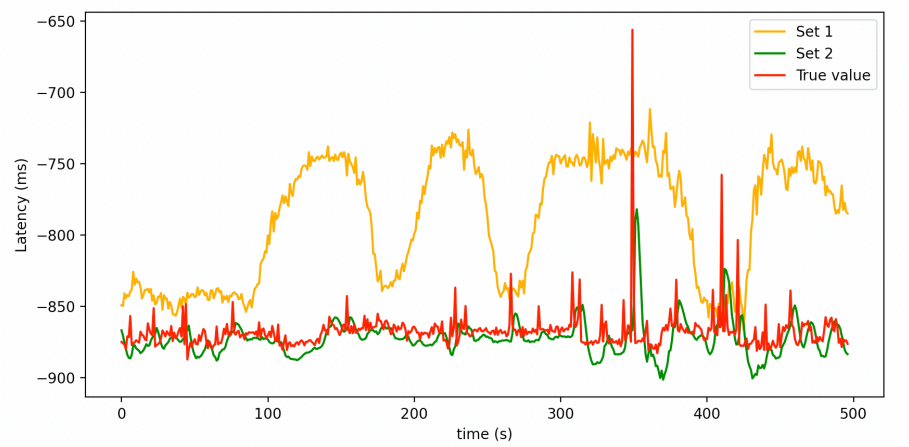


Figure 5.8: Prediction graphs using deep RNN for both sets on dataset 1

The predictions in green are very close to the true values whereas the model using the set S_9 produces predictions far from the target and results in a Mean Absolute Error of $147ms$. Therefore, using the set S_{10} is a better solution as it gives smaller errors. In the following comparisons, only the results obtained with this set will be taken into account. The two algorithms, Recurrent Neural Networks and Long Short-Term Memory can be compared and their errors (in ms) are gathered in Table 5.7.

		Dataset 1	Dataset 2	Dataset 3
1 layer	RNN	10.5	92.1	14.1
	LSTM	46.5	9.49	14.4
Deep model	RNN	11.1	32.9	14.0
	LSTM	58.5	9.94	14.4

Table 5.7: Table gathering the errors for the comparison between RNN and LSTM

For the dataset 3, the results are very close between the models using RNN and LSTM

doi: 10.6342/NTU2022XXXXX

(only $0.3ms$ and $0.4ms$ difference respectively for the 1 layer and deep models). For the dataset 1, the gap is wider. Indeed, the errors from RNN models are approximatevely five times smaller than those from LSTM models. On the contrary, the LSTM models are performing a lot better for the dataset 2. The error is more than 9 times bigger for the 1-layer RNN model and more than 3 times bigger for the deep RNN model. Hence it is hard to conclude on the best algorithm as they both perform very well on two datasets but poorly on one.

As we saw in the previous part when using simple Neural Networks to predict the latency of the signal, the dataset 2 presents very unusual values of latency in the testing set compared to the training set, which makes the predictions harder to achieve precisely. In this new experiment, the LSTM models proved that an extremely good accuracy was doable. However, these models performed bad on the dataset 1 and have slightly bigger errors on the dataset 3. The deep RNN model has small errors for the dataset 1 and the dataset 3 and achieved satisfactory accuracy with an error of $32.9s$, considering the wider range of latency values from this dataset. This model has an error only three times higher than LSTM for dataset 2, while performing more than five times better on dataset 1.
As regards the comparison between the 1-layer and deep models of the RNN algorithm, they give similar results for the datasets 1 and 3. However, the error is almost three times lower when using a deeper model on dataset 2. This comparison is illustrated with the graphs of Figure 5.9.

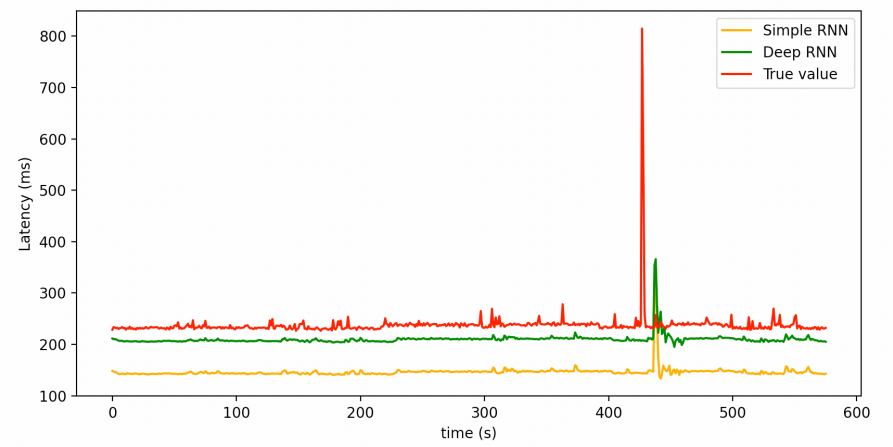


Figure 5.9: Comparison graphs between 1-layer and deep RNN on dataset 2

doi: 10.6342/NTU2022XXXXX

It clearly shows the difference between the two models, with the deeper model's predictions being noticeably closer to the actual values. Therefore, the best time-series model to use for the prediction of the latency appeard to be a Deep RNN model, using the set S_{10} of feature. With this configuration, the errors of the three datasets are respectively 11.1ms, 32.9ms and 14.0ms. Compared to the range of the latency values in these datasets (respectively 1659ms, 1808ms and 731ms), theses errors represents 0.7%, 1.8% and 1.9% of their range. The Deep RNN model achieved a prediction of the latency of the signal 10 seconds ahead with errors that represent less than 2% of the range for the three datasets.

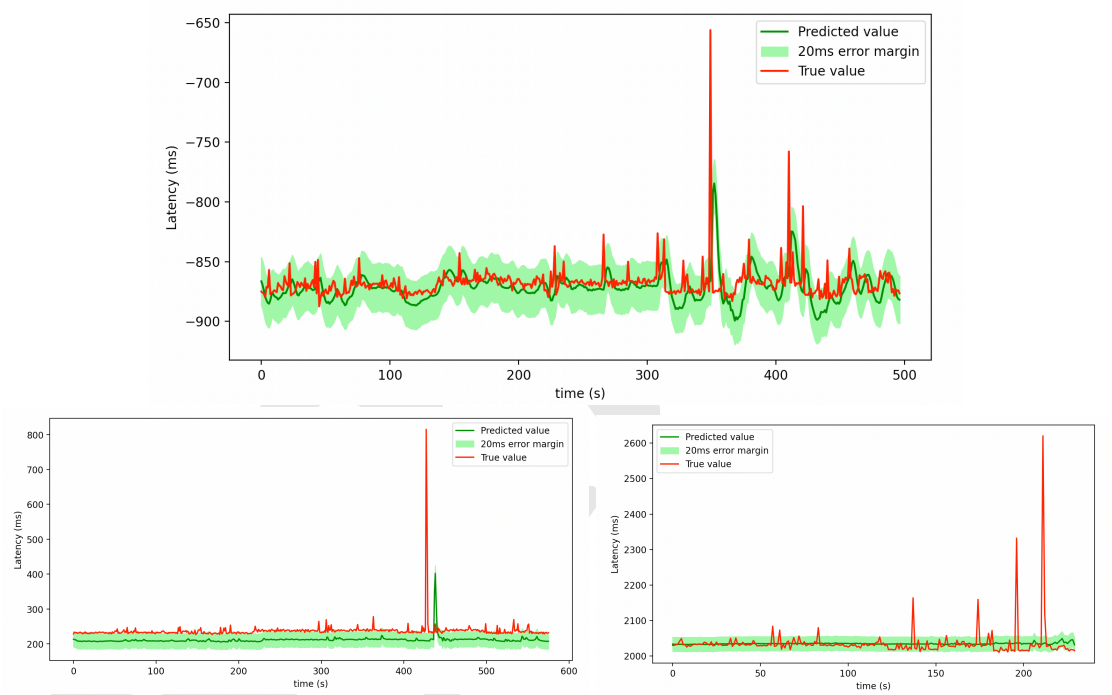


Figure 5.10: Latency prediction graphs for the three datasets

We can visualize the predictions of the chosen models for the three datasets in Figure 5.10. For the three datasets, the true values are almost always staying in the error margin. However, the graph for dataset 2 (located at the bottom left) shows that the actual values are at the limit of the error margin for the majority of the testing set. Overall, the model is giving satisfactory results with reliable predictions on each dataset.

doi: 10.6342/NTU2022XXXXX

5.2.2 ARIMA

ARIMA stands for AutoRegressive Integrated Moving Average. It is a model that predicts future values of time series, based on some statistical aspects of the observed data. This is a generalization of the ARMA model for non-stationary models, in the sense of the mean, i.e. when the mean of the data is changing over time. This characteristic is observed in the latency data, such as in dataset 2, where we saw that the means of training and testing sets are completely different. The ARIMA model addresses this non-stationarity by incorporating differencing operations to remove the changing mean.

This method is taking a single feature and forecast its future values on a chosen time, at each time step. Therefore, the set S_{10} that corresponds to the latency feature (n°10) is used and the model will predict the values for the next ten seconds. Only the prediction of the 10th second will be retained to match the prediction gap that was taken for the Recurrent Neural Networks models and ensure a reliable comparison between these two algorithms.

This algorithm needs to choose the values of three parameters: p, d and q. The process for dataset 2 will be taken as an example for the determination of these parameters values.

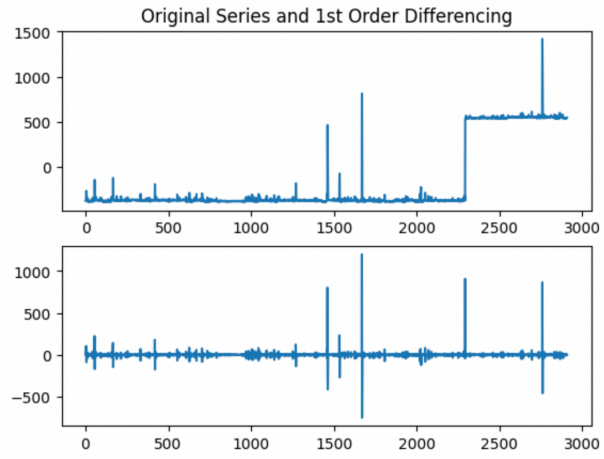


Figure 5.11: Graphs of the original series and 1st order differencing for the dataset 2

The order of differencing d has to be defined. Figure 5.11 shows that the series is not stationary as the mean is changing over time. It can be oberserved that the mean is not

doi: 10.6342/NTU2022XXXXX

changing anymore in the 1st order differencing graph. Therefore, $d = 1$ can be taken.

The next step is to find p, the order for the autoregressive model, by inspecting the partial autocorrelation plot, which measures the correlation between the time-series data and a certain lag. The partial autocorrelation graph is shown on Figure 5.12. The most significant lag is the first one. Thus, we have $p = 1$.

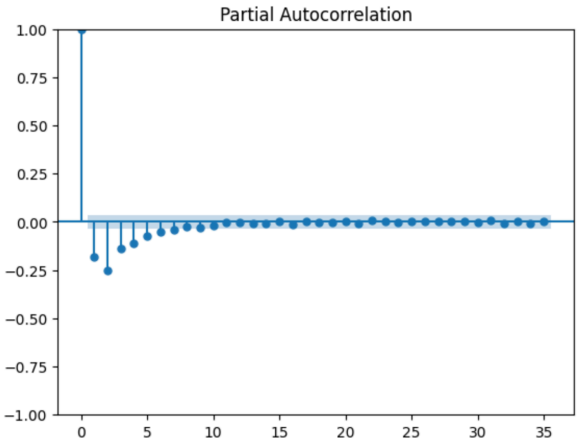


Figure 5.12: Partial autocorrelation graph of dataset 2

q is the number of lagged forecast errors in the prediction equation. It can be estimated by looking at the number of lags crossing the threshold on the autocorrelation graph of Figure 5.13. q can be set to 3.

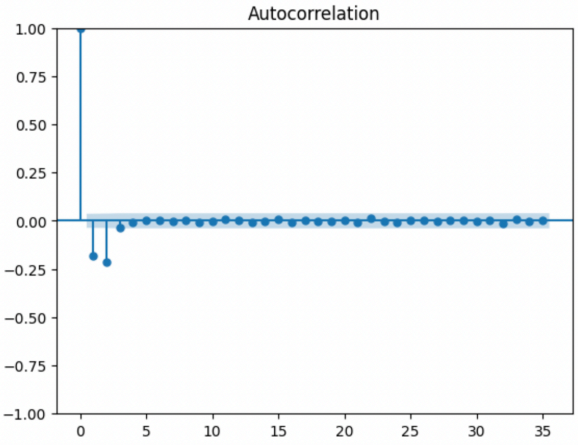


Figure 5.13: Autocorrelation graph of dataset 2

These operations are made on two of the three datasets to select the best parameters for the testing part. The chosen model is tested on all three testing sets, the two that served in

doi: 10.6342/NTU2022XXXXX

the parameters values selection and the third one that did not influence this decision and will assess the generalization ability and overall performance of the model.

The model presents a Mean Absolute Error of 8.92ms on dataset 1, 11.47ms on dataset 2 and 17.31ms on dataset 3. These three errors are low and can translate a good performance of the model.

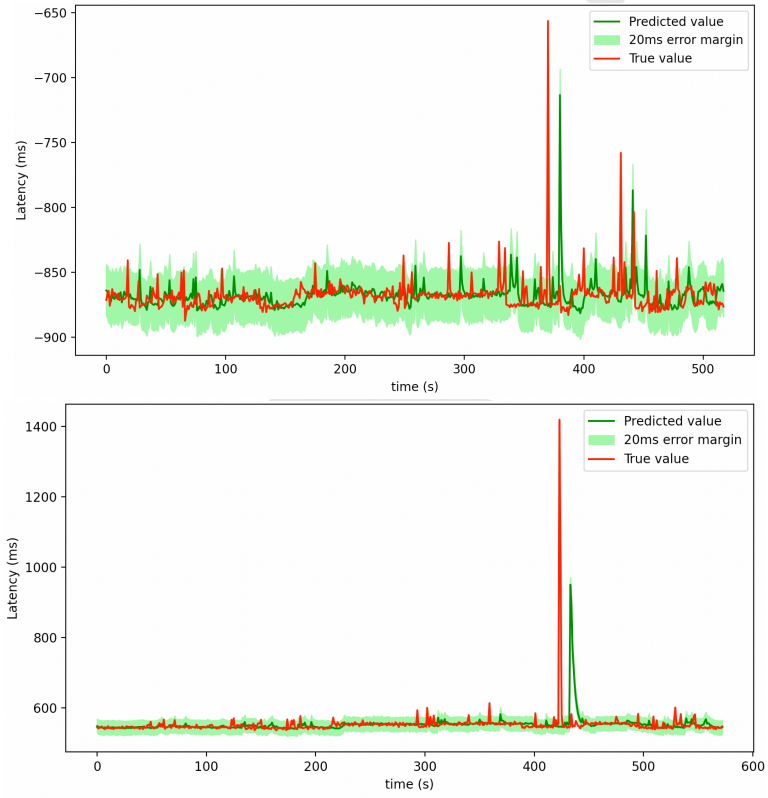


Figure 5.14: Graphs of the latency prediction with ARIMA on datasets 1 and 2

The graphs of the actual values and predictions for the datasets 1 and 2 can be seen on Figure 5.14. The same analysis can be made for both datasets. As expected with the low errors observed, the true values are almost always within the error margin of the predictions. The model is very reliable for these two datasets.

The results obtained on the third dataset are plotted on Figure 5.15. Although the Mean Absolute Error seemed to be good, the vizualisation of the predictions enables to see that the model predicted a constant value throughout the testing set. This result is not reliable and should be avoided.

doi: 10.6342/NTU2022XXXXX

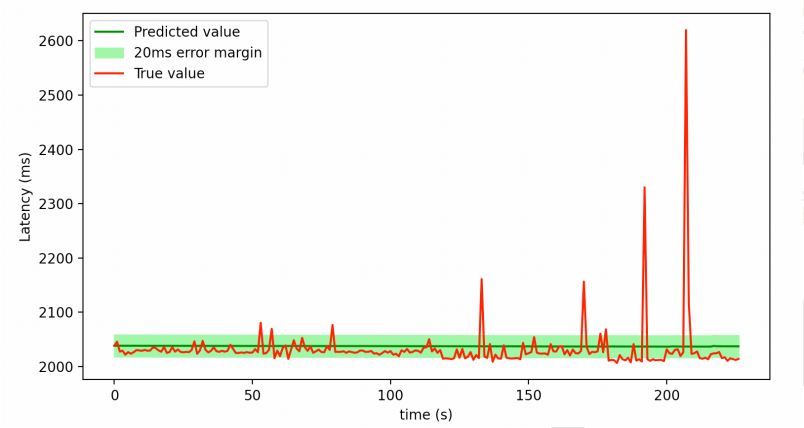


Figure 5.15: Graphs of the latency prediction with ARIMA on datasets 3

As mentionned before, the ARIMA algorithm forecasts future values of a time series at each step of a specified timeframe. In this application, predictions were generated on the next 10 seconds following the input, but only the final prediction for each input in the testing set is retained. The evolution of the predictions on this 10 seconds period for each sample is depicted in Figure 5.16. The first predictions exhibit a significant variability for the different samples. However the predictions are converging towards the same value around 2050ms for all samples. For all instances, the predictions become constant after the 8th step. Therefore, the ARIMA model produced constant and unchanging predictions on dataset 3 for the forecasting of the latency 10 seconds ahead.

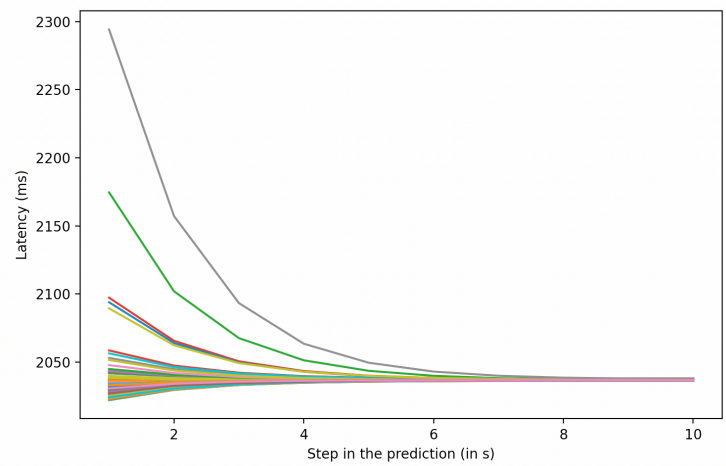


Figure 5.16: Graphs of the latency prediction with ARIMA on datasets 3

doi: 10.6342/NTU2022XXXXX

5.2.3 Conclusion

The results from the two retained models, the Deep RNN using the set S_{10} and the ARIMA algorithm with the same set, are expressed in ms in Table 5.8 and the lowest error for each dataset is highlighted in green.

	Dataset 1	Dataset 2	Dataset 3
RNN	10.46	23.67	14.13
ARIMA	8.92	11.47	17.31

Table 5.8: Table gathering the errors for the comparison between RNN and ARIMA

The ARIMA model seems to have better results, as its resulting errors are significantly lower for two datasets. Indeed the Mean Absolute Error is two times bigger for the RNN model on the dataset 2. However, the analysis of the predictions evolution on dataset 3 over the 10 seconds timeframe used in this application puts forward an important issue. The predictions are consistently converging towards a constant value and this makes the model unreliable. This model is usually used to forecast the future evolution of the time series, and it has been demonstrated that it struggles with single long-term predictions, as is the case in this study.

Therefore, the best model to predict the latency with a 10 seconds delay is the use of a Deep Recurrent Neural Networks algorithm, the set of feature S_{10} and the Mean Absolute Error as the error metric.

The chosen time series model can not be directly compared to the model previously studied with simple Neural Networks algorithm, as the latest did not take the 10 seconds delay for the prediction. Its use served as a foundation for insights and understanding to develop the time series model. It can still be noted that the time series model has significantly lower errors for two of the three datasets although the simple Neural Networks model had a simpler task by dismissing the forecasting delay and predicting the latency at t+1. The time series approach resulted in a reliable model that outperforms the first approach while adding complexity in the objective.

doi: 10.6342/NTU2022XXXXX

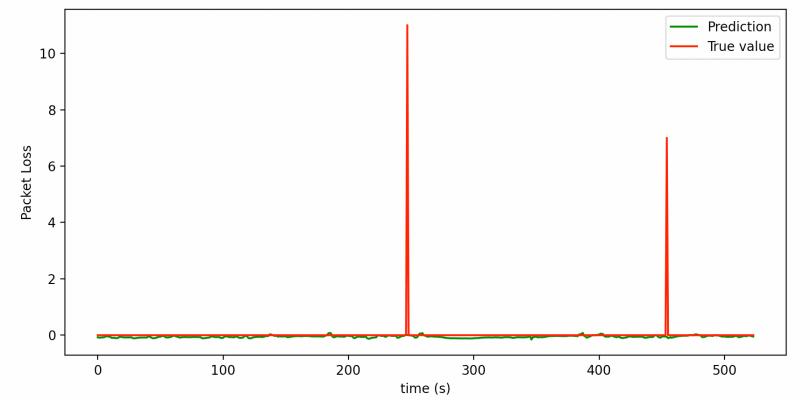
5.3 Packet Loss

The study of the number of packet loss prediction for the time series approach has been conducted with the same reasonning than for the simple Neural Networks model. The important issue to solve is the tendency of the models to always return a null value. This was succesfully treated in the previous experiments with the three tested methods. The two best ones appeared to be the use of NZ MAE as the error metric and the prediction of the feature n°17 designed for this application. These two methods will be tested on the Recurrent Neural Networks algorithm and their results will show if the same effects are percieved.

Similarly to the latency prediction, the problem is complexified by adding a 10 seconds delay for the prediction. The time series models have $t_{pred} = 10s$ and t_{step} value is defined in the same way than for the other parameters studied, with the cross validation method.

5.3.1 Results

First, a regression RNN algorithm taking the set S_7 to predict the feature n°11, which is the actual number of packet loss, is tested. NZ MAE is used to build the model. The use of this error metric during the Cross Validation phase to pick the model's parameters values did not have a significant impact on the model's performance.



doi: 10.6342/NTU2022XXXXX

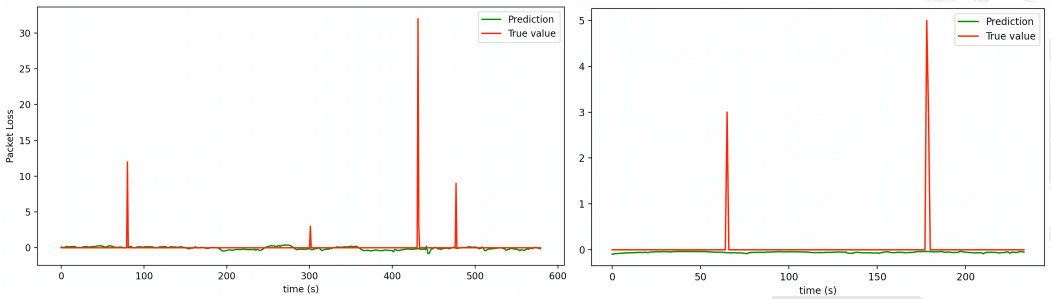


Figure 5.17: Graphs of the packet loss prediction with time series model using NZ MAE for the three datasets

Indeed, as illustrated in Figure 5.17, all the resulting predictions are very low for the three datasets. The green lines corresponding to the predictions are always around zero in the three graphs. This method did not have the desired effect and failed to prevent the selection of a model always predicting zero packet loss.

The other tested model is also built with the set of features S_7, but the output is the feature n°17 (the enlarged loss) and the error metric used to evaluate the model is the Mean Absolute Error. Figure 5.18 displays the graphs of the predictions and true values for the three datasets.

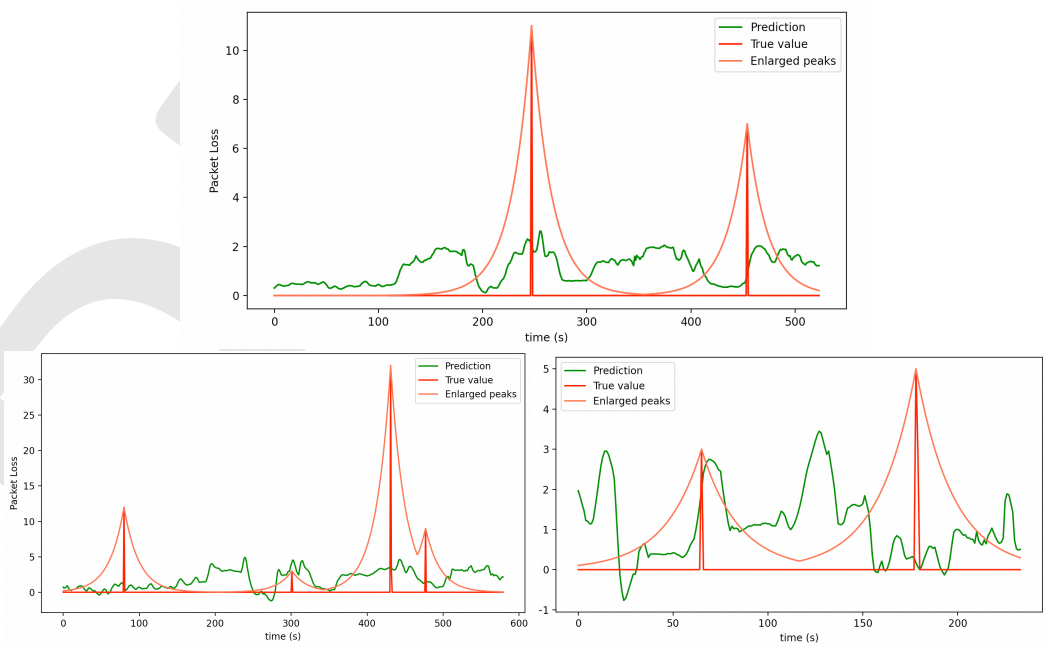


Figure 5.18: Graphs of the enlarged loss prediction with time series model for the three datasets

This method had more impact than the previous one, with more varying values. Indeed,

doi: 10.6342/NTU2022XXXXX

four distinct bumps can be identified, including two for the actual packet loss occurences. The model also mainly produced non-zero predictions with the other datasets, with half of their packet losses correctly detected. It should be noted that the values of the predictions are relatively low and may not accurately reflect the actual trend of the packet loss evolution in the datasets. The identified bumps are low and could be missed in real application.

5.3.2 Conclusion

As seen with the experiments, the use of NZ MAE did not lead to a reliable model. Indeed, all the resulting predictions are approximatevely zero. The same method was more succesful when applied with the simple Neural Networks algorithm, detecting several packet losses for each dataset. The extra complexity caused by the 10 seconds delay added between the input and the prediction can be a reason of this difference of performance. Besides, the time series input format, where each feature is considered for every time step, may also exmplain this discrepancy, as the increased amount of information can have a detrimental effect on the model's performance.

As discussed earlier, the model that was used for the prediction of the "enlarged loss" feature demonstrated better performance. However, the positive effect is less important than when it was applied with simple Neural Networks. Similarly to the first method, this difference can be explained by the 10 seconds delay in the prediction and the extra information brought by the time series input format.

Therefore, the simple Neural Networks algorithm should be used for this application. Combined with the use of the error metric NZ MAE or the prediction of the feature "enlarged loss", it appears to be a reliable solution to effectively forecast the packet loss with a large majority of null values in the datasets.

doi: 10.6342/NTU2022XXXXX

Chapter 6. Conclusions and Future Directions

To conclude, this study focused on the application of Machine Learning techniques to predict three telecommunication parameters, namely base station changes, the latency of the signal, and the number of packet loss, in the case of metro passengers itineraries. This work provided a comparison of various machine learning models to identify the most effective ones for real-time predictions. Moreover, it explored the prediction of these parameters a few seconds into the future, which provides useful information in real works application.

This research developed error metrics to meet the specific needs of this application. The models using it consistently outperformed the others, proving that these error metrics facilitated the selection of the most reliable models. Besides, novel approaches were proposed for predicting the base station changes, by taking the time before the next change, and estimating the number of packet losses, by considering a modified version of this parameter. This last approach is a novel solution to address the challenges posed by datasets predominantly composed of zero values.

It is important to acknowledge the limitations of this work. Firstly, the study did not include a dataset that had a substantial number of packet losses. It enabled to develop methods to adress the issue of having an overwhelming majority of zeros, but it could be interesting to investigate this other context. Secondly, even if the selection of the models was driven

doi: 10.6342/NTU2022XXXXX

by the specific requirements of the parameters, other Machine Learning algorithms could have been included in the study and could be considered in future research to complete this analysis. Finally, the proposed solutions to adress the issue of dealing with 99% of zeros in the data should be tested on other datasets and different contexts to validate the promising performance they showed.

Overall, this work has advanced the understanding of machine learning techniques in the context of predicting telecommunication parameters for passenger metro itineraries. By accurately forecasting the three parameters a few seconds in the future, the proposed models can help in the phone settings selection to optimize network performance and enhance the user experience. The findings and suggested solutions can serve as a foundation for further research in this field.

doi: 10.6342/NTU2022XXXXX

Bibliography

[1] Anna Giannakou, Dipankar Dwivedi and Sean Peisert, "A machine learning approach for packet loss prediction in science flows," *Future Generation Computer Systems*, vol. 102, pp. 190–197, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19305850

[2] Dr. Kalpana Saha (Roy) and Tune Ghosh, "Study of packet loss prediction using machine learning," *International Journal of Mobile Communication Networking*, vol. 11, pp. 1–11, 2020.

[3] Amir F. Atiya, Sung Goo Yoo, Kil To Chong and Hyongsuk Kim, "Packet loss rate prediction using the sparse basis prediction model," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 950–954, 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4182366

[4] Hooman Homayounfard, "Packet-loss prediction model based on historical symbolic time-series forecasting," Ph.D. dissertation, University of Techology of Sidney, 2013. [Online]. Available: http://hdl.handle.net/10453/24097

[5] Chun You and Kavitha Chandra, "Time series models for internet data traffic," in *Proceedings 24th Conference on Local Computer Networks. LCN'99*, 1999, pp. 164–171. [Online]. Available: https://ieeexplore.ieee.org/document/802013

[6] Rishabh Chauhan and Sunil Kumar, "Packet loss prediction using artificial intelligence unified with big data analytics, internet of things and cloud computing technologies," in *2021 5th International Conference on Information*

 doi: 10.6342/NTU2022XXXXX

Systems and Computer Networks (ISCON), 2021, pp. 01–06. [Online]. Available: https://ieeexplore.ieee.org/document/9702517

[7] Ali Safari Khatouni, Francesca Soro and Danilo Giordano, "A machine learning application for latency prediction in operational 4g networks," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 71–74. [Online]. Available: https://ieeexplore.ieee.org/document/8717807

[8] Jefferson Silva, Marcio Kreutz, Monica Pereira and Marjory Da Costa-Abreu, "An investigation of latency prediction for noc-based communication architectures using machine learning techniques," *Journal of Supercomputing*, pp. 1–19, 2019. [Online]. Available: http://shura.shu.ac.uk/25392/

[9] Robert Beverly, Karen Sollins and Arthur Berger, "Svm learning of ip adress strucure for latency prediction," in *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, ser. MineNet '06, 2006, pp. 299–304. [Online]. Available: https://doi.org/10.1145/1162678.1162682

[10] Bruno Astuto Arouche Nunes, Kerry Veenstra, William Ballenthin, Stephanie Lukin and Katia Obraczka, "A machine learning framework for tcp round-trip time estimation," *EURASIP Journal on Wireless Communications and Networking*, no. 47, 2014. [Online]. Available: https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/1687-1499-2014-47

[11] Desta Haileselassie Hagos, Paal E. Engelstad, Anis Yazidi and Carsten Griwodz, "A deep learning approach to dynamic passive rtt prediction model for tcp," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, 2019, pp. 1–10. [Online]. Available: https://ieeexplore.ieee.org/document/8958727

[12] Salman Memon and Muthucumaru Maheswaran, "Using machine learning for handover optimization in vehicular fog computing," in *Proceedings of the 34th*

 doi: 10.6342/NTU2022XXXXX

ACM/SIGAPP Symposium on Applied Computing, ser. SAC '19, 2019, p. 182–190. [Online]. Available: https://doi.org/10.1145/3297280.3297300

[13] Hadeel Abdah, Joao Paulo Barraca and Rui L. Aguiar, "Handover prediction integrated with service migration in 5g systems," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/9149426

[14] Le Luong Vy, Li-Ping Tung and Bao-Shuh Paul Lin, "Big data and machine learning driven handover management and forecasting," in *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2017, pp. 214–219. [Online]. Available: https://ieeexplore.ieee.org/document/8088624

[15] James A. Green, "Too many zeros and/or highly skewed? a tutorial on modelling health behaviour as count data with poisson and negative binomial regression," *Health psychology and behavioral medicine 2021*, vol. 9, no. 1, pp. 436–455, 2021. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/21642850.2021.1920416

[16] Rob J. Hyndman and Anne B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006. [Online]. Available: https://doi.org/10.1016/j.ijforecast.2006.03.001

[17] Alexei Botchkarev, "A new typology design of performance metrics to measure errors in machine learning regression algorithms," *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 14, pp. 45–79, 2019. [Online]. Available: https://doi.org/10.28945/4184

 doi: 10.6342/NTU2022XXXXX

Appendix A: 附錄標題一

			Dataset 1	Dataset 2	Dataset 3
MAE	NN	MAE	26.4	41.9	9.72
		MAE 60	7.95	10.3	9.72
		MAE 30	6.53	6.71	7.75
	DNN	MAE	28.8	66.4	10.8
		MAE 60	10.4	11.9	10.8
		MAE 30	7.60	6.38	7.74
MAE 60	NN	MAE	26.7	68.6	9.27
		MAE 60	9.06	12.7	9.07
		MAE 30	6.61	7.26	6.85
	DNN	MAE	24.8	82.3	8.50
		MAE 60	7.24	12.7	8.50
		MAE 30	6.03	9.93	6.27
MAE 30	NN	MAE	26.4	41.9	9.72
		MAE 60	7.95	10.3	9.72
		MAE 30	6.53	6.71	7.75
	DNN	MAE	24.8	47.9	9.86
		MAE 60	7.70	10.9	9.80
		MAE 30	5.99	5.31	7.70

Table A.1: Table gathering the errors from all the models tested to predict the time before the next change of base station with NN, using S_1

doi: 10.6342/NTU2022XXXXX

Appendix B: 附錄標題二

			Dataset 1	Dataset 2	Dataset 3
MAE	NN	MAE	37.1	62.6	8.50
		MAE 60	15.9	11.5	8.50
		MAE 30	8.00	6.45	6.73
	DNN	MAE	26.2	64.1	9.49
		MAE 60	9.52	14.4	9.49
		MAE 30	6.94	7.66	7.66
MAE 60	NN	MAE	37.4	62.2	8.49
		MAE 60	16.2	11.6	8.49
		MAE 30	8.20	6.54	6.72
	DNN	MAE	25.8	82.2	8.50
		MAE 60	8.12	12.0	8.50
		MAE 30	6.77	5.50	6.71
MAE 30	NN	MAE	34.1	72.0	8.35
		MAE 60	16.9	12.5	8.34
		MAE 30	10.2	7.15	6.65
	DNN	MAE	26.0	77.4	10.3
		MAE 60	8.25	12.4	10.3
		MAE 30	6.93	5.92	7.01

Table B.1: Table gathering the errors from all the models tested to predict the time before the next change of base station with NN, using S_2

doi: 10.6342/NTU2022XXXXX

學位口試參考 SOP

(本表以電信所、電機所為例，且不保證符合最新規定，行政相關流程請再次確認相關單位要求)

研究進度

1. 儘早完成研究，研究做完，開始寫論文。
2. 取得指導教授同意畢業口試。
3. *口試前一個月 (June)*完成論文初稿。
4. *口試前兩週*完成內部報告 Rehersal。
   請先個人計時練習報告。
   碩士班報告時長 30 分鐘，不含 Q&A。
   博士班報告時長 60 分鐘，不含 Q&A。
5. 每週向指導教授更新論文最新版本。
6. 本論文模板要確認相關資訊填寫正確。

口試前準備流程

1. *畢業學期初 (April)*在 myNTU 申請學位考試。向系所繳交*應屆畢業生成績審核表*，畢業當學期要選課*碩士*論文或*博士*論文。
2. 依照系所規定繳交*論文題目*及*論文大綱*。
3. *注意收信*，密切關注信箱內與畢業流程相關之公告。
4. *口試前兩週*向系所繳交*考試委員名冊*。
   - ***除非所辦要求，否則不要主動向口委要求任何個資。***
   - 以預計考試日期回推，需要在*預定口試日期前 1 個月*請指導教授邀請口試委員。
   - 在請指導教授邀請口試委員之前，必須先完成論文初稿。
   - 一般而論，指導教授親自協調口試時間較有禮貌。但如果有需要由畢業生來協調，請務必注意 Email 禮儀及禮貌用語。*所有與考試委員的信件需要副本指導教授。*
   - 在訂出口試日期，並預約好口試的會議室 (需要多預約當天預定考試時間前的 1 小時做準備) 之後，需要寄信通知所有的考試委員*日期、時間、地點、線上口試連結 (若無則免)、完整的口試委員名單*。
   - Email 要夾帶論文初稿。
   - Email 禁止使用密件副本。避免使用群發郵件與考試委員聯繫。
   - Email 屬名附上詳細聯繫方式。
5. 詢問口委是否需要先看紙本論文，若無回應則預設需要郵寄紙本。若需要的話，可連同聘函、邀請函郵寄至口委所在單位辦公室地址，或是在 Email 詢問收件地址。聘函 (線上參與者免)、邀請函、公文紙袋向所辦領取。有些系所邀請函要自行列印。台大小福郵局寄送論文有打折。
6. *考試前幾天*自行列印或向系所索取相關文件，並在當天考試開始前將以下資料給指導教授。
   - *領據* 依實體參與口試委員人數 (不含指導教授) 向系所領取相應份數。
   - *停車券* 依實體參與口試委員人數 (不含指導教授) 向系所領取相應份數。可事先調查口試委員實際開車的情形，但一般來說多拿並無大礙。
   - *口試委員審定書* 即為本樣板第 2 頁，總共一份自行列印。列印之前先加上浮水印並確認 doi 輸入正確。
   - *學位審查意見評分表* 依實體參與口試委員人數 (含指導教授) 自行列印。可使用本樣板輸出之版本或是按照系所規定格式，如果不確定請先與系所確認。
   - *口試紀錄表* 總共一份自行列印。可使用本樣板輸出之版本或是按照系所規定格式，如果不確定請先與系所確認。

口試前一到兩天

1. 寄信提醒所有口委 (包含指導教授) 口試時間、地點 (線上連結)、各口委參與方式 (使否線上)、論文題目、最終版論文草稿附件、口試投影片。多人一起口試的話，要標明口試順序。
2. 準備足份的紙本論文。可黑白。
3. 準備足份的紙本投影片。A4 一面兩頁投影片。可黑白。
4. 和所辦借預定的口試場地，提前 (不可當日才測試) 測試實際口試場地的所有設備。網路盡量要有線網路。
5. 多人一起口試請協調共用同一台筆電來報告。
6. 訂點心、飲料、過中午要訂便當 (可先問指導教授是否需要)。
7. 確認當天的場地借用，使否有預借包含口試開始前 1 小時的時間來完成場地預備。

口試當天

1. 著正裝
2. 足量的筆 (確認有水)
3. 衛生紙
4. 飲料、食物
   - 自費
   - (足份 +1) 瓶裝水
   - (足份 +1) 茶
   - (足份 +1) 咖啡 or 果汁
   - (足份 +1) 水果盒
   - (足份 +1) 便當 (可先問指導教授是否需要)+ 餐具
   - (足份 +1) 小點心 or 餐盒
   - 糖包、奶精、攪拌棒、吸管
5. (足份 +1) 紙本論文、投影片
6. 筆電、簡報筆、螢幕轉接頭、電源線
7. 音響、麥克風 (若無線上參與者則免)
8. 請一位同學協助記錄口試當天的過程，主要包含口委的意見、提問、建議等。作為後續修改論文的依據。
9. 避免進出，口試開始要關門。
10. 口試開始前將所有相關文件、表格給指導教授。
11. *如果口委有提出修改論文標題的要求，請馬上到實驗室重印新的論文審定書後再給口委簽名。否則後續會需要多花時間重新完成線上的論文審定書電子簽名。*

口試後

1. 依照紀錄完成論文的修改。修改完成後寄給指導教授和所有的口委。
2. 所辦在論文審定書完成所長簽名後會通知領取。掃描成電子檔之後連同論文上傳至圖書館審查系統`http://www.lib.ntu.edu.tw/node/103`。審查約需 2 至 4 個工作天。
3. 完成圖書館審查之後，可以送印論文。完成送印後，先到離校系統`https://my.ntu.edu.tw/StudLeave/Login.aspx`點選圖書館申請清查，再到圖書館 1F 櫃台繳交。
   - 指導教授 x1 (精裝)
   - 圖書館 x2 (精平裝皆可，部分系所需 3 冊)
   - 實驗室 x1 (精平裝皆可)
   - 可詢問口委是否需要紙本完稿論文。
   - 博士班精裝封面黑底燙金字，碩士班精裝封面紅底燙金字，內頁 A4 80 磅白紙，平裝封面 200 磅淺色銅西卡紙或雲彩紙上亮 P。
4. 完成系所各自要求的線上離校系統程序。
   - 包含指導教授離校單簽名。
   - 依規定繳交論文原創性聲明。如需要跑分，請找指導教授 (或是當過助教的同學) 協助使用 NTU Cool 產生原創性比對報告。
5. 完成實驗室交接流程 (交接光碟、程式碼、各類文件檔案、財產清點、薪資結清)。
6. 至研教組辦理離校。若要同時申請英文學位證書須至機器先繳納 100 元申請費。如果離校之後才申請的話是拿補發版。

國立臺灣大學電機資訊學院電信工程學研究所
碩士論文
Graduate Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

Using Machine Learning to Predict 3 Telecommunication Parameters in the Case of Metro Passengers Itineraries

Alexandre Benayoun

指導教授：魏宏宇 博士
Advisor: Hung-Yu Wei, Ph.D.

中華民國 111 年 9 月
September, 2022

國立臺灣大學
電信工程學研究所

碩士論文

撰

111
9

國立臺灣大學電信工程學研究所
學位論文學術倫理暨原創性聲明書

(111.7.25 經 110 學年度第九次學術委員會通過)

本人已經自我檢核，確認無違反學術倫理情事，論文倘有造假、變造、抄襲、由他人代寫，或涉其他一切有違著作權及學術倫理之情事，及衍生相關民、刑事責任，概由本人負責，概無異議。

聲明人：_____()

學號：R10942164

中華民國 111 年 8 月 3 日

指導教授確認簽章：

_____

National Taiwan University
Graduate Institute of Communication Engineering
Statement of Academic Ethics and Originality

(Approved by the ninth academic committee of the 110 academic years on July 25, 2022)

As the author of this thesis/dissertation, I adhere to the principles of academic ethics. I confirm that I have checked my thesis/dissertation and there is no violation of academic ethics. I will be responsible for any falsification, alteration, plagiarism, thesis or dissertation written by someone else, or any other violation of copyright and academic ethics, as well as any related civil or criminal liability.

Student Name (Please Print Clearly and Signature):

(Alexandre Benayoun)

Student Number: R10942164

Date: August 3, 2022

Signature of Advisor of Thesis:

國立臺灣大學碩士應屆畢業生成績審核表
Academic Achievement Record for Graduate Students
（請注意碩士生須修業達第二學始可申請學位考試）

所組代碼：9420 Institute/Division Code　　所別：電信工程學研究所 Department/Institute of study　　組別：通訊與信號處理組 Division

學年：110 Academic Year　　學期：2 Semester　　學號：R10942164 Student ID No.　　姓名： Name

審核日期：民國：　　年　　月　　日
Date (Year/Month/Day)

學生已依本所之修業規定，修習本組相關課程學分至少 12 學分，同意申請學位考試。

指導教授簽章：_____
Signature of Advisor

審核人簽章：_____
Processed by

所長簽章：_____
Signature of Head of Institute

教務單位承辦人簽章：_____
Signature of Office of Academic Affairs Staff in charge

表單編號：A404000-2-042A-04

110 學年度第二學期電信工程學研究所碩士學位考試委員名冊

碩士班研究生	考　試　委　員				
	校內外	姓名	服務單位及職稱	服務單位所在縣市	備註
學號： R10942164 姓名： 預定口試日期： 111 年 9 月 12 日	內	魏宏宇	台大電信所 教授	台北市	*
	外	張仲儒	陽明交大電機系 榮譽退休教授	新竹市	

說明：

1. **系所請於研究生學位考試委員人選確定時，填寫聘函逕發各考試委員，並繕造本表，於報領考試委員審查費及交通費時，一併送交教務承辦單位(研教組、醫教分處)，以利核對。**
2. **請於備註欄以『＊』註明指導教授。**
3. **碩士學位考試委員三至五人，校內、外委員比例不限，惟因校內、外委員之酬勞不同，故仍請註明校內或校外。**
   兼任教師得視為校外委員，惟兼任教師若擔任該候選人之指導教授時，則應視為『校內』考試委員。
4. **有關碩士學位考試委員，敬請參考本校『博士暨碩士學位考試規則』規定辦理。**
5. **本表核畢後即於當學期銷毀，如有需要請系所自行影印留存。**

指導教授簽章：＿＿＿＿＿＿＿＿＿＿　＿＿＿年＿＿＿月＿＿＿日

所　　長簽章：＿＿＿＿＿＿＿＿＿＿　＿＿＿年＿＿＿月＿＿＿日

國立臺灣大學電機資訊學院電信工程學研究所
Graduate Institute of Communication Engineering, National Taiwan University
No. 1, Sec. 4 Roosevelt Road, Taipei 10617, Taiwan·TEL: +886-2-33663075·FAX: +886-2-23683824

魏宏宇博士　道鑑：

研究生（指導教授：魏宏宇教授）碩士學位論文口試，謹訂於111年9月12日（星期一）上午10時00分起假本校電機二館103室舉行（線上連結：https://meet.google.com/abc-defg-hij）。茲檢附委員聘函及研究生論文初稿，敬請　屆時蒞臨本所共同主持，毋任感荷。

口試題目：
口試時間：**111年9月12日（星期一）上午10時00分**
口試地點：**本校電機二館103室（線上連結：https://meet.google.com/abc-defg-hij）**

肅此　順頌
時祺

國立臺灣大學電信工程學研究所　敬啟
中華民國112年6月28日

附註：

1. 因本校實行車輛管制，自行開車之委員請務必攜帶本通知函及聘函代替通行證，謝謝。
2. 依台大核銷規定，報支校外委員臨時薪資（出席費、車馬費、演講費、論文審查費……）一律以郵局帳戶辦理，如確無郵局帳戶，以其他金融機構帳戶辦理付款。敬請教授出席本所學生論文口試時，務請協助提供存摺封面影本，俾便辦理核銷匯款；如有不便，敬請見諒。

國立臺灣大學電機資訊學院電信工程學研究所
Graduate Institute of Communication Engineering, National Taiwan University
No. 1, Sec. 4 Roosevelt Road, Taipei 10617, Taiwan·TEL: +886-2-33663075·FAX: +886-2-23683824

張仲儒博士　道鑑：

研究生（指導教授：魏宏宇教授）碩士學位論文口試，謹訂於111年9月12日（星期一）上午10時00分起假本校電機二館103室舉行（線上連結：https://meet.google.com/abc-defg-hij）。茲檢附委員聘函及研究生論文初稿，敬請　屆時蒞臨本所共同主持，毋任感荷。

口試題目：
口試時間：**111年9月12日（星期一）上午10時00分**
口試地點：**本校電機二館103室（線上連結：https://meet.google.com/abc-defg-hij）**

肅此　順頌
時祺

國立臺灣大學電信工程學研究所　敬啟
中華民國112年6月28日

附註：

1. 因本校實行車輛管制，自行開車之委員請務必攜帶本通知函及聘函代替通行證，謝謝。
2. 依台大核銷規定，報支校外委員臨時薪資（出席費、車馬費、演講費、論文審查費……）一律以郵局帳戶辦理，如確無郵局帳戶，以其他金融機構帳戶辦理付款。敬請教授出席本所學生論文口試時，務請協助提供存摺封面影本，俾便辦理核銷匯款；如有不便，敬請見諒。

國立臺灣大學電機資訊學院電信工程學研究所
Graduate Institute of Communication Engineering, National Taiwan University
No. 1, Sec. 4 Roosevelt Road, Taipei 10617, Taiwan·TEL: +886-2-33663075·FAX: +886-2-23683824

110 學年度碩士學位審查意見評分表

論文題目：

作者姓名：

評分項目	百分比	分數	評語
內容充實性	30%	分	
觀點與創見	30%	分	
文字與組織	25%	分	
參考資料	15%	分	
總分	100%	分	

總評語：

對本案之整體觀點：(請勾選)	A+	A	A-	B+	B	B-	C+ 以下
	100-90 分	89-85 分	84-80 分	79-77 分	76-73 分	72-70 分	70 分以下

審查人（簽章）：　　111 年 9 月 12 日

國立臺灣大學電機資訊學院電信工程學研究所
Graduate Institute of Communication Engineering, National Taiwan University
No. 1, Sec. 4 Roosevelt Road, Taipei 10617, Taiwan·TEL: +886-2-33663075·FAX: +886-2-23683824

110 學年度碩士學位審查意見評分表

論文題目：

作者姓名：

評分項目	百分比	分數	評語
內容充實性	30%	分	
觀點與創見	30%	分	
文字與組織	25%	分	
參考資料	15%	分	
總分	100%	分	

總評語：

對本案之整體觀點：(請勾選)	A+	A	A-	B+	B	B-	C+ 以下
	100-90 分	89-85 分	84-80 分	79-77 分	76-73 分	72-70 分	70 分以下

審查人（簽章）：　　111 年 9 月 12 日

國立臺灣大學研究所碩士學位考試試卷 (口試紀錄表)
Thesis Defense Grade Sheet

110 學年度 (Year) 第 2 學期 (Semester)　　考試日期 (Date)：111 年 9 月 12 日

系所組別 (Department)：　　考試地點 (Place)：電機二館 103 室

電信工程學研究所 通訊與信號處理組　　記錄 (Recorder)：_____

學號 (ID Number)：R10942164

姓名 (Name)：

論文題目 (Title of Thesis)：

※ 學位考試成績： ☐ **A+** ； ☐ **A** ； ☐ **A-** ； ☐ **B+** ； ☐ **B** ； ☐ **B-** ；

☐ **C+** ； ☐ **C** ； ☐ **C-** ； ☐ **F** ； ☐ **X** ；

(請勾選成績，塗改請核章。成績評量定義詳見下列說明，研究生及格標準為 B-。)
(Please tick the grades. Please refer to the following description for the definition of grades. The passing standard for graduate students is B-.)

※ 本委員會確認學位論文是否符合專業領域： ☐ 是 (Yes) ☐ 否 (No)
(The committee confirms whether the thesis/dissertation meets the professional field)

考試委員簽章 (Sign of committee members)：

※ 成績評量定義：Definition of Grades

A+	**90-100**	所有目標皆達成且超越期望（All goals achieved beyond expectation）
A	**85-89**	所有目標皆達成（All goals achieved）
A-	**80-84**	所有目標皆達成，但需一些精進（All goals achieved, but need some polish）
B+	**77-79**	達成部分目標，且品質佳（Some goals well achieved）
B	**73-76**	達成部分目標，但品質普通（Some goals adequately achieved）
B-	**70-72**	達成部分目標，但有些缺失（Some goals achieved with minor flaws）
C+	**67-69**	達成最低目標（Minimum goals achieved）
C	**63-66**	達成最低目標，但有些缺失（Minimum goals achieved with minor flaws）
C-	**60-62**	達成最低目標但有重大缺失（Minimum goals achieved with major flaws）
F	**59(含) 以下**	未達成最低目標（Minimum goals not achieved）
X	**0**	因故不核予成績（Not graded due to unexcused absences or other reasons）

圖書館審核之複製用目錄列表。輸出檔案因保全設定無法複製，請直接在 Overleaf 的 PDF 檢視器複製。

圖書館審核之複製用參考文獻列表。輸出的檔案因保全設定無法複製，請直接在 Overleaf 的 PDF 檢視器複製。

[1] Anna Giannakou, Dipankar Dwivedi and Sean Peisert, "A machine learning approach for packet loss prediction in science flows," *Future Generation Computer Systems*, vol. 102, pp. 190–197, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19305850

[2] Dr. Kalpana Saha (Roy) and Tune Ghosh, "Study of packet loss prediction using machine learning," *International Journal of Mobile Communication Networking*, vol. 11, pp. 1–11, 2020.

[3] Amir F. Atiya, Sung Goo Yoo, Kil To Chong and Hyongsuk Kim, "Packet loss rate prediction using the sparse basis prediction model," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 950–954, 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4182366

[4] Hooman Homayounfard, "Packet-loss prediction model based on historical symbolic time-series forecasting," Ph.D. dissertation, University of Techology of Sidney, 2013. [Online]. Available: http://hdl.handle.net/10453/24097

[5] Chun You and Kavitha Chandra, "Time series models for internet data traffic," in *Proceedings 24th Conference on Local Computer Networks. LCN'99*, 1999, pp. 164–171. [Online]. Available: https://ieeexplore.ieee.org/document/802013

[6] Rishabh Chauhan and Sunil Kumar, "Packet loss prediction using artificial intelligence unified with big data analytics, internet of things and cloud computing technologies," in *2021 5th International Conference on Information Systems and Computer Networks (ISCON)*, 2021, pp. 01–06. [Online]. Available: https://ieeexplore.ieee.org/document/9702517

[7] Ali Safari Khatouni, Francesca Soro and Danilo Giordano, "A machine learning application for latency prediction in operational 4g networks," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 71–74. [Online]. Available: https://ieeexplore.ieee.org/document/8717807

[8] Jefferson Silva, Marcio Kreutz, Monica Pereira and Marjory Da Costa-Abreu, "An investigation of latency prediction for noc-based communication architectures using machine learning techniques," *Journal of Supercomputing*, pp. 1–19, 2019. [Online]. Available: http://shura.shu.ac.uk/25392/

[9] Robert Beverly, Karen Sollins and Arthur Berger, "Svm learning of ip adress strucure for latency prediction," in *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, ser. MineNet '06, 2006, pp. 299–304. [Online]. Available: https://doi.org/10.1145/1162678.1162682

[10] Bruno Astuto Arouche Nunes, Kerry Veenstra, William Ballenthin, Stephanie Lukin and Katia Obraczka, "A machine learning framework for tcp round-trip time estimation," *EURASIP Journal on Wireless Communications and Networking*, no. 47, 2014. [Online]. Available: https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/1687-1499-2014-47

[11] Desta Haileselassie Hagos, Paal E. Engelstad, Anis Yazidi and Carsten Griwodz, "A deep learning approach to dynamic passive rtt prediction model for tcp," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, 2019, pp. 1–10. [Online]. Available: https://ieeexplore.ieee.org/document/8958727

[12] Salman Memon and Muthucumaru Maheswaran, "Using machine learning for handover optimization in vehicular fog computing," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '19, 2019, p. 182–190. [Online]. Available: https://doi.org/10.1145/3297280.3297300

[13] Hadeel Abdah, Joao Paulo Barraca and Rui L. Aguiar, "Handover prediction integrated with service migration in 5g systems," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/9149426

[14] Le Luong Vy, Li-Ping Tung and Bao-Shuh Paul Lin, "Big data and machine learning driven handover management and forecasting," in *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2017, pp. 214–219. [Online]. Available: https://ieeexplore.ieee.org/document/8088624

[15] James A. Green, "Too many zeros and/or highly skewed? a tutorial on modelling health behaviour as count data with poisson and negative binomial regression," *Health psychology and behavioral medicine 2021*, vol. 9, no. 1, pp. 436–455, 2021. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/21642850.2021.1920416

[16] Rob J. Hyndman and Anne B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006. [Online]. Available: https://doi.org/10.1016/j.ijforecast.2006.03.001

[17] Alexei Botchkarev, "A new typology design of performance metrics to measure errors in machine learning regression algorithms," *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 14, pp. 45–79, 2019. [Online]. Available: https://doi.org/10.28945/4184